Towards Behavior-based Analysis of Android Obfuscated Malware

Zakaria Sawadogo , Muhammad Taimoor Khan , George Loukas, Jean-Marie Dembele , Georgia Sakellari , and Gervais Mendy

Gaston Berger University, Centre for Sustainable Cyber Security, University of Greenwich, Cheikh Anta Diop University

Abstract

In this article, we present the preliminary results of an ongoing project aimed at rigorously detecting obfuscated Android malware. The detection of Android malware has become increasingly complex due to the use of various obfuscation techniques by malicious developers. Existing approaches have primarily focused on specific obfuscation methods, but the evolving nature of these techniques makes it difficult to account for all possible variations. To address this challenge, we have developed an innovative behavioral methodology for analyzing obfuscated malware. Our approach combines model-based techniques with AI-based methods, making it the first initiative to integrate these two approaches for obfuscated malware detection. Given that deobfuscation is an extremely complex computational problem (i.e., NP-hard), our methodology circumvents obfuscation by indirectly observing malware behavior through the real-time behavior of target services controlled and operated by Android applications. This work has already been published on SpringerLink.

Related Work

This section reviews related work on behavior-based malware analysis, both online and offline. Recent advancements in AI/ML-based malware detection leverage machine learning and deep learning algorithms to analyze Android applications, focusing on features like permissions, code structure, API calls, opcodes, and intents. These techniques have improved model efficiency, particularly in detecting obfuscated malware by integrating code analysis, memory dumping, and machine learning [2,5,6]. However, challenges remain, including biases that affect model reliability and robustness, as well as the dynamic nature of malware and evolving attack techniques that hinder the development of highly accurate models. Additionally, obfuscation and evasion techniques employed by malware developers complicate detection.

Other approaches have utilized heuristics and formal modeling techniques to model malicious application behavior. Heuristics-based methods depend on expert knowledge and predefined rules to identify suspicious behaviors, while formal modeling involves creating mathematical or logical models to analyze application behavior against precise specifications. Formal methods allow for detailed analysis of malicious code, identifying vulnerabilities, and understanding malware's impact on systems [2,3,7]. Cimitile et al. [4], proposed a formal method for detecting obfuscation in mobile applications, but their model was based on a limited dataset and required model checking, which involves translating the system into a form that can be rigorously analyzed.

Modelling of Behavior

(B)luetooth service works (e.g., plays audio or video) either offline or online.
(D)ownload service downloads data by automatically synchronizing connectivity.
(S)ynchronisation service is required by the Download service.
(T)ransfer service is enabled when multimedia content download is required by the Bluetooth service.

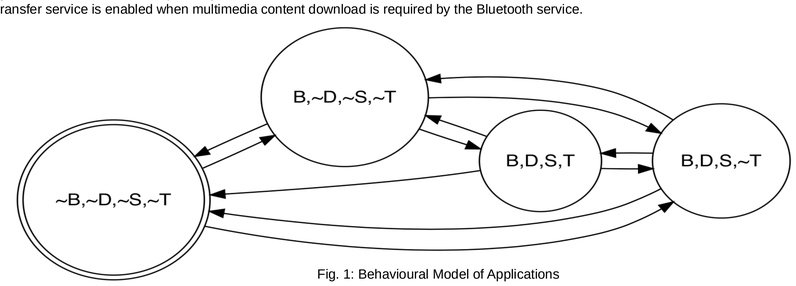


Fig. 1: Behavioural Model of Applications

Current State \ Next State	s_0	s_1	s_2	s_3
s_0	✗	✓	✓	✗
s_1	✓	✗	✓	✓
s_2	✓	✓	✗	✓
s_3	✓	✓	✓	✗

Fig. 1: Possible Behavioral Transitions

- $s_0 = \langle \neg B, \neg D, \neg S, \neg T \rangle$ - in the initial and final state all the services are inactive.
- $s_1 = \langle B, \neg D, \neg S, \neg T \rangle$ - when only Bluetooth is active and working offline
- $s_2 = \langle B, D, S, \neg T \rangle$ - when Bluetooth is active and working in online mode.
- $s_3 = \langle B, D, S, T \rangle$ - when Bluetooth is active and working in online mode by transferring multimedia contents.

Workflow of Malware Detection

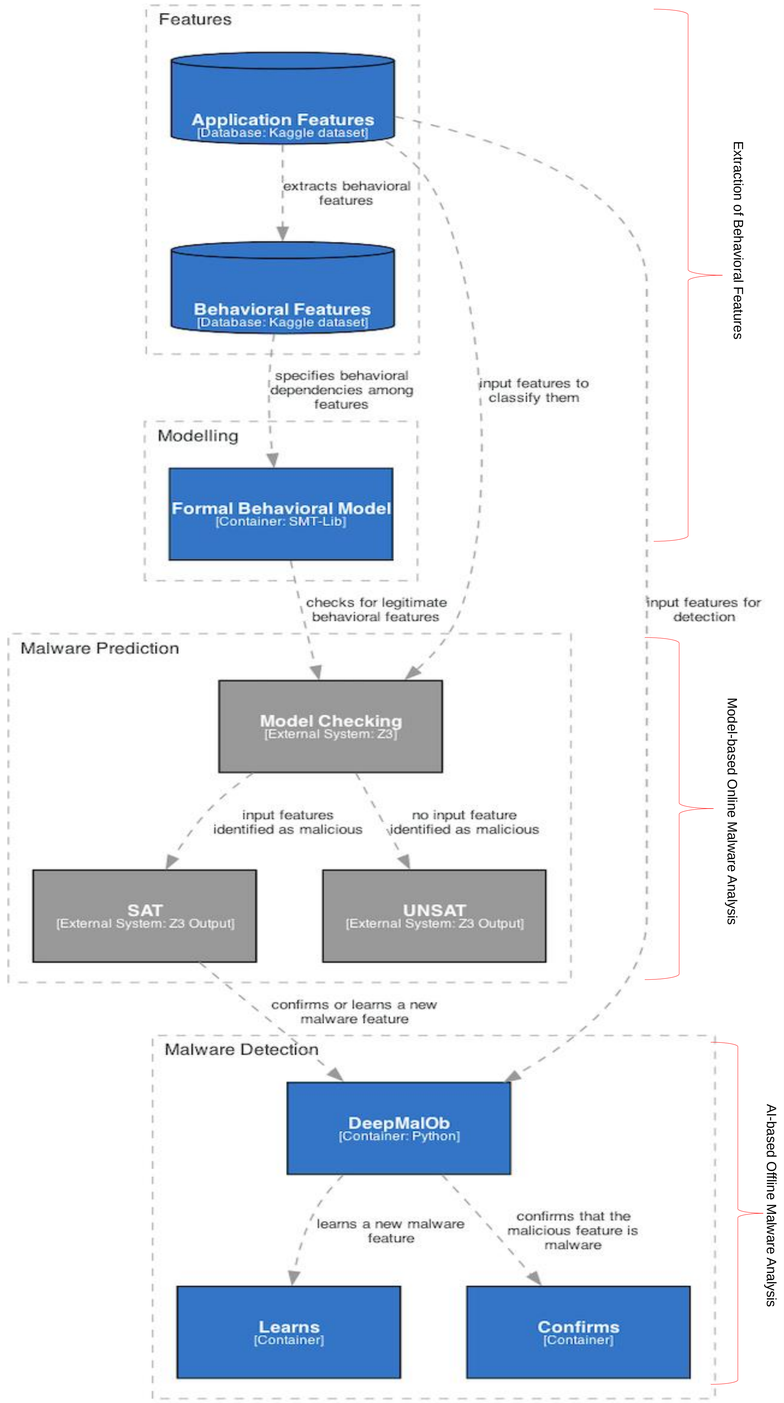


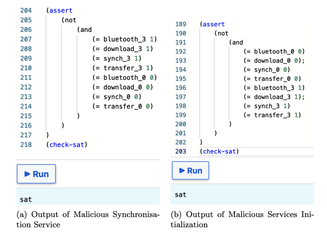
Fig. 3: Workflow of Malware Detection

Model-based Online Malware Analysis

Algorithm 1 Algorithm for Online Malware Behaviour Detection

Require: $O : (\langle b_i, d_i, s_i, t_i \rangle)_{i=1}^{n}$ s.t. $|O| \geq 2$
▷ A sequence of Observed behavioural states of services, i.e., b: Bluetooth, d: download, s: synchronisation, and t: transfer.
Require: $E : \mathbb{S}$ ▷ A set of Expected behavioural states of different services
Ensure: $P : \varnothing \vee (\langle b_i, d_i, s_i, t_i \rangle) \sqsubseteq (\langle b_j, d_j, s_j, t_j \rangle)_{j=1}^{n} \vee (\langle b_{j-1}, d_{j-1}, s_{j-1}, t_{j-1} \rangle, \langle b_j, d_j, s_j, t_j \rangle) \sqsubseteq (\langle b_j, d_j, s_j, t_j \rangle)_{j=1}^{n}$
▷ Either empty, when no malware Pattern,
▷ Or a single state, when either previous or next state is not an expected behavioural state
▷ Or a sequence of two (previous and next) states, when the two states characterise a malware behavioural transition
$P \leftarrow \varnothing$ ▷ There is no malicious pattern, initially
$i \leftarrow 0$
$l \leftarrow |O|$
while $i < l - 1$ **do**
  if $O[i] \in E$ **then** ▷ If the current state is an expected state
    if $O[i+1] \in E$ **then** ▷ If the next state is an expected state
      if $\neq \text{next}(O[i], O[i+1])$ **then**
        return $P \sqcup O[i] \sqcup O[i+1]$
▷ Return the sequence of two (previous and next) states when the transition from the previous state to the next state is unexpected
      end if
    else
      return $P \sqcup O[i+1]$ ▷ Return the next state, if the next state is not an expected state
    end if
  else
    return $P \sqcup O[i]$ ▷ Return the current state, if the current state is not an expected state
  end if
  $i \leftarrow i + 1$
end while
return $\varnothing$ ▷ No malicious pattern found

Fig. 4 : Algorithm for Online Malware Behaviour Detection



(a) Output of Malicious Synchronisation Service

(b) Output of Malicious Services Initialization

Fig. 5: Verification of Online Malware Analysis

To implement the model, we have used model checking of the transition system using Z3 [1], which is an Satisfiability Modulo Theory (SMT) Solver checking satisfiability of formulas in a decidable first-order theory (Fig.5). Z3 accepts SMT-Lib format model, which implies that our previously developed model is amenable for satisfiability check by Z3. We ask the question to the solver that given the transition model, the given sequence of observed behavioral states are malicious, i.e., they do not satisfy the model. Then the solver answers:
– either "SAT" which implies that the two given observed states are indeed malicious
– or "UNSAT" which implies that the two given observed states are not malicious.

AI-based Offline Malware Analysis

For offline malware analysis, we use DeepMalOb, a malware detection technique that detects obfuscated malware without relying on specific obfuscation techniques. This approach has been described in a published article [6] .

Conclusion

Currently all the different modules of the workflow (see Fig.3) are developed independently. As a next step, we want to automate the entire workflow through

- ***intra-module integration*** that aims at automating the internal processing of each module. Specifically, we want to automatically generate the scalable Z3 model based on large features of the dataset in a sound way considering more complex behavioral properties, e.g., permissions that require access not only to other applications and services but to underlying hardware, e.g., memory and other peripheral interfaces.
- ***inter-module integration*** that aims at automating interfacing of different modules by making input/output of the modules interoperable. Specifically, we want to extend AI-based offline analysis to learn new malware patterns as identified by the model based online analysis. Beside the integration, we aim to assess the performance of the model on diverse datasets and real-world scenarios. This involves comparing the malware detection results obtained by our model with those achieved by other existing approaches. Through this evaluation, we aim to identify any challenges or limitations encountered during the analysis process

Acknowledgments

The Partnership for Skills in Applied Sciences, Engineering, and Technology -Regional Scholarship and Innovation Fund (PASET RSIF) provided support for this work

References

1. Z3 Theorem Prover, https://github.com/Z3Prover
2. Ahvanooey, M.T., Li, P.Q., Rabbani, M., Rajput, A.R.: A Survey on Smartphones Security : Software Vulnerabilities , Malware , and Attacks
3. Barrett, C., Stump, A., Tinelli, C., et al.: The smt-lib standard: Version 2.0. In:Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, UK). vol. 13, p. 14 (2010)
4. Cimitile, A.,Martinelli, F., Mercaldo, F., Nardone, V., Santone, A.: Formal methods meet mobile code obfuscation identification of code reordering technique. In: 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). pp. 263–268(2017).https://doi.org/10.1109/WETICE.2017.23
5. Khan, M.T., Serpanos, D., Shrobe, H., Yousuf, M.M.: Rigorous machine learning for secure and autonomous cyber physical systems. In: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). vol. 1, pp. 1815–1819 (2020). https://doi.org/10.1109/ETFA46521.2020.9212074
6. Sawadogo, Z., Dembele, J.M., Tahar, A., Mendy, G., Ouya, S.: DeepMalOb: Deep Detection of Obfuscated Android Malware. In: Ngatched Nkouatchah, T.M., Woungang, I., Tapamo, J.R., Viriri, S. (eds.) Pan-African Artificial Intelligence and Smart Systems. pp. 307–318. Springer Nature Switzerland, Cham (2023). https://doi.org/https://doi.org/10.1007/978-3-031-25271-6 19
7. Taimoor Khan, M.: Towards practical and formal security risk analysis of iot (internet of things) applications. IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2022-September (2022). https://doi.org/10.1109/ETFA52439.2022.9921511