NetQi: A Model checker for Anticipation Game

Elie Bursztein

LSV, ENS Cachan, CNRS, INRIA
eb@lsv.ens-cachan.fr

Abstract. NetQi is a freely available model-checker designed to analyze network incidents such as intrusion. This tool is an implementation of the anticipation game framework, a variant of timed game tailored for network analysis. The main purpose of NetQi is to find, given a network initial state and a set of rules, the best strategy that fulfills player objectives by model-checking the anticipation game and comparing the outcome of each play that fulfills strategy constraints. For instance, it can be used to find the best patching strategy. NetQihas been successfully used to analyze service failure due to hardware, network intrusion, worms and multiple-site intrusion defense cooperation.

1 Introduction

Using model-checking for intrusion analysis is an active area of research [7, 8, 6]. Models and tools have been developed to analyze how an intruder can combine vulnerabilities as step-stones to compromise a network. However, Anticipation Game (AG) [4, 3] is currently the only game framework for network security. Netqi [2] is the complete implementation of AG. With Uppaal Tiga [1], NetQi is the only model-checker for timed ATL (Alternating-time Temporal Logic).

Anticipation games are an evolution of attack graphs based on game theory. More specifically they are timed games based on a TATL variant designed for network security analysis purpose. An AG is a kripke structure where each node represents a network state and the transition between the nodes models the players (administrator and intruder) actions on the network. Therefore an AG models the evolution of the network as the result of players actions on it. Typically it is used to analyze how the network will be impacted by various attacks and how administrator actions can counter them. Using Anticipation game instead of attack graph offers the following advantages.

First it allows to model the concurrent interaction of the intruder and the administrator with the network. For example it is possible to model that the intruder is trying to exploit a vulnerability while the administrator is trying to patching.
Secondly the use of timed rules allows to model the temporal dimension of the attack. It captures that each interaction with the network requires a different time. For instance developing and launching an exploit is somewhat slower than downloading and launching a public available one. Modeling the time also allows to model the so called "*element of surprise*" [5], which occurs when one player takes the other by surprise because he is faster. For example when the administrator is patching a service she can be taken by surprise by the intruder if he is able to exploit the vulnerability before the patch is complete.

Finally since anticipation game has been designed for network security analysis, it takes into account network topological information such as dependency between network services. This allow to model attack *collateral effects*. For example that when a DNS server is unavailable by collateral effect the web service is merely available because the DNS resolution failed.

The reminder of this paper is organized as follows. Sect. 2, details how the anticipation game framework and NetQi differs from previous tools and work on TATL and what makes NetQi effective for network security analysis. In Sect. 3 discusses how NetQi is implemented and presents some of its main optimizations. Sect. 4 presents an example of the game analyzed by NetQi. In Sect. 5 we conclude and give future directions.

2 The Framework

The AG framework differs from standard timed games in several points. The two most prominent features are the use of a dual layer structure and the use of a compact model description. In AG, the lower layer is used to model network information. It is composed of two parts, a graph used to model network service dependency that is fixed over time and a set of states that is meant to evolve over the time. States, which are Boolean values, are used to describe nodes information such as which are compromised, and which are vulnerable. States value are changed by player action effect. The upper layer is a standard timed game structure used to model the evolution of the network layer due to player action. Legal actions for each player are described by sets of timed rules. Each rule is of the form:

$$\Gamma_x : \mathbf{Pre}\ F \xrightarrow{\Delta,\ \text{p, a, c}} P$$

where F is the set of *preconditions* that needs to be satisfied in order to use the rule. Δ is the amount of time needed to execute the rule, p is the player that uses the rule, a is the rule label (string), c is the rule cost. P is the rule *post-condition* that states rule effects and Γ_x is the rule location. Locations are used to restrict rules to a specific set of network nodes. A example of rules set is given in Section 4. Describing the model only with the network initial state and a set of rules relieve the security analyst from the tedious and error prone burden of explicitly describing each network states and transitions. While working on large network, explicitly describing each network state is almost impossible because such game have millions of states. Therefore in AG the model-checking algorithm uses the set of rules to infers automatically every transitions and network states reachable from the network initial state. As a result, it is possible to express very large and complex model in a very compact form which is handy while working on large network and complex attack. Additionally modeling players action by rules allows to capture security expert reasoning in an intuitive manner as it allows to write things like: if a service is vulnerable then an intruder can compromise it, and if a service is compromised then an administrator can isolate it by using a firewall.

Beside anticipation games specificity, the main differences between Uppaal Tiga and NetQi are the game representation and the analysis goal. While Uppaal Tiga requires an expended model, NetQi uses a concise model described above. The other difference is that Uppal Tiga verifies TATL property whereas NetQi searches for strategy that matches player objectives. The use of player objectives is required to select between all the play that fulfills the TATL property, the one that will be the most efficient. For example an administrator wants to find a play that allows to patch her network efficiently but she probably wants the one that allows her to patch it efficiently for the minimal cost. Being able to provide the most effective solution in term of time and cost is a central issue in network security. Many natural questions that arise in network security require such answer for instance : which attack will cause the most damage ? what is the patching strategy that will minimize my loss ? Specifying analysis goal as strategy objective is possible because each rule has a cost and a possible reward. The rule reward is based on the value of the targeted network node. The use of time, cost and reward allows to take into account the financial and temporal dimension of the attack in the analysis. Theses objectives are described in the game file by a strategy objective tuple. This tuple is:

$$S : (name, player, objectives, objectives\ order, constraints, location)$$

where $name$ is the strategy name, $player$ specifies for which player this strategy is, $objectives$ are strategy objectives based on player and opponent cost and reward, $objectives\ order$ is used to indicate which objectives are the more important, $constraints$ is a set of LTL constraints used to determine if a play should be considered as a potential strategy, and finally $location$ specifies which group of service the strategy has to consider. There are five possible objectives : player cost and reward, opponent cost and reward and strategy time. A typical set of objectives is player cost minimization and opponent cost maximization. This is equivalent to search for the less costly strategy against opponent best play because in network security there is a direct correlation between the cost of the attack and its efficiency. For finding a new vulnerability (0 day exploit) is more expensive than reusing a public one. On the other hand the 0 day exploit is more efficient because there is no patch or intrusion detection rule to catch it. Constraints can be used to express that a strategy is valid if and only if no host was compromised during the play or that at the end of the strategy at least one host is compromised for instance. Note that NetQi is able to model-check TATL property as well. Model-checking a property is used for instance to prove that given an initial network state and a given set of rules, whatever the intruder do, he cannot compromise a given set of services.

3 The implementation

NetQi is composed of two parts: the game engine written in C for speed, and the frontend gui written in java for portability. NetQi can be run on Linux, OSX and Windows. NetQi takes as input a game file (see figure 2) that describes the network information, the strategy objectives, and player rules. It returns the strategy found either on the standard output or in a XML file. The Gui is used to build and display the game file and analyze the output file. It draws a visual representation of the lower layer graph and the strategy timeline.

NetQi uses a search in depth strategy based on rules time and node value. This search strategy is driven by the idea that since the fastest rule wins in timed game, if the player is not able to reach his strategy objective with his fastest rules, it is unlikely that he will reach them with slower rules. NetQi does not suffer from memory explosion because it is used to find memoryless strategy. One of the most effective optimization used in NetQi at runtime, is the use of early cut, which apply when the strategy constraints use a standard LTL $\Box$ operator. This operator is used to have a constraint that holds during the entire play. In this case, the strategy constraints are evaluated dynamically at each step on the play. If a step violates the strategy constraint, NetQi cuts the play and immediately backtracks, because this play will never be an acceptable strategy. As shown in figure 1, this optimization can reduce greatly the number of plays and states considered during the analysis. The standard defense strategy uses the $\Box$ operator to ensure that no host is ever compromised.

Early cut	plays	states	time (s)
No	6 113 459	18 444 859	515
Yes	124 047	366 829	9

Fig. 1. Early cut impact on performance

Before the execution of the analysis, NetQi perform a static analysis of the game file to determine if some optimization can be made. For example the static analysis involves removing the set of rules that will never be executed because there pre-conditions are in conflict with strategy constraints. For example, if the strategy constraints requires that not a single service is ever compromise then all the rules that requires in their pre-conditions that one service is already compromised are removed.

4 Example

An hello world example of game file is presented in figure 2, the resulting strategy found by NetQi is depicted in figure 3 . The game file is composed of five sections.The first section is the general options section, in the example only the number of lower-layer node to consider is specified but other options exist such as timeout. The second section contains the set of states used and their initial values. In the example two sets are used: The $Vuln$ set is used to model that the node 1 is vulnerable and the set $Compr$ is used to model that no node is compromised at the beginning. The third section is the rule section. In the example 3 rules are used. The first one states in its precondition that if a node is vulnerable ($Vuln$) , then in 3 units of time for a cost of 200$ the intruder (I) can compromise ($Compr$) it (rule effect). The two other rules state that the administrator can patch a vulnerable ($Vuln$) service. They differ by their costs and execution time. The fast rule requires less time, 1 instead of 6, but for a greater cost: 5000$ instead of 500$. The fourth section is the dependency graph description. Here only the node 1 value is specified (42) but other type of information can be specified here such as dependency between nodes and node location. Finally the last section is the strategy objectives. The administrator (A) example strategy called `Defense` aims at minimizing player cost and ensuring by constraints that in the selected play no node is ever compromised. The play considered as the best strategy, visible on the right of the figure, is as expected the one that uses the fast patch rule. Note that the intruder is taken by surprise by the administrator and its compromise rule fails.

```
nodes=1
<sets>
Vuln:1
Compr:false
</sets>
<rules>
I:3:Compromise:Vuln=>Compr=200
A:6:Patch slow:Vuln=>!Vuln=500
A:1:Patch fast:Vuln=>!Vuln=5000
</rules>
<graph>
1=42
</graph>
strategy(Defense,A,MIN(Cost),Cost,|!Compr)
```

Fig. 2. The Game file

Time	Player	Action	Rule	Target	Reward	Cost
0	Intruder	choose	Compromise	1	-	-
0	Admin	choose	Patch fast	1	-	-
1	Admin	**execute**	Patch fast	1	42	5000
3	Intruder	`fail`	Compromise	1	0	200

Fig. 3. The resulting strategy

5 Conclusion

NetQi has been run successfully on complex network attack scenario. For example to find a multiple-site defense strategy, that involves 5 sites with 10 services each and 11 action rules analysis, the run-time on a Linux core2 2.9 GHz is less than 5 minutes. We expect that with a suitable service collapsing abstraction, NetQi will scale to much larger network. NetQi is very stable and therefore can be run on very large example without crash or memory leak. So far, the biggest successful analysis was a game with 1 148 893 596 distinct states. This analysis took 527 minutes which is an average of

36290 states per seconds. In addition because NetQi game files are easy to write even for a non game theory specialist, we hope it will be of use and interest to security experts. For futher information on NetQi, including downloads, examples, and documentation, see `http://www.netqi.org`

References

1. G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In W. Damm and H. Hermanns, editors, *CAV*, volume 4590 of *Lecture Notes in Computer Science*, pages 121–125. Springer, 2007.
2. E. Bursztein. Netqi http://www.netqi.org.
3. E. Bursztein. Network administrator and intruder strategies. Technical Report LSV-08-02, LSV, ENS Cachan, Jan 2008.
4. E. Bursztein and J. Goubault-Larrecq. A logical framework for evaluating network resilience against faults and attacks. In *12th annual Asian Computing Science Conference (ASIAN)*, pages 212–227. Springer-Verlag, Dec. 2007.
5. L. de Alfaro, M. Faella, T. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *14th International Conference on Concurrency Theory*, volume 2761 of *LNCS*, pages 144–158. Springer-Verlag, 2003.
6. C. Ramakrishan and R. Sekar. Model-based analysis of configuration vulnerabilities. In *Journal of Computer Security*, volume 1, pages 198–209, 2002.
7. R. W. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 156–165, Washington, DC, USA, 2000. IEEE Computer Society.
8. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 273 – 284, Washington, DC, USA, 2002. IEEE Computer Society.