Verification of the Protection Services in Antivirus Systems by Using NuSMV Model Checker

Monire Norouzi¹, Saeed Parsa²*

¹Department of Computer Engineering, Shabestar Branch, Islamic Azad University, Shabestar, Iran
²Department of Software Engineering, Iran University of Science and Technology, Tehran, Iran

ABSTRACT

In this paper, a model of protection services in the antivirus system is proposed. The antivirus system behavior separate in to preventive and control behaviors. We extract the properties which are expected from the model of antivirus system approach from control behavior in the form of CTL and LTL temporal logic formulas. To implement the behavior models of antivirus system approach, the ArgoUML tool and the NuSMV model checker are employed. The results show that the antivirus system approach can detects fairness, reachability, deadlock free and verify some properties of the proposed model verified by using NuSMV model checker.

KEYWORDS

verification, antivirus system, protection services, kripke structure, NuSMV

1. INTRODUCTION

Today, antivirus software [1] is very important in business and software development, because in any computer, system should install a security application on itself to maintain regularize and secure of its data. In the recent years, many attacks [2] are occurred to variety systems like bank servers, military systems and home systems by variety Viruses and Malwares [3]. Information maintenance and prevent to data accessibility is the cause of attacking and destroying data which has been occurred by Invasive malware [4, 5] widely and suddenly. So, the customers want to a reliable secure software [6].

The attacks in system security [7] section is divided in to two parts. The first set of attacks are enabled to eliminate security files or security applications of a computer system. After disabling these files or applications, these attacks cannot access to the important data in the computer system [8]. The second set of attacks can remove a set of special files and destroying important files. These attacks are executed and handled from intelligence agencies and hacker on computer systems of the users. These attacks have not needed to destroy system security of computer. But, these attacks are executed and run by hiding themselves in URL web addresses or in created files using routing software such as Microsoft office, adobe reader, win zip, etc. By notice to these attacks, a computer system need to an online security system. Of course, an offline security system prevent to data accessibility in computer system slightly. Unfortunately, by progression of technology and software system complexity attackers have discovered some new
vulnerable areas in systems that informally called "holes". So, by this ways they disturb system security.

By the above reasons, we find that testing and verifying the security applications such as the antivirus systems is very important and essential in Security Discussion [9] of the computer systems. There are some antivirus applications in software development market such as Bitdefender\(^1\) Antivirus, Kaspersky\(^2\) Antivirus, Avira\(^3\) Antivirus and etc. Of course, computer viruses [10], Spywares [11], Trojans, Worms [12] and other new malwares debut every day.

Model checking technique of the software systems verification is an automatic technique [13]. It has many advantages over simulation, testing, and deductive reasoning. Formal verification techniques have an important role in validation processes of the software systems. Formal verification of Antivirus System approach is essential as it protects all type of the system from the attacks and viruses. State space searching involves exhaustive testing or scenario analysis.

In this paper, a protection services in Antivirus System approach has been presented. Our model separated in to two behaviors: preventive behavior and control behavior. In particular, the contributions of this paper are:

- Proposing an ideal Antivirus System based on Avira Antivirus approach.
- Presenting an Antivirus System behavior model to decouple preventive and control behavior of Antivirus System approach.
- A formal approach extracted from the expected properties of Antivirus System approach from control behavior in the form of Temporal Logic formulas.
- Implementing the behavior models of Antivirus System approach by using ArgoUML\(^4\) tool and NuSMV\(^5\) model checker.

The rest of this paper is organized as follows. In section 2, we discuss some related works in formal verification. Section 3 describes the preventive and control behaviors of the Antivirus System. Section 4 presents a conversion of behavioral models to Kripke structure. Furthermore, the behavioral properties of behavioral models are defined by using linear temporal logic and computation tree logic languages. These properties can be checked by the specification of control behavior. In section 5 we implement the proposed behavioral models by ArgoUML tool and NuSMV model checker. Finally, Section 6 shows conclusions and future works.

2. RELATED WORKS

Model checking and verification of software and hardware systems have becoming an interesting topics in the past few years and many researchers have studied them around the world.

Ravn, et al. [14] presented a formal method of the Web Services Atomic Transaction (WS-AT) protocol. They described an algorithm for achieving agreement on the outcome of a distributed transaction. The protocol has been verified by using UPPAAL\(^6\) model checker. Their model is based on a formalization technique by using the mathematical language of the Temporal Logic of Actions (TLA+).

\(^1\) http://www.bitdefender.com/site/view/our-story.html
\(^2\) http://www.kaspersky.com/about
\(^3\) http://www.avira.com/en/for-home
\(^4\) http://argouml-stats.tigris.org/
\(^5\) http://nusmv.fbk.eu/
\(^6\) www.uppaal.org
As another new researches in Web services Kova, et al. [15] designed and verified a behavior of composite Web services based on two behaviors, control and operational. These behaviors are communicated together by using conversation messages. They used to state charts for modelling the composite Web services and verifying the synchronization of the conversations by using NuSMV model checker, too.

Yeung [16] described a formal modeling approach for choreography-based web services composition and conformance verification. Apart from Web Services Choreography Description Language (WS-CDL) and Web Services Business Process Execution Language (WS-BPEL), their approach also supported the use of visual modeling notations such as UML in modeling choreographies and orchestrations.

Bentahar, et al. [17] modeled the composite web services based on a division of interests between operational and control behaviors. Some favorable properties such as deadlock freedom, safety and reachability have been analyzed. The proposed behaviors have been converted to Kripke structure by using model checking techniques based on BDD.

El-Menshawy, et al. [18] presented a new technique for verifying multi-agent commitment-based protocols by using model checking techniques. They used a reduction technique to formally transform the model checking Action Computation Tree Logic (ACTL). They proved that the reduction technique is sound and fully implemented it on top of the CWBN model checker.

Xitong, et al. [19] proposed a Petri Net (PN) approach to analyzing behavioral compatibility and similarity of web services. Also they developed the Service Workflow Net (SWN) formalism based on colored Petri Net in order to model Web services. Then formal definition of behavioral compatibility is presented based on the SWN model. Finally, the authors used PIPE editor tool [20] to automate the verification of behavioral similarity and equivalence of Web services.

Souri and Navimipour [21] proposed an adapted antivirus system approach to address multi-attribute queries in grid computing. They presented a behavioral model for their proposed approach that separate into data gathering, discovery and control behaviors. So, they used to Kripke structure for modeling these behaviors and verified their behavioral models by using NuSMV model checker.

Table 1 presents some related works in verification approaches.

Table 1. Summary of the modeling and verification methods of the related works

<table>
<thead>
<tr>
<th>Papers</th>
<th>Case Study</th>
<th>Modeling</th>
<th>Temporal logic</th>
<th>Model Checker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xitong, et al. [19]</td>
<td>Web Services</td>
<td>Petri Net</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bentahar, et al. [17]</td>
<td>Composite Web Services</td>
<td>Java Converter Tool</td>
<td>LTL &amp; CTL</td>
<td>NuSMV</td>
</tr>
</tbody>
</table>
3. ANTIVIRUS BEHAVIOR MODEL

This section presents an antivirus behavior model by using formal and model checking techniques, which separates its behavior into preventive behavior and control behavior. We use statechart semantics to modeling preventive and control behaviors.

![Figure 1. Model of preventive behaviour](image)

Figure 1 shows the preventive behavior of the ideal antivirus software. The preventive behavior starts from protection approach by initial state system protection and specifies system mode by checking online or offline mode. The offline mode shows PC Protection and includes protection approach for two states Real-time Protection and System Scanner. In the state Real-time Protection, system protected automatically in the windows system directory (WSD) in path: C:\windows\system32. The state detecting file as a type of Detection approach discover any type of infections in suspicious files and send it to state check cleaning operation as a type of Identification approach. If the infected file deleted, then the Removal approach is executed and system receives the safe result in this process. If suspicious file ignored for any reason, then system receives unsafe result in this process. The recycling operation divides into two parts. In the first part, by finding infection position, system performs Identification approach and removes the infection from the file so the file is safe and Removal approach is executed. In the second part, a suspicious file is found and state check clearing operations executes Identification approach. If this file is deleted then Removal approach receives the safe result, otherwise system receives the unsafe result.
The control behavior navigates the execution flow of the approaches of antivirus system. In the figure 2, the control behavior of the antivirus system presents a number of states extracted from the preventive behavior by using the Buchi automata. These states include Not Activated, Activated, Process, and Recognition, Done and Aborted.

3.1 EXPLANATION OF ANTIVIRUS SYSTEM APPROACHES

In the antivirus system behaviors we define four approaches. These approaches are Protection Approach, Detection Approach, Identification Approach and Removal Approach. Connections between preventive and control behaviors is possible by these approaches. Each of these approach are mapped on the states of preventive behavior and control behavior. The Protection Approach includes Not Activated and Activated states in control behavior and also System Protection, PC Protection and Real-Time Protection in preventive behavior. Detection Approach includes Process state in control behavior and Detecting Files state in preventive behavior. Identification Approach includes Recognition state in control behavior and Check Cleaning Operations in preventive behavior and finally the removal Approach includes Done state in control behavior and Deliver Safe Status state in preventive behavior.

4. VERIFICATION OF PREVENTIVE AND CONTROL BEHAVIORS

In this section, we describe the model checking mechanism for verifying the behavior models of the antivirus system by using Kripke structure of behaviors definition. We first define the Kripke Structures of the behavioral models.

A Kripke structure is a nondeterministic finite state machine that is used in model checking to represent system behavior and is a 4-tuple $KS = (S, s_0, Tr, L)$, where:

- $S$ is a finite set of states,
- $s_0 \in S$ is an initial state,
- $Tr \subseteq S \times S$ is a transition relation that $\forall s \in S: \exists s' \in S : (s, s') \in Tr$.
- $L: S \rightarrow 2^{AP}$ is labeling function.

Now we present expected rules in control behavior by using symbolic model checking techniques. For checking the properties in the Kripke structure $MK$ which is associated with the antivirus system behaviors, these properties can be formulated by using LTL and CTL formulas.

Now, by using the following initials we can define logical properties for model checking: $In= \{ \text{Not Activated: Na; Activate: Ac; Process: Pr; Identification: Id; Invoked: In; Received: Re; Aborted: Ab; Done: Do; End: En} \}$.

Let $\rightarrow$ be the logical implication. We define some examples of CTL properties which can be verified for the control behavior:

- \( C_1 = \mathit{AG} (\mathit{Na} \rightarrow \mathit{AX Ac}) \). There is always a path from state Not Activated to state Activated.
- \( C_2 = \mathit{AG} (\mathit{Ac} \rightarrow \mathit{AXAF} (\mathit{Ac} \mathcal{V} \mathit{Pr})) \). There is always a path from state Activated to state Activated or process.
- \( C_3 = \mathit{AG} (\mathit{Pr} \rightarrow \mathit{AX AF} (\mathit{Id} \mathcal{V} \mathit{Ac})) \). There is always a path from state Process to state Identification or Activated.
- \( C_4 = \mathit{AG} (\mathit{In} \rightarrow \mathit{AX AF} (\mathit{Ab})). \) There is always a path from state Invoked to state Aborted.
- \( C_5 = \mathit{AG} ((\mathit{Do} \mathcal{V} \mathit{Ab}) \rightarrow \mathit{AX AF} (\mathit{En})). \) There is always a path from state Done or Aborted to state End.
- \( C_6 = \mathit{AGEF} (\mathit{Ac}). \) State Activated is always potentially reachable.
- \( C_7 = \mathit{AGEF} (\mathit{Ab} \mathcal{V} \mathit{Re}). \) State Aborted or state Received is always potentially reachable.
- \( C_8 = \mathit{AGEF} (\mathit{Pr} \rightarrow \mathit{Ac}). \) State Activated comes after state Process is always potentially reachable.
- \( C_9 = \mathit{AGEF} (\mathit{In} \rightarrow \mathit{Re}). \) State Received comes after state Invoked is always potentially reachable.
- \( C_{10} = \mathit{AGEF} (\mathit{Do}). \) State Done is always potentially reachable.
- \( C_{11} = \mathit{AGEF} (\mathit{En}). \) State End is always potentially reachable.

Examples of LTL properties that can be verified from the control behavior are:

- \( L_1 = \mathit{G} (\mathit{Na} \rightarrow \mathit{X} (\mathit{Ac})). \) Always an Activated state comes after a Not Activated state.
- \( L_2 = \mathit{G} (\mathit{Do} \mathcal{V} \mathit{Ab}) \rightarrow \mathit{XF} (\mathit{En})). \) Always a End state comes after a Done state or Aborted state.
- \( L_3 = \mathit{G} (\mathit{Na} \rightarrow \mathit{XF} (\mathit{Do} \mathcal{V} \mathit{Ab})). \) Always a Done state or Aborted state comes after Not Activated state.
- \( L_4 = \mathit{G} (\mathit{Pr} \rightarrow \mathit{XF} (\mathit{Ac} \mathcal{V} \mathit{Id})). \) Always an Activated state or Identification state comes after a Process state.
- \( L_5 = \mathit{G} (\mathit{Id} \rightarrow \mathit{X} (\mathit{Pr})). \) Always a Process state comes after an Identification state.

After defining some examples of CTL and LTL rules, we present a translation of the preventive behavior into a Kripke structure and then for transforming the initial kripke model to the kripke structure model, we conform the preventive behavior states to the corresponding state in the control behavior. This translation is from the initial kripke model of preventive behavior states to the symbols \( \mathit{Na}, \mathit{Ac}, \mathit{Pr}, \mathit{Id}, \mathit{In}, \mathit{Re}, \mathit{Ab}, \mathit{Do} \) and \( \mathit{En} \). The result of the reduced Kripke model is presented in figure 3.

Figure 3. The Reduced Kripke model of Preventive Behavior
5. SYSTEM IMPLEMENTATION

In this section, we implementation the proposed model by using ArgoUML tool that is a open source UML modeling tool [22] for behavior specification in figure 4. The behaviors represented in ArgoUML [23] are shown as statechart diagrams [24].

![Figure 4. Specification of antivirus system behaviors in ArgoUML tool](image_url)

Figure 4. Specification of antivirus system behaviors in ArgoUML tool

Figure 5 shows the translated SMV code from the Kripke structures which is written by Roudabeh tool[^25]. By using the Roudabeh tool [26], we can have translation of the reduced Kripke model to SMV code.

![Figure 5. The translation of SMV code by using Roudabeh tool](image_url)

Figure 5. The translation of SMV code by using Roudabeh tool

[^25]: http://ece.ut.ac.ir/fml/rebeca_verifier.htm
Figure 6 presents the results of the model checking of CTL properties by using NuSMV model checker.

![Figure 6. Checking CTL properties in NuSMV Interactive model](image)

In Figure 7, the LTL properties checked in NuSMV model Checker.

![Figure 7. Checking LTL properties in NuSMV Interactive mode.](image)

Also in Figure 8, we analyze the states and transitions of antivirus system approach in reachability and fairness conditions. By compute_reachable command we check state reachability, state fairness and transition fairness. The result shows that all states are reachable and fair and all transitions are fair and by check_fsm command we check the deadlock problem antivirus system model. It is showed that the antivirus system approach has not deadlock.
6. PERFORMANCE EVALUATION

In this section the proposed approach is evaluated according to the faithfulness of the formal models and their usefulness for model checking. The valid behaviors of antivirus system approach are satisfied by the CTL formulas in verification procedure. Also the incorrect traces checked by appropriated counterexample in the LTL formulas. The verification results of antivirus system model shows its completeness and soundness. Table 3 shows performance evaluation results of checking the model of antivirus system by NuSMV model checker tool.

Table. 2 Verification statistics for antivirus system approach.

<table>
<thead>
<tr>
<th>Property</th>
<th>Result</th>
<th>Time (s)</th>
<th>Memory (KB)</th>
<th>Temporal Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory in use (byte)</td>
<td>4678263</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of BDD variables</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of sifted variables</td>
<td>1100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of swapped variables</td>
<td>1800000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>total number of nodes</td>
<td>1300</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Also table 3 shows the evaluation results to verify the total number of properties in antivirus system approach which are obtained by NuSMV model checker tool.

Table. 3 Verification result of all properties in antivirus system approach (PC Intel Core Duo 6600, 2.4 GHz, 2GB RAM, Windows 7, NuSMV 2.4.3).

<table>
<thead>
<tr>
<th>Property</th>
<th>Result</th>
<th>Time (s)</th>
<th>Memory (KB)</th>
<th>Temporal Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG (Na → AX Ac)</td>
<td>Satisfied</td>
<td>0.016</td>
<td>26,236</td>
<td>CTL</td>
</tr>
<tr>
<td>AG (Ac → AXAF (Ac ∨ Pr))</td>
<td>Satisfied</td>
<td>11.778</td>
<td>42,952</td>
<td>CTL</td>
</tr>
<tr>
<td>AG (Pr → AX AF (Id ∨ Ac))</td>
<td>Satisfied</td>
<td>26.785</td>
<td>57,792</td>
<td>CTL</td>
</tr>
<tr>
<td>AG (In → AX AF (Ab))</td>
<td>Satisfied</td>
<td>1.935</td>
<td>17,792</td>
<td>CTL</td>
</tr>
<tr>
<td>AG ((Do ∨ Ab) → AX AF (En))</td>
<td>Satisfied</td>
<td>16.597</td>
<td>56,956</td>
<td>CTL</td>
</tr>
<tr>
<td>AGEF (Ac)</td>
<td>Satisfied</td>
<td>85.332</td>
<td>89,136</td>
<td>CTL</td>
</tr>
<tr>
<td>AGEF (Ab ∨ Re)</td>
<td>Not Satisfied</td>
<td>11.22</td>
<td>44,765</td>
<td>CTL</td>
</tr>
</tbody>
</table>
7. CONCLUSION

In this paper, we presented an antivirus system approach. We focus on the antivirus application and we show that in our proposed model, the real-time protection and web protection perform more secure. The antivirus system approach is divided into preventive and control behaviors. The expected properties of antivirus system are extracted from control behavior in the form of temporal logic formulas. Also we implemented the behavior models of antivirus system approach by ArgoUML tool and the NuSMV model checker. The results showed that the antivirus system approach can successfully detects fairness, reachability, deadlock free and verify all of the CTL and LTL rules. For future work, we try to extend this proposed mechanism for the other section of antivirus system as well as its formal verification and behavioral modeling.

REFERENCES


