A Formal Verification of Mutation-based Moving Target Defense

Muhammad Abdul Basit Ur Rahim, Qi Duan, Ehab Al-Shaer
mabdulb1,qduan,ealshaer@uncc.edu
University of North Carolina, Charlotte, NC

ABSTRACT

Static system configuration provides the advantage for the attackers to discover the vulnerabilities of the system and launch attacks. Moving Target Defense (MTD) can break this asymmetry for the defenders by mutating certain configuration parameters proactively and at the same time maintaining the run-time correctness and operational integrity. MTD is essentially distributed by nature since the processes or actions in MTD are executed in an interleaved manner. For any distributed system, it is important to verify the correctness and integrity, since they may be jeopardized by design errors or run time inconsistencies. In this work we present a framework for formal verification of MTD techniques. We describe MTD techniques with formal ontology and model the system behaviors with timed automata, and verify the correctness, liveness, fairness and deadlock-free properties of the system. We use Random Host Mutation (RHM) as the case study or MTD formal verification. Our experimentation validates the feasibility and scalability of the formal verification framework.

1 INTRODUCTION

Moving target defense (MTD) allows the cyber system to proactively defend against a wide-scale vector of sophisticated attacks by dynamically changing the system parameters and defense strategies in a timely and economical fashion [1]. It can provide robust defense by deceiving attackers from reaching their goals, disrupting their plans via changing adversarial behaviors and deterring them through prohibitively increasing the cost for attacks. MTD is essentially distributed by nature since the processes or actions in MTD are executed in an interleaved manner. For any distributed system, it is important to verify the correctness and integrity, since they may be jeopardized by design errors or run time inconsistencies. Therefore, verification of the MTD technique is performed in earlier design phase to make the system consistent and secure.

In this work, we present a framework for formal verification of MTD techniques. We describe MTD techniques with formal ontology and model the system behaviors with timed automata. We use Random Host Mutation (RHM), which is a dynamic and randomized methodology that mutates over time and does not affect the reachability and performance of the network, case study of formal verification. The RHM is consisted of number of hosts, number of moving target gateways (MTG), router, and moving target controller. We have modeled RHM using timed automata which is used to specify the time for individual components of the system. The UPPAAL is a model checking tool for verification of real-time systems and results as satisfied or not-satisfied [2][4]. UPPAAL verifies the model against the extended computation tree logic (ECTL). If the property is not satisfied then UPPAAL shows the counter-example (a path and state where the property does not hold). Then a user can make appropriate modification for correctness of model.

More specifically, the RHM architecture is consisted of hosts, MTGs, DNS, router, and MTC. The hosts communicate with each other through MTGs. The MTG contains the translation table (*tt_mtg*) which contains virtual IP (*vIP*) of all hosts. While domain name server (DNS) contains real IP (*rIP*) of all hosts. The MTC mutates *vIP* after certain interval. Figure 1 shows the architecture of framework. The host requests to MTG to obtain *vIP* of destination host. Then MTG forwards request to DNS and gets back *rIP* of destination host. The MTG performs translation from the *rIP* to *vIP*. The MTG receives packet forward request from host and forwards to the router. The router forwards packet to receiving MTG along-with *rIP* of source and destination. At the end MTG forwards the packet to destination host using *rIP* and shares the sender *vIP* information. The hosts with receiving MTG send back the packet in a same way however both hosts can only see the *vIP* of each other instead of *rIP*.

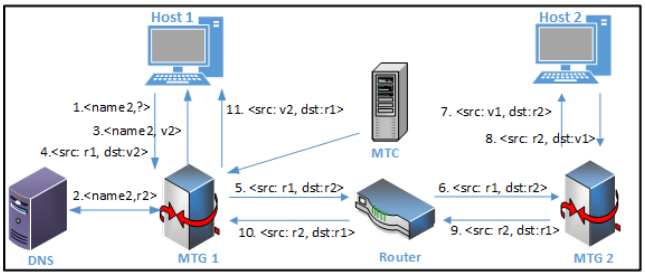


Figure 1: Architecture of MTD

2 RHM VERIFICATION - A CASE STUDY

We have verified the RHM to ensure the correctness of design. The RHM is modeled using timed automata which is used to specify the time for individual components of the system. The model is verified against the user requirements which are specified in form of ECTL properties. UPPAAL performs verification of model and results as satisfied or not-satisfied. If the property is not satisfied then UPPAAL shows the counter-example (a path and state where the property does not hold). Then a user can make appropriate modification for correctness of model. Figure **??**(b) describes the methodology for verification of RHM. Designing the generalized model for RHM is quiet interesting because the instances of each

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
HotSoS, April 1–3, 2019, Nashville, TN, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7147-6/19/04…$15.00
https://doi.org/10.1145/3314058.3317723

automaton interact with instances of other automata. In this way, the generalized model has made easier the verification of different sizes of networks.

3 MTD μ-CHECKER

MTD μ-Checker ensures the correctness of dynamic and randomized MTD over various sizes of networks and parameters. The mutation process has made RHM dynamic and randomized and that's why the design must satisfy the correctness properties. Our results prove the correctness, effectiveness, and reliability of RHM. The initialization process makes sure that the network is correctly designed before simulation and implementation. It makes sure that every component gets equal opportunity to communicate with the respective components. We have found that the RHM satisfies all the correctness properties and ensures the correctness over various mutation intervals, MTG propagation delays, and DNS delays. We have generalized the design of RHM, therefore, we are able to make it scalable. We only need to specify the number of hosts, the number of (MTG), and number of hosts over a subnet, DNS delay, and time interval for MTC. Moreover, the RHM framework is modeled, simulated, and verified over various sizes of networks which contains up to 200 mutation hosts, up to 100 MTGs, and up to 40 hosts over a subnet. The framework is also evaluated over various important parameters which can affect the reachability. The MTC mutates the *vIP* after certain time intervals, the DNS can hold the *rIP* request for an interval, and the MTG can hold the packets for a certain delay (propagation delay) before forwarding to the router. The MTD μ-Checker verified the RHM over different delays.

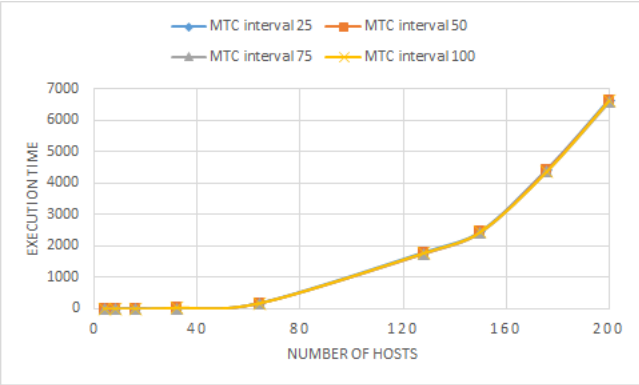


Figure 2: Reachability analysis over MTC intervals

We have verified the reachability (no packet-loss), liveness (*vIP* mutation after certain interval), fairness (equal distribution of resources), and deadlock-freeness properties. Figures 2 and 3 show the reachability and liveness analysis respectively. We analyse the execution time and memory consumption for all these properties. These properties consume time up to 6700 milliseconds and up to 1.4 GB memory. The RHM does not use enough resources that's why MTD μ-Checker is more scalable. The dynamic RHM can beat scanning by frequently mutating the random IP addresses without any packet loss or affecting the network.

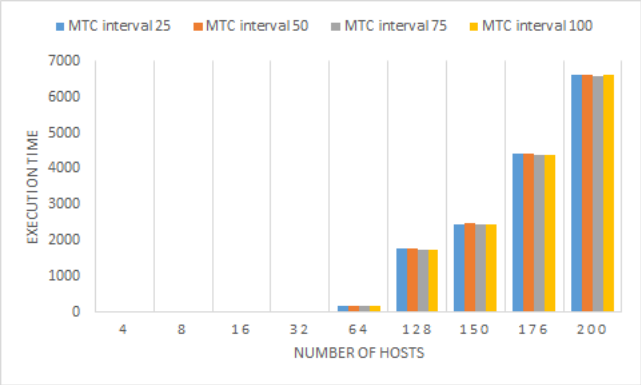


Figure 3: Liveness analysis over MTC intervals

3.1 Design Correctness

The RHM design is verified using ECTL temporal properties. The reachability, liveness, and deadlock-freeness properties are specified using ECTL.

3.1.1 Reachability. Reachability describes that every good state is reachable and every bad state is unreachable. In our model PacketLoss state of MTG is a bad state where the model stops and does not progress. The model should avoid this state.

3.1.2 Liveness. The liveness describes that system is progressing to achieve specific goal. In our model, the RHM should mutate the *vIP* after every specific interval.

3.1.3 Deadlock-freeness. Deadlock-freeness ensures that the system is not stopped and it is always in progressing state. The system is in deadlock state when it stops in a state and does not progress to other states [3].

3.1.4 Fairness. Equal distribution of resources is knowns as fairnes [3]. In RHM, every host must get a new *vIP* after certain interval.

Designing and simulating a system are not enough to ensure the correctness and reliability of a system. The correct design still needs the correct configuration over different sizes of networks. We have evaluated the novel framework over various sizes of networks, DNS delays, and mutation intervals. However, the results almost remain the same for all these specifications. All the correctness properties satisfy the design of RHM. In future, we will extend RHM with routing mutation, and analyze the composition for design correctness.

REFERENCES

[1] Ehab Al-Shaer, Qi Duan, and Jafar Haadi Jafarian. 2012. Random Host Mutation for Moving Target Defense. In *SecureComm*, Vol. 106. Springer, 310–327.
[2] Gerd Behrmann, Alexandre David, and Kim G. Larsen. 2004. A Tutorial on Uppaal. In *In proceedings of the 4thInternational School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT'04)*. Springer-Verlag, Berlin, Heidelberg.
[3] S. Sultana and F. Arif. 2017. Computational Conversion via Translation Rules for Transforming C++ Code Into UPPAALâĂŹs Automata. *IEEE Access* 5 (2017), 14455–14467. https://doi.org/10.1109/ACCESS.2017.2728860
[4] Muhammad Abdul Basit ur Rahim and Fahim Arif. 2016. Translating Activity Diagram from Duration Calculus for Modeling of Real-Time Systems and its Formal Verification using UPPAAL and DiVinE, Vol. 35(1). Mehran University Research Journal of Engineering and Technolog, Berlin, Heidelberg, 139–154.