Themis: Ambiguity-Aware Network Intrusion Detection based on Symbolic Model Comparison

Zhongjie Wang*, Shitong Zhu§, Keyu Man§, Pengxiong Zhu§, Yu Hao§, Zhiyun Qian§
Srikanth V. Krishnamurthy§, Tom La Porta†, and Michael J. De Lucia‡
Baidu Security*, University of California, Riverside§, Pennsylvania State University†, U.S. Army Research Laboratory‡

ACM Reference Format:
Zhongjie Wang*, Shitong Zhu§, Keyu Man§, Pengxiong Zhu§, Yu Hao§, Zhiyun Qian§ and Srikanth V. Krishnamurthy§, Tom La Porta†, and Michael J. De Lucia‡. 2021. Themis: Ambiguity-Aware Network Intrusion Detection based on Symbolic Model Comparison. In *Proceedings of the 8th ACM Workshop on Moving Target Defense (MTD '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3474370.3485669

many different implementations at the same time. With evaluation, we demonstrate that Themis can successfully capture all existing NIDS evasion strategies while introducing only negligible overhead (around 1%) to the NIDS.

1 INTRODUCTION

Network Intrusion Detecion Systems (NIDS) are inherently vulnerable to evasion attacks that exploit implementation-level discrepancies [5]. In recent years, the evasion technique has evolved from manual strategy generation to automatic strategy generation [1, 8]. As defenders, we seek to regain the advantage and proactively prevent such potential attacks from happening, instead of reactively patching NIDSs. We propose a novel framework, Themis, to defend against evasion attacks. The key idea is that one can learn the discrepancies among different implementations ahead of time, and fork the connection states on the NIDS when ambiguous packets are encountered. The NIDS will then analyze the plurality of forked states in parallel, ensuring that one of the connection states will be synchronized with that of the endhost.

Themis leverages exhaustive symbolic execution to extract high-fidelity models from different versions of Linux TCP implementations. Compared to manually reconstructed models, our models are faithful representations of the actual software running on the endhosts, and are guaranteed to have exactly the same behaviors. We employ state merging [3] to solve the scalability issue in symbolic execution. The extracted models are in the form of high-level SMT formulas; thus, it is easy to use SMT solvers to automatically compare two models to find discrepancies. Upon finding a comprehensive set of discrepancies, we then build an ambiguity-aware NIDS based on nondeterministic finite automata (NFA) that can effectively and simultaneously support multiple different implementations. This approach has several benefits. First, since we go straight to the endhost implementations, we can abandon the existing over-simplified and over-approximated NIDS implementations that have potentially many more discrepancies. Second, with the distilled discrepancies, we no longer need to blindly run

MTD '21, November 15, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8658-6/21/11.
https://doi.org/10.1145/3474370.3485669

2 OFFLINE PHASE: DISCOVERING TCP IMPLEMENTATION DISCREPANCIES

The offline phase of Themis that finds discrepancies between any two TCP implementations has three key components, as shown in **??**. The first is *Symbolic Model Extraction*, which runs symbolic execution exhaustively on different versions of TCP implementations and extracts high-fidelity models to accurately reflect detailed behaviors of each implementation. We model the TCP behaviors by defining critical states, e.g. high-level TCP states such as SYN_RECV or ESTABLISHED, and receive buffer events. If the same sequence of input packets drives two TCP implementations into different critical states, then they will trigger different behaviors. Via exhaustive symbolic execution, we extract a mapping from path constraints to critical states. We solve the path explosion problem in symbolic execution with the idea of state merging [3], and apply heuristics summarized from our domain expertise in TCP.

The second component is called *Model Comparison*, which compares two symbolic models and automatically generates concrete examples that will trigger the discrepancies between them. We group the exectuion traces from symbolic execution by their critcal states, and then combine their path constraints with disjunction. The combined path constraints reflect all possible inputs that will drive the TCP implementation into certain critical state. Then we compare the combined path constraints from two different TCP implementations, for each critical state, with a constraint solver. The constraint solver will either prove that the two path constraints are equivalent or generate a concrete counterexample that is accepted by one of the path constraints but not the other.

The last component is *Discrepancy Analysis*, which empirically analyzes the execution traces corresponding to the concrete examples and determines the root cause of a discrepancy in the behaviors between the two implementations. The process is iterative in that we feed the discrepancies summarized from *Discrepancy Analyis* back to the *Model Comparison* to exclude them from the models in the next round of concrete example generation, until there are no discrepancies between the two models. These discrepancies will also be integrated into the NIDS to enable online operations of Themis as discussed in §3.

3 ONLINE PHASE: AMBIGUITY-AWARE NIDS

We propose a novel, NFA-based model for the NIDS. In NFA, upon receiving an input, the state could transition into multiple different new states non-deterministically. In practice, an NFA "clones" its state when there are multiple possible next states. Thus, it enables the NIDS to handle packets with ambiguities. We integrate the discrepancies discovered in §2 into the existing DFA of the NIDS and turn it into an NFA. In this way, we are essentially merging multiple DFAs into an NFA while reusing the common parts in the DFAs to the maximum extent possible. Note that for each connection and ambiguity, we only fork once and remember the behavior associated with the copy of the connection state. Assuming there are k ambiguities, then the maximum number of forked connection states is 2^k. As a further optimization, we take version coherence of the behaviors into account and reduce the growth rate to a linear rate. The idea is that we maintain a profile for a set of versions that have the exact same behaviors. Therefore the maximum number of states forked will be dictated by the number of profiles.

4 EVALUATION

Themis's offline phase is built upon S2E [2] and Z3 [9], and online phase built upon Zeek (formerly Bro) [10]. In our exhaustive symbolic execution, we send 3 symbolic TCP packets with TCP options and payloads, and run until all execution paths finish. This can cover all the labeled critical states. Without state merging, it took 13.5 hours on average to finish symbolic execution on Linux kernel v4.4. After enabling state merging, it takes less than 4 minutes to finish, and the total number of execution paths decreases from 1,219,938 to 1386. In addition, without state merging, it takes more than a week to compare the rather huge models between Linux kernel v4.4 and v5.4. After enabling state mering, it takes only 15 seconds to compare the two versions.

We analyze 5 major LTS versions of the Linux kernel over the past decade from 3.0 to 5.10, viz., 3.0, 3.10, 4.4, 5.4, and 5.10, and found 9 discrepancies, including 4 previously unknown discrepancies, which all can be exploited. We also validate our findings with the commit history of the Linux kernel, confirm that the discrepancies discovered are true, and did not find any new discrepancies. Most of the discrepancies were introduced around 2012, while some newer ones were introduced around 2017. A major reason contributing to these discrepancies, is the change proposed in RFC 5961 [6]. There are also other reasons stemming from buggy implementations, performance improvements and compatibility with other operating systems.

In order to understand the effectiveness of Themis in defending against evasion attacks, we implement 34 evasion strategies, summarized from previous work [1, 5, 7, 8] and this work. We even design composite strategies that leverage multiple discrepancies in a single connection. Our Themis-enabled NIDS can successfully detect all the evasion attacks with a 100% success rate, outperforming a recent state-of-the-art machine-learning-based mitigatgion, CLAP [11], which only has an AUC-ROC of 0.963 in detecting NIDS evasion attacks.

In addition, we evaluate the overheads incurred due to Themis at runtime. We use 8-day network traces from the MAWI Traffic Archive [4], from April 25 to May 1, 2021, and April 7, 2020 (i.e., the dataset used by CLAP). We found only a very small fraction of natural/benign traffic contain packets that cause ambiguities (due to diverse endhost implementations). As for the resulting overhead, we find that compared to the original Zeek, our robust version incurs only about 1.07% additional processing time, indicating only negligible levels of operational cost. In comparison, Themis is 30 times faster than CLAP, due to the computational cost of deep learning model employed by CLAP.

5 CONCLUSIONS

In this work, we aim to defend against attacks that seek to evade network intrusion detection systems, by exploiting the discrepancies between its TCP implementation and that at a targeted end server. We design a novel lightweight system Themis which is extremely effective in defending against such attacks. It contains an offline phase, where it identifies and models discrepancies in TCP implementations across OS versions using symbolic execution. The models are then employed at runtime, and by applying a non-deterministic automaton the proper implementation versions are forked to handle packets correctly and block evasion attempts. Themis is extremely effective and is able to block all known evasion attempts to date with negligible additional overhead on a NIDS.

REFERENCES

[1] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. 2019. Geneva: Evolving Censorship Evasion Strategies. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 2199–2214. https://doi.org/10.1145/3319535.3363189

[2] Vitaly Chipounov, Volodymyr Kuznetsov, and George Candea. 2011. S2E: A Platform for In-vivo Multi-path Analysis of Software Systems. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (Newport Beach, California, USA) *(ASPLOS XVI)*. ACM, New York, NY, USA, 265–278. https://doi.org/10.1145/1950365.1950396

[3] Volodymyr Kuznetsov, Johannes Kinder, Stefan Bucur, and George Candea. 2012. Efficient State Merging in Symbolic Execution. In *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation* (Beijing, China) *(PLDI '12)*. Association for Computing Machinery, New York, NY, USA, 193–204. https://doi.org/10.1145/2254064.2254088

[4] MAWI [n.d.]. *MAWI Working Group Traffic Archive.* https://mawi.wide.ad.jp/mawi/

[5] Thomas H Ptacek and Timothy N Newsham. 1998. *Insertion, evasion, and denial of service: Eluding network intrusion detection.* Technical Report. SECURE NETWORKS INC CALGARY ALBERTA.

[6] Anantha Ramaiah, R Stewart, and Mitesh Dalal. 2010. *Improving TCP's Robustness to Blind In-Window Attacks.* RFC 5961. RFC Editor. 1–19 pages. https://www.rfc-editor.org/rfc/rfc5961.txt

[7] Zhongjie Wang, Yue Cao, Zhiyun Qian, Chengyu Song, and Srikanth V. Krishnamurthy. 2017. Your State is Not Mine: A Closer Look at Evading Stateful Internet Censorship. In *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17)*. ACM, New York, NY, USA, 114–127. https://doi.org/10.1145/3131365.3131374

[8] Zhongjie Wang, Shitong Zhu, Yue Cao, Zhiyun Qian, Chengyu Song, Srikanth V Krishnamurthy, Kevin S Chan, and Tracy D Braun. 2020. SymTCP: eluding stateful deep packet inspection with automated discrepancy discovery. In *Network and Distributed System Security Symposium (NDSS)*.

[9] Z3Prover/z3 [n.d.]. *The Z3 Theorem Prover.* https://github.com/Z3Prover/z3

[10] Zeek [n.d.]. *The Zeek Network Security Monitor.* https://zeek.org/

[11] Shitong Zhu, Shasha Li, Zhongjie Wang, Xun Chen, Zhiyun Qian, Srikanth V. Krishnamurthy, Kevin S. Chan, and Ananthram Swami. 2020. You Do (Not) Belong Here: Detecting DPI Evasion Attacks with Context Learning. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies* (Barcelona, Spain) *(CoNEXT '20)*. Association for Computing Machinery, New York, NY, USA, 183–197. https://doi.org/10.1145/3386367.3431311