CAVAEva: an engineering platform for evaluating commercial anti-malware applications on smartphones

Jiang, Hao; Meng, Weizhi; Su, Chunhua; Choo, Kim-Kwang Raymond; Liu, Zhe; Yung, Moti

Published in:
Information Security and Cryptology

Link to article, DOI:
[10.1007/978-3-030-42921-8_12](https://doi.org/10.1007/978-3-030-42921-8_12)

Publication date:
2020

Document Version
Peer reviewed version

[Link back to DTU Orbit](#)

Citation (APA):
Jiang, H., Meng, W., Su, C., Choo, K-K. R., Liu, Z., & Yung, M. (Ed.) (2020). CAVAEva: an engineering platform for evaluating commercial anti-malware applications on smartphones. In *Information Security and Cryptology* (pp. 208-24). Springer. Lecture Notes in Computer Science Vol. 12020 https://doi.org/10.1007/978-3-030-42921-8_12

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

CAVAEva: An Engineering Platform for Evaluating Commercial Anti-malware Applications on Smartphones

Hao Jiang^{1,2}, Weizhi Meng^{2,3(✉)}, Chunhua Su⁴,
and Kim-Kwang Raymond Choo⁵

¹ CyberTech, Hong Kong, China

² Department of Applied Mathematics and Computer Science,
Technical University of Denmark, Kongens Lyngby, Denmark
weme@dtu.dk

³ Department of Computer Science, Guangzhou University, Guangzhou, China

⁴ The University of Aizu, Aizuwakamatsu, Japan

⁵ The University of Texas at San Antonio, San Antonio, USA

Abstract. The pervasiveness of mobile devices, such as Android and iOS smartphones, and the type of data available and stored on these devices make them an attractive target for cyber-attackers. For example, mobile malware authors seek to compromise devices to collect sensitive information and data from the smartphones. To mitigate such a threat, a number of online scanning platforms exist to evaluate existing anti-malware applications. However, existing platforms have a number of limitations, such as configuration inflexibility. Also, in practice, the code protection and different structures complicate efforts to effectively evaluate different commercial anti-malware software in a configurable and unified platform. Hence in this work, we design *CAVAEva*, an engineering platform for commercial anti-malware application evaluation, in which users/researchers have the capability to configure the platform based on their needs and requirements. In particular, we show how to design such a platform and introduce its performance. Specifically, we present a comparative summary of seven commercial anti-malware software, and collect the feedback from a user study. Experimental results demonstrate the potential utility of our platform in evaluating commercial anti-malware software in a real-world smartphone deployment.

Keywords: Mobile device · Smartphone security · Malware detection · Evaluation platform · Anti-malware software

1 Introduction

Smartphones, such as iPhone 11 and Galaxy S10+, are becoming more computationally capable (including storage capacities), and the amount and range of data stored on or accessible from such devices are also increasing, such as personal

and sensitive information [1, 11]. Hence, it is not surprising that these devices have become an attractive target for cyber-attackers, ranging from script kiddies to state-sponsored actors (e.g., advanced persistent threat (APT) actors). Malware remains one of the key threats to mobile device security, which can collect and infer sensitive information and data from smartphones [45, 47], as evidenced by studies from commercial security organizations such as McAfee [27] and the research community [13, 40, 42, 56].

Anti-virus (AV; also known as *anti-malware*) applications are a commonly used tool to detect malware. Such applications can use signatures and/or behavioral characteristics to identify and detect malicious applications (or malware). However, these AV engines may be targeted by cyber-attackers as well. For instance, Min and Varadharajan [38] presented an advanced AV parasitic malware, AV-Parmware, which is designed to attack protected components of AV software by exploiting their security weaknesses (similar to how systems can be compromised). They also identified weaknesses in AV software from four major vendors that can be exploited. In addition, in recent years there have been many attempts to design adversarial machine learning approaches to circumvent detection by AV applications [8]. This, therefore, necessitates the evaluation and benchmarking of different AV software, in terms of their effectiveness and efficiency (e.g., performance vs. cost) against malware.

There are a number of online scanning platforms that users can upload suspicious files for examination and obtain a report from several participating AV engines. However, such a platform is generally not configurable and sufficiently flexible. For example, it is challenging for a user to add their own AV engines, or a particular engine version. In addition, users often want to know the workload of different AV engines on their smartphones in order to select an appropriate AV application, which is not available from existing online platforms. In the current market, various AV engines are available¹, but there is no valid platform for users/researchers to evaluate different AV applications based on their own requirements. This is the gap we seek to address in this paper. The contributions in this work can be summarized as below:

- In this paper, we design an engineering platform for commercial AV application evaluation (hereafter referred to as *CAVAEva*). The platform allows users to configure the platform, for example by adding or removing AV software of interest, and to examine suspicious file(s) on a real smartphone environment. We also remark that we are not seeking to replace existing online tools, rather we are seeking to complement existing approaches.
- In addition, the code protection and different structures used by AV engines complicate efforts in implementing and automating the evaluation. Therefore, we present an approach to customize and deploy *CAVAEva* on a smartphone and a computer. In the evaluation, we evaluate seven commercial AV applications in terms of their classification performance and the incurred CPU usage on the smartphones.

¹ <https://www.digitalcitizen.life/how-choose-great-security-product-thats-right-you>.

- We further perform a user study with over 100 participants from academia and industry to collect their feedback regarding the utility of our platform. They can tune our platform according to their requirements and evaluate different AV applications. It is found that most participants considered our platform to be very useful in practice.

The remaining sections of this paper are organized as follows. Section 2 briefly reviews the relevant literature on smartphone threats and relevant studies on anti-malware comparison. Section 3 presents our proposed *CAVAEva* and its implementation. In Sect. 4, we describe an evaluation via a case study by comparing seven well-known commercial anti-virus software, and introduce a user study with over 100 participants. We also discuss some learnt experience and challenges. Finally, we conclude this paper in Sect. 5.

2 Background and Related Work

In this section, we first introduce the background on smartphone threats, and then review existing studies on anti-malware comparison.

2.1 Background on Smartphone Threats

There is a wide range of threats to smartphones, such as malware, side channel attacks and physical channel attacks.

Malware. Lin *et al.* [20] revealed that the Android Debug Bridge (ADB) capability could be utilized by applications (apps) with the INTERNET permission installed on the same device, and built *Screenmilk* to demonstrate how such a feature can be exploited. Specifically, Screenmilk is designed to monitor the user's screen and sniff user's key-in password in real-time. When studying Android's updating mechanism, Xing *et al.* [54] identified the Pileup flaws, through which a malicious app could strategically declare a set of privileges and attributes on a lower version operating system, and wait until it is able to escalate its privileges on the new system. Consequently, a malicious app can acquire a set of newly added system and signature permissions, as well as determining their settings. Andriesse and Bos [2] introduced a code hiding approach for trigger-based malware, which can conceal malicious code inside seemingly innocuous code fragments. Min and Varadharajan [38] introduced an AV parasitic malware, AV-Parmware, designed to attack protected components of anti-virus software by exploiting their weaknesses. We refer the interested reader to [13, 42, 56] for a summary of existing malware research.

Accelerometer Side Channel Attacks. Side channel is often utilized by malware to steal information on mobile devices. For example, Cai and Chen [7] presented a side channel attack targeting touchscreen smartphones with only soft keyboards. Specifically, they demonstrated that when users clicked on the soft

keyboard, especially when he/she holds the phone by hand rather than placing it on a fixed surface, the phone vibration on touchscreens are highly correlated to the keys being typed. They conducted a study and showed that they were able to infer correctly more than 70% of the keys typed on a number-only soft keyboard on a smartphone. Marquardt *et al.* [23] also demonstrated that an app with access to accelerometer readings on a smartphone could use side channel attacks to recover text entered on a nearby keyboard. They showed that by characterizing consecutive pairs of keypress events, up to 80% of typed content can be recovered. Do *et al.* [11] demonstrated how one can design an innocuous app to exfiltrate data from a smartphone (or any computing device) via sound frequency in the 20–22 kHz. Such an attack does not require the mobile app to request access to any permission, and uses only the built-in speaker or a connected ear phone.

Owusu *et al.* [41] demonstrated how accelerometer readings can be used to extract entire sequences of entered text on a smartphone. They showed how a background application can use the accelerometer to spy on keystroke information during sensitive activities, such as account login. In their study, the authors reportedly found 59 out of 99 passwords using only accelerometer measurements logged during text entry. Miluzzo *et al.* [39] presented *TapPrints*, a framework for inferring the location of taps on touchscreens using motion sensor data. Their approach reportedly accuracy rates of between 80% and 90% for English letters. A summary of other earlier works can be found in [4, 15, 18, 21, 46, 51, 57].

Physical Side Channel Attacks. These attacks are mainly based on physical objects, such as oil residues left on the touchscreen or the screen reflection from nearby objects. Aviv *et al.* [5] first explored the feasibility of smudge attacks on touchscreens with different lighting angles and light sources. They indicated that the pattern could be partially identifiable in 92% and fully in 68% of the tested lighting and camera setups. For the screen reflection, Raguram *et al.* [43] showed that automated reconstruction of text typed on a mobile device’s virtual keyboard is viable, by observing the reflections (e.g. reflection of the phone on the user’s lens).

Lau *et al.* [22] designed an early charging attack, *Mactans*, where the malicious charger (BeagleBoard) was used to inject malware onto smartphones when they were plugged into the malicious charger. However, their attacks require users to unlock the phone screen and install developer license in advance. Spolaor *et al.* [48] also described how an adversary could leverage a malicious charging station to exfiltrate smartphone data via a USB charging cable. Specifically, they designed *PowerSnitch*, an application designed to send data in the form of power bursts by manipulating the power consumption of the device’s CPU. One limitation of this attack is that users have to pre-install a small app on their phones. Meng *et al.* [32, 33] developed JFC attacks, which can record screen information during the whole charging period. Such an approach does not require any request for permission or phone unlock action, and cannot be detected by existing anti-AV applications. To launching this attack, an additional hardware of VGA/USB

interface is needed, which can be easily purchased online. Therefore, charging attacks are a real risk for smartphone users [34,35].

2.2 Related Work

To mitigate the above threats, it is necessary and important to deploy appropriate security mechanisms on smartphones. Anti-malware applications are one of the most commonly used methods in both academia and industry. To identify malware, a number of machine learning approaches, such as those reported in [6,10,19,37], have been proposed. Such approaches generally require some known instances to be used as training data. With the evolution of evading techniques, there is an increasing need to evaluate the performance and robustness of existing commercial AV engines. In current market, there are various commercial AV applications, while few studies focus on evaluating the performance among different AV engines.

Garuba et al. [14] conducted a comparative analysis of three methods in defeating malware on client computers such as anti-malware software, patch management, and host-based firewalls. Their work aims to guide organizations with limited security budgets and resources to find a proper defensive solution. Morales et al. [26] then focused on commercial anti-malware programs and introduces a measurement to evaluate the effectiveness. They found that several anti-malware programs may leave infected objects unresolved by producing numerous incorrectly treated or untreated true positives and false negatives. Talal et al. [50] studied the functionalities and services of several anti-malware applications in terms of the design mechanisms, features and strategies. There are also some studies investigated the security of detection engines against some particular attacks such as transformation attacks [44], Bytecode Obfuscation [12], anti-virus assisted attacks [53], etc.

There are also a number of online scanning platforms that can help compare the detection performance among commercial AV engines. However, these platforms are not configurable and flexible. For instance, users cannot add their interested commercial engines, and the practical running performance on their mobile devices is not available. This motivates our work in designing a more configurable and flexible platform to evaluate the performance among different commercial AV engines. In addition to AV software, there is a need to implement other solutions such as applying continuous authentication on smartphones based on behavioral biometrics [28,31] and examining CPU usage [17,36].

3 Proposed CAVAEva and Implementation

In this section, we first introduce how our proposed CAVAEva platform works, prior to explaining how it can be implemented.

Fig. 1. Proposed *CAVAEva* platform: a simplified workflow.

3.1 Platform Design

There are many machine learning-based malware detection algorithms proposed in the literature (e.g. [9]), and this number is increasing. However, due to the node protection and different structures, it is challenging to compare and benchmark the performance of different AV engines on a particular smartphone. As discussed earlier, existing online tools such as VirusTotal [52] do not generally provide the configuration flexibility for users or researchers. Our proposed platform, commercial anti-virus application evaluation (*CAVAEva*), is designed to enable users, researchers and practitioners to configure the environment (e.g. adding and removing AV engines of interest, and choosing how to output the decisions and results, such as using visualization), and facilitate evaluation on an actual smartphone.

Figure 1 presents a simplified typical *CAVAEva* workflow, comprising interactions among a smartphone, a computer and users (also described below). Specifically, users can use a computer to send relevant commands to the connected smartphone, for example for configuration (e.g. installing or removing AV engines). To ensure a realistic environment for AV engines, the connected smartphone upon receiving the commands from the computer can configure the smartphone’s environment, which can be used for malware detection (e.g., recording of detection performance, and CPU usage caused by the installed AV engines). Moreover, users can also install other detection approaches on the phone, such as deployment of machine learning-based detection with classification models and a knowledge database.

3.2 Platform Implementation

In comparison to other, existing detection approaches, it is more difficult to compare different AV applications. For example, AV vendors often employ some techniques to protect the code of their software and therefore, different AV engines

Fig. 2. An example of how APK files and malware are stored on the smartphone.

Fig. 3. Examples of where AV engines store their scanning results.

often have unique structure. Therefore, in this part, we introduce an engineering way of implementing *CAVAEva* in a real-world scenario.

Environmental Setup: To realize the platform implementation, we used a Linux machine (with Intel(R) Core(TM) i5-5300U CPU 2.3GHz, 16G RAM) and a Samsung Galaxy S6 smartphone, and adopted Java and Python as the programming language. This allows us to use various libraries provided by Python, such as matplotlib [24] and sqlite3 [49].

Preparation: After the environmental configuration, the first step is to create a folder with the same name on both the computer and the smartphone (e.g., to store AV apps and malware samples). In our platform, we choose to move all APK files and malware to */sdcard/Download/* on the smartphone. Figure 2 presents an example of how APK files and malware are stored on the smartphone. To automate the evaluation, the folder contains a subfolder of malware and various APK files for AV engines. Users can now add new AV APK files here, or remove those that are not of interest.

Then, it is very important to identify the location where one AV engine stores their scanning result. Figure 3 shows location examples on where these AV applications would store their scanning output, i.e., Lookout uses *seclog.txt* to store the result. Due to the different structures of AV engines, it is a challenge to identify such location automatically.

Detection Phase: Then, we develop a script to start installing AV engines on the smartphone, and check whether there is any installation problem. Figure 4


```
cmd = "adb shell pm install -r '/sdcard/Download/%s'" % am_name
result = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
output, err = result.communicate()
time.sleep(6)
installed = 0
count = 0
```

Fig. 4. An example script to start installing AV applications.

```
try:
    con = lite.connect('test.db')
    cur = con.cursor()
    cur.execute("INSERT INTO Result VALUES (?, ?, ?, ?)", (d, Malware_name, package_name, Malware))
    con.commit()
except lite.Error, e:
    if con:
        con.rollback()

    print "Error %s:" % e.args[0]
    sys.exit(1)
finally:
    if con:
        con.close()
    con = lite.connect('test.db')
```

Fig. 5. Saving the scanning result in a database (test.db): an example.

describes our script for installing AV engines, and before the installation, our script will clean and uninstall existing AV engines on the smartphones. We remark that the MonkeyRunner [25] app can be used to complete the process of installation for any AV engine that requires assistance from user input. This app will press the screen button to continue the process and sign up for a temporarily account that is obtained in advance. Then, our script starts installing malware on the phone to evaluate the performance of different AV engines, and then remove the installed malware before the installation of the next one. This loop will end until all malware have been tested. The results (e.g., execution information like CPU usage and detection performance) can then be forwarded to the computer.

Findings: In this phase, the main purpose is to output the scanning result in an expected format for each AV engine (using table and graphic visualization). In the example shown in Fig. 5, when all suspicious files have been successfully scanned, the result is saved in a database (test.db) using sqlite3. When the result is out, users can configure the environment to compare the performance among different AV applications, in terms of their workload and detection rate. As each AV engine may use a different way to store the scanning result, our platform provides a consistent way to help users perform a comparative benchmark.

Fig. 6. Classification accuracy and CPU usage.

4 Evaluation

In this section, we present a case study by evaluating seven anti-virus applications on *CAVAEva*, collect the feedback from over 100 participants in both academia and industry, and discuss some learnt experience and challenges.

4.1 Case Study

In the case study, we present our findings based on our evaluation of the following seven AV applications (all are free version in 2018): Avast, AVG, Lookout, Eset, Dr.Web, Kaspersky, and Avira. In our evaluation, we compiled our dataset comprising 11,232 Android apps by crawling app stores, such as Google Play (78%) and Anzhi (20%), as well as those obtained via direct download (e.g., DREBIN [3] - 2%).

Then, these apps were used to test our *CAVAEva* platform (comprising the Linux machine and Samsung Galaxy S6), and evaluate the performance of the seven AV applications. Figure 6 shows the classification accuracy and the CPU usage, where classification accuracy here indicates the capability of identifying both benign and malicious applications. It was observed that all seven AV engines achieved over 97% classification accuracy, with Lookout achieving 99.8% classification accuracy. In terms of CPU usage, all seven AV engines required at least 32% additional usage due to the scanning operations and interactions, among which Kaspersky needed more CPU usage while Avira required less usage than others.

Table 1. Participants' background in the user study.

Age range	Male	Female	Occupation	Male	Female
18–35	31	32	Students from computer science	17	14
35–45	22	15	Researchers	21	24
41–50	4	2	Industry engineers	19	11

Table 2. Major questions and relevant scores collected from the user study.

Questions (CS students)	Average score
1. The platform is more flexible and configurable	8.8
2. I think the platform is useful in practice	9.1
3. I may consider using this platform in my future work	8.5
Questions (Researchers)	Average score
1. The platform is more flexible and configurable	8.3
2. I think the platform is useful in practice	8.5
3. I may consider using this platform in my future work	8.2
Questions (Industry engineers)	Average score
1. The platform is more flexible and configurable	8.9
2. I think the platform is useful in practice	8.7
3. I may consider using this platform in my future work	9.1

4.2 User Study

To collect the feedback on the utility of our platform, we perform a user study with over 100 participants by distributing our platform to both researchers and industry engineers. The participants' background is summarized in Table 1. All participants have an interest on our work and were recruited via online platforms and social recommendations. In particular, the students are from the computer science major including both master and PhD students. Researchers including postdoc fellows and faculty members. The industry engineers are from several IT companies.

Before the start, we introduced our purpose to all participants. They could use the platform for a week and then completed a feedback form, which contains a set of questions regarding the platform performance and usage. Each question employs ten-point Likert scales: namely, 1-score indicates strong disagreement and 10-score indicates strong agreement. Table 2 shows the questions and the collected feedback.

- *The first question.* For this question, we asked the participants to consider several existing online detection platforms, as compared with our platform. It is found that most participants supported our platform to be more flexible and configurable. Specifically, industry engineers provided the highest score, indicating the promising performance.

```
cmd = "./monkeyrunner execute-lookout.py"
subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
time.sleep(60)

from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice, MonkeyImage
device = MonkeyRunner.waitForConnection()
device.touch(562, 1796, 'DOWN_AND_UP')
MonkeyRunner.sleep(1)
device.touch(562, 1796, 'DOWN_AND_UP')
MonkeyRunner.sleep(1)
device.touch(562, 1796, 'DOWN_AND_UP')
MonkeyRunner.sleep(1)
device.touch(150, 700, 'DOWN_AND_UP')
```

Fig. 7. An example on how to use MonkeyRunner to activate Lookout after the installation.

- *The second question.* This question aims to investigate the participants' attitude towards the usage of our platform. It is visible that all scores from different groups are above 8.5, indicating the importance of our platform. In our informal interview, many participants consider that our platform can complement the existing online detection tools and help enhance the commercial anti-malware applications.
- *The third question.* According to this question, we found that most participants are willing to use our platform in their future work. In particular, researchers provided the lowest score of 8.2 while industry engineers provided the highest one of 9.1. In our informal interview, it is found that researchers focused more on how to extend our platform according to their demands, and then design a new/updated one. By contrast, industry engineers focused more on how to apply this platform to their work directly. This is the main reason for explaining their different scores.

Overall, the collected feedback and scores are positive, demonstrating that our platform has a potential to be applied in practice, with many demanding features like flexibility and extensibility.

4.3 Discussion

In this work, our main purpose is to design a configurable and flexible platform for automatic commercial anti-virus application evaluation. Our platform is still developed at an early stage, and can be enhanced in many aspects. In this part, we discuss some learnt experience and challenges. A summary of our findings is presented as below:

- *File size.* In the evaluation, we found that free online services may have a size limit for the uploaded apps, ranging from 32 Megabyte to 64 Megabyte. Our observation echoed that of [16]. In comparison, there is no such limit for our platform, as *CAVAEva* will install and evaluate one app on an actual smartphone.

Number	Malware_type	Package_name	Detect
1	jSMShider	app.BatteryMonitor	Ture
2	jSMShider	com.AudioConsole	Ture
3	jSMShider	hider.ApplInstall.nvanmoshiriji_V31_mumayi_aff08	Ture
4	jSMShider	com.trendmicro.tmmpersonal.apac	False
5	anserverBot	com.keji.danti604	Ture
6	anserverBot	com.keji.danti607	Ture
7	HippoSMS	com.ku6.android.videobrowser	Ture
8	HippoSMS	com.ku6.android.videobrowser	Ture
9	HippoSMS	com.Video.XiaoQiang	Ture
10	HippoSMS	com.mass.sms	Ture

Fig. 8. The figure shows how to output the scanning results in a table.

- *Screen activation.* To automate the evaluation process, some AV engines require users to click a ‘Next’ button to activate the next step. In this work, we employed MonkeyRunner to mitigate this limitation – see Fig. 7. Other approaches can also be used to automate the evaluation process.
- *AV engine selection.* In our platform, users can freely deploy their AV software of choice, and this flexibility is not commonly found in other tools/platforms.
- *Output format.* In this platform, the scanning results can be displayed in a variety of formats, such as tables and charts – see Fig. 8. Users can also customize the display formats, such as using visualization.

As our platform is only a proof-of-concept, the platform can be enhanced in a number of aspects. Examples include the following:

- *Output speed.* As compared to existing online scanning tools, our platform may require more time to complete the examination and producing the result. This is because CAVAEva has to install each AV engine and test all files or apps on an actual smartphone. Hence, performance optimization is one of our future agenda.
- *Malware label and lack of a standard naming convention.* Hurier *et al.* [16] reported that current online scanning services lack consensus, in the sense that binary decisions from different AV engines may produce conflicting results on the same samples. In addition, they also noted that malware labels are very challenging to compare due to the lack of a naming convention for malware samples. Hence, both limitations should also be studied in the future.
- *Smartphone support.* Our current platform supports most Samsung phones, and users can select their smartphone model. For other smartphones (i.e., different makes, such as iPhones), there is a need to configure the platform. In the future, we plan to extend the platform to be compatible with other smartphones.
- *Online scanning tool comparison support.* In this work, we only focused on designing a flexible and automated platform to facilitate evaluation of commercial AV applications. In the future, we intend to extend the platform to also include existing online scanning tools.

- *Full automation.* One main purpose of our platform is to automate the evaluation process of different AV engines, whereas each AV engine may have different design structures, making the task difficult. To achieve our purpose, we implemented various tools and scripts on *CAVAEva*. In our future work, we plan to test the platform performance in a systematic manner.

Overall, how to develop a unified and flexible platform for commercial anti-virus application evaluation is a challenge, i.e., there is a need to solve many technical issues. Our platform is one step for addressing such challenge, with the purpose of complementing the current literature.

5 Conclusion and Future Work

Detecting malware on existing computing devices, such as smartphones, is crucial and remains a challenging topic. While there are many online scanning tools and AV applications, there is no user friendly manner to evaluate and benchmark such tools and applications. In addition, many of these online scanning tools have limitations such as file-size restriction. Thus, there is a need to design a configurable and unified platform for evaluating different AV engines.

In this work, we designed *CAVAEva*, an engineering platform for commercial AV software evaluation. The platform allows users, researchers and practitioners to configure the platform based on their own needs, and the evaluation process is automated. To investigate the utility of our platform, we performed a case study by comparing seven well-known commercial anti-virus applications regarding detection performance and CPU usage, and held a user study with over 100 participants from both academia and industry to collect their feedback. Our results demonstrate that our platform can provide many demanding features, i.e., showing the caused CPU usage on a real smartphone, and is useful in practice. Our work attempts to complement existing research on malware detection and stimulate more research on such challenge.

Acknowledgments. Weizhi Meng was partially supported by H2020-SU-ICT-03-2018: CyberSec4Europe with No. 830929, and National Natural Science Foundation of China (No. 61802077). Chunhua Su is supported by JSPS Kiban(B) 18H03240 and JSPS Kiban(C) 18K11298.

References

1. Anand, S.A., Saxena, N.: Speechless: analyzing the threat to speech privacy from smartphone motion sensors. In: Proceedings of the 2018 IEEE Symposium on Security and Privacy, pp. 1000–1017 (2018)
2. Andriessse, D., Bos, H.: Instruction-level steganography for covert trigger-based malware. In: Dietrich, S. (ed.) DIMVA 2014. LNCS, vol. 8550, pp. 41–50. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08509-8_3
3. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K.: DREBIN: effective and explainable detection of android malware in your pocket. In: Proceedings of NDSS (2014)

4. Asonov, D., Agrawal, R.: Keyboard acoustic emanations. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 3–11 (2004)
5. Aviv, A.J., Gibson, K., Mossop, E., Blaze, M., Smith, J.M.: Smudge attacks on smartphone touch screens. In: Proceedings of the 4th USENIX Conference on Offensive Technologies (WOOT), pp. 1–7 (2010)
6. Backes, M., Nauman, M.: LUNA: quantifying and leveraging uncertainty in Android malware analysis through Bayesian machine learning. In: EuroS&P 2017, pp. 204–217 (2017)
7. Cai, L., Chen, H.: TouchLogger: inferring keystrokes on touch screen from smart-phone motion. In: Proceedings of the 6th USENIX Conference on Hot Topics in Security (HotSec), pp. 1–6 (2011)
8. Chen, S., et al.: Automated poisoning attacks and defenses in malware detection systems: an adversarial machine learning approach. *Comput. Secur.* **73**, 326–344 (2018)
9. Chen, X., et al.: Android HIV: a study of repackaging malware for evading machine-learning detection. *IEEE Trans. Inf. Forensics Secur.* **15**, 987–1001 (2019)
10. Chen, Z., et al.: Machine learning based mobile malware detection using highly imbalanced network traffic. *Inf. Sci.* **433–434**, 346–364 (2018)
11. Do, Q., Martini, B., Choo, K.-K.R.: Exfiltrating data from Android devices. *Comput. Secur.* **48**, 74–91 (2015)
12. Faruki, P., Bharmal, A., Laxmi, V., Gaur, M.S., Conti, M., Rajarajan, M.: Evaluation of Android anti-malware techniques against Dalvik bytecode obfuscation. In: Proceedings of TrustCom, pp. 414–421 (2014)
13. Faruki, P., et al.: Android security: a survey of issues, malware penetration, and defenses. *IEEE Commun. Surv. Tutor.* **17**(2), 998–1022 (2015)
14. Garuba, M., Liu, C., Washington, A.N.: A comparative analysis of anti-malware software, patch management, and host-based firewalls in preventing malware infections on client computers. In: Proceedings of ITNG, pp. 628–632 (2008)
15. Han, J., Owusu, E., Nguyen, L., Perrig, A., Zhang, J.: ACComplice: location inference using accelerometers on smartphones. In: Proceedings of the 4th International Conference on Communication Systems and Networks (COMSNETS), New York, NY, USA, pp. 1–9 (2012)
16. Hurier, M., Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y.: On the lack of consensus in anti-virus decisions: metrics and insights on building ground truths of Android malware. In: Caballero, J., Zurutuza, U., Rodríguez, R.J. (eds.) DIMVA 2016. LNCS, vol. 9721, pp. 142–162. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40667-1_8
17. Jiang, L., Meng, W., Wang, Y., Su, C., Li, J.: Exploring energy consumption of juice filming charging attack on smartphones: a pilot study. In: Yan, Z., Molva, R., Mazurczyk, W., Kantola, R. (eds.) NSS 2017. LNCS, vol. 10394, pp. 199–213. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64701-2_15
18. Kune, D.F., Kim, Y.: Timing attacks on PIN input devices. In: Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS), pp. 678–680. ACM, New York (2010)
19. Li, J., Sun, L., Yan, Q., Li, Z., Srisa-an, W., Ye, H.: Significant permission identification for machine-learning-based Android malware detection. *IEEE Trans. Ind. Inform.* **14**(7), 3216–3225 (2018)
20. Lin, C.-C., Li, H., Zhou, X., Wang, X.: Screenmilk: how to milk your Android screen for secrets. In: Proceedings of Annual Network and Distributed System Security Symposium (NDSS), pp. 1–10 (2014)

21. Liu, J., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uWave: accelerometer-based personalized gesture recognition and its applications. *Pervasive Mob. Comput.* **5**(6), 657–675 (2009)
22. Lau, B., Jang, Y., Song, C.: Mactans: injecting malware into iOS devices via malicious chargers. *Blackhat USA* (2013)
23. Marquardt, P., Verma, A., Carter, H., Traynor, P.: (sp)iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In: *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pp. 551–562. ACM, New York (2011)
24. Matplotlib: Python plotting. <https://matplotlib.org/>
25. MonkeyRunner: A monkeyrunner class that contains static utility methods. <https://developer.android.com/studio/test/monkeyrunner/MonkeyRunner>
26. Morales, J.A., Sandhu, R.S., Xu, S.: Evaluating detection and treatment effectiveness of commercial anti-malware programs. In: *Proceedings of MALWARE*, pp. 31–38 (2010)
27. McAfee: McAfee Mobile Threat Report Q1, 2018. <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2018.pdf>
28. Meng, Y., Wong, D.S., Schlegel, R., Kwok, L.: Touch gestures based biometric authentication scheme for touchscreen mobile phones. In: Kutylowski, M., Yung, M. (eds.) *Inscript 2012. LNCS*, vol. 7763, pp. 331–350. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38519-3_21
29. Meng, Y., Li, W., Kwok, L.-F.: Enhancing click-draw based graphical passwords using multi-touch on mobile phones. In: Janczewski, L.J., Wolfe, H.B., Shenoi, S. (eds.) *SEC 2013. IAICT*, vol. 405, pp. 55–68. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39218-4_5
30. Meng, W., Li, W., Kwok, L.F.: EFM: enhancing the performance of signature-based network intrusion detection systems using enhanced filter mechanism. *Comput. Secur.* **43**, 189–204 (2014)
31. Meng, W., Wong, D.S., Furnell, S., Zhou, J.: Surveying the development of biometric user authentication on mobile phones. *IEEE Commun. Surv. Tutor.* **17**, 1268–1293 (2015)
32. Meng, W., Lee, W.H., Murali, S.R., Krishnan, S.P.T.: Charging me and I know your secrets! Towards juice filming attacks on smartphones. In: *Proceedings of the Cyber-Physical System Security Workshop (CPSS)*, in conjunction with AsiaCCS 2015. ACM (2015)
33. Meng, W., Lee, W.H., Murali, S.R., Krishnan, S.P.T.: JuiceCaster: towards automatic juice filming attacks on smartphones. *J. Netw. Comput. Appl.* **68**, 201–212 (2016)
34. Meng, W., Fei, F., Li, W., Au, M.H.: Harvesting smartphone privacy through enhanced juice filming charging attacks. In: Nguyen, P., Zhou, J. (eds.) *ISC 2017. Lecture Notes in Computer Science*, vol. 10599, pp. 291–308. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69659-1_16
35. Meng, W., Lee, W.H., Liu, Z., Su, C., Li, Y.: Evaluating the impact of juice filming charging attack in practical environments. In: Kim, H., Kim, D.-C. (eds.) *ICISC 2017. LNCS*, vol. 10779, pp. 327–338. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78556-1_18
36. Meng, W., Jiang, L., Wang, Y., Li, J., Zhang, J., Xiang, Y.: JFCGuard: detecting juice filming charging attack via processor usage analysis on smartphones. *Comput. Secur.* **76**, 252–264 (2018)
37. Milosevic, N., Dehghantanha, A., Choo, K.K.R.: Machine learning aided Android malware classification. *Comput. Electr. Eng.* **61**, 266–274 (2017)

38. Min, B., Varadharajan, V.: Design, implementation and evaluation of a novel anti-virus parasitic malware. In: Proceedings of SAC 2015, pp. 2127–2133 (2015)
39. Miluzzo, E., Varshavsky, A., Balakrishnan, S., Choudhury, R.R.: TapPrints: your finger taps have fingerprints. In: Proceedings of MobiSys, New York, NY, USA, pp. 323–336 (2012)
40. Nguyen, G., Nguyen, B.M., Tran, D., Hluchy, L.: A heuristics approach to mine behavioural data logs in mobile malware detection system. *Data Knowl. Eng.* **115**, 129–151 (2018)
41. Owusu, E., Han, J., Das, S., Perrig, A., Zhang, J.: ACCessory: password inference using accelerometers on smartphones. In: Proceedings of the 12th Workshop on Mobile Computing Systems & Applications (HotMobile), pp. 1–6. ACM, New York (2012)
42. Peng, S., Yu, S., Yang, A.: Smartphone malware and its propagation modeling: a survey. *IEEE Commun. Surv. Tutor.* **16**(2), 925–941 (2014)
43. Raguram, R., White, A.M., Goswami, D., Monrose, F., Frahm, J.-M.: iSpy: automatic reconstruction of typed input from compromising reflections. In: Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), pp. 527–536. ACM, New York (2011)
44. Rastogi, V., Chen, Y., Jiang, X.: DroidChameleon: evaluating Android anti-malware against transformation attacks. In: Proceedings of AsiaCCS, pp. 329–334 (2013)
45. Rudd, E.M., Rozsa, A., Ganther, M., Boulton, T.E.: A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions. *IEEE Commun. Surv. Tutor.* **19**(2), 1145–1172 (2017)
46. Schlegel, R., Zhang, K., Zhou, X., Intwala, M., Kapadia, A., Wang, X.: Soundcomber: a stealthy and context-aware sound trojan for smartphones. In: Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, pp. 17–33 (2011)
47. Sen, S., Aydogan, E., Aysan, A.I.: Coevolution of mobile malware and anti-malware. *IEEE Trans. Inf. Forensics Secur.* **13**(10), 2563–2574 (2018)
48. Spolaor, R., Abudahi, L., Moonsamy, V., Conti, M., Poovendran, R.: No free charge theorem: a covert channel via USB charging cable on mobile devices. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 2017. LNCS, vol. 10355, pp. 83–102. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61204-1_5
49. SQLite. <https://www.sqlite.org/>
50. Talal, M., et al.: Comprehensive review and analysis of anti-malware apps for smartphones. *Telecommun. Syst.* **72**(2), 285–337 (2019)
51. Vuagnoux, M., Pasini, S.: Compromising electromagnetic emanations of wired and wireless keyboards. In: Proceedings of the 18th Conference on USENIX Security Symposium, pp. 1–16 (2009)
52. VirusTotal: Analyze suspicious files and URLs to detect types of malware, automatically share them with the security community. <https://www.virustotal.com/#/home/upload>
53. Wressnegger, C., Freeman, K., Yamaguchi, F., Rieck, K.: Automatically inferring malware signatures for anti-virus assisted attacks. In: Proceedings of AsiaCCS, pp. 587–598 (2017)
54. Xing, L., Pan, X., Wang, R., Yuan, K., Wang, X.: Upgrading your Android, elevating my malware: privilege escalation through mobile OS updating. In: Proceedings of the 2014 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, pp. 393–408 (2014)

55. Xu, N., Zhang, F., Luo, Y., Jia, W., Xuan, D., Teng, J.: Stealthy video capturer: a new video-based spyware in 3G smartphones. In: Proceedings of the 2nd ACM Conference on Wireless Network Security (WiSec), pp. 69–78 (2009)
56. Ye, Y., Li, T., Adjero, D.A., Iyengar, S.S.: A survey on malware detection using data mining techniques. *ACM Comput. Surv.* **50**(3), 41:1–41:40 (2017)
57. Zhuang, L., Zhou, F., Tygar, J.D.: Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur.* **13**(1), 1–26 (2009)