Developing Knowledge-Based Security-Sense of Networked Intelligent Robots

M. Omar Faruque Sarker[1], ChangHwan Kim[1], Bum-Jae You[1], and Mohammed Golam Sadi[2]

[1] Intelligent Robotics Research Center
Korea Institute of Science and Technology(KIST)
P.O. Box 131, Cheongryang, Seoul, 130-650, Korea
writefaruq@gmail.com, {ckim, ybj}@kist.re.kr
[2] Computer Engineering Department
Hankuk Aviation University, Korea
sadi@hau.ac.kr

Abstract. Distributed processing capabilities of robotic-applications using state-of-the-art component-based middleware and mobile-agent software technologies are continuously helping to develop more powerful intelligent robots. These application development frameworks provide several attractive conceptual solutions for network-based operations, however, these benefits cannot be realized unless the appropriate security mechanisms are in place. In this paper, this important issue is addressed in a systematic way: firstly, by analyzing the key limitations of traditional security models, the new security requirements for agent-based networked robots are pointed out and secondly, an effective defense mechanism is devised by constructing a knowledge-based security-aware robot control model. The main contributions of this work are, to address the security problem in a knowledge-based approach and to show the roles and mechanisms of the proposed security architecture. The feasibility of our approach is examined through a case study on development of robot security-sense using Linux-based reliable security-tools.

1 Introduction

Network-based intelligent robots provide a new outlook to design and implement cost-effective and powerful solutions for a great number of users. Recent developments in Component Based Software Engineering (CBSE) and in state-of-the-art Mobile-Agent Software (MAS) [2] technologies are continuously creating new opportunities to build more powerful and cost-effective intelligent robots. These application development frameworks provide several attractive conceptual solution for application development: real-time capabilities, dynamic deployment, QoS etc., but they hardly satisfy the dynamic security requirement which is highly essential for implementing a mission critical networked robot. Generally security and privacy technologies are divided into three categories: prevention, avoidance and detection [3]. In this paper, we have covered mainly the security issues where privacy is implicitly considered in our security architecture.

B. Gabrys, R.J. Howlett, and L.C. Jain (Eds.): KES 2006, Part I, LNAI 4251, pp. 874–881, 2006.

Rest of this paper: Section 2 describes the traditional security models along with their key pros and cons. New security requirements and preview of our knowledge-based approach have been presented at Section 3. Section 4 discusses our proposed model. Section 5 provides the details of our implementation. Section 6 describes our case study of Linux-based Netfilter package to develop security sense of our robot. Finally, Section 7 concludes the paper with summary and outlook.

2 Related Works

Recent security researches on distributed computing, in general, propose architectures that offer security in a traditional way like role-based access control (RBAC) or in context-aware form. Some modified versions have also been proposed like Usage Control Model (UCON), Ubiquitous Context-based Security Middleware (UbiCOSM) [3]. Though RBAC model simplifies management and review of access control by defining a set of roles, it assumes a centrally administered trusted computing environment. It is therefore difficult to specify security policy for a dynamic environment by this model. In UCON model, core components comprise subjects, objects and rights with additional components like authorizations, conditions and obligations. Here subjects hold rights on objects satisfying conditions. This model is very promising in ubiquitous environment as it covers both security and privacy. But the delegation of rights and administration issues makes it difficult to apply in an autonomous environment. In case of the component based agents, most are usually based on Java and are often kept simple in security design. As a result, they hardly meet the dynamic security demands caused by remote software agent interacting with other agent/platform [1] [2].

3 Knowledge-Based Approach for Robot Security

Security researchers have classified security threats for a networked system into two broad categories: general threats (e.g., unauthorized access, disclosure and corruption of information, denial of service etc.) and software agent based threats (e.g., masquerading, eavesdropping, alteration etc.) [1], [3]. Here, we have considered the later case where agents forms a complicated security-threat matrix among agent to platform, agent to agent and agent to others. These can cause serious damages to every element of surrounding sensor networks including human users and any other living objects. According to Section 2, it is seen that traditional models are not well suited for preventing new kinds of threats in a decentralized, autonomous and dynamic environment. Therefore, the knowledge-based approach has been considered as an alternate. For example, a fuzzy-logic based intrusion detection system has been successfully implemented by Botha and Solms [6]. Along with fuzzy-logic, Susan and Rayford have also used neural network and genetic algorithm for anomaly and misuse detection [7].

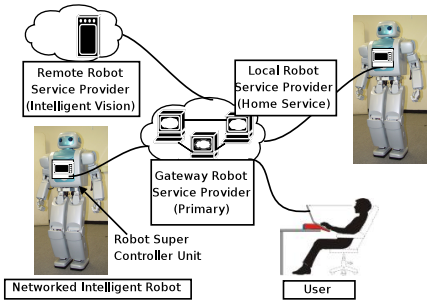

Fig. 1. Main elements of security-aware networked robot control model

However, those works are limited in network intrusion detection and trend-analysis after the actual attack has been performed. Since, they have no run-time protection mechanism, it is not possible to apply them in dynamic decision-making and protection of robot against agent based threats. So we have proposed an active knowledge-based system which is not only capable of detecting run-time security threats but also can infer a decision from its past experiences in an unknown situation.

In this paper, we have mainly demonstrated how our system can sense and react against the agent based threats. For example, suppose we have deployed a third-party vision agent which is responsible for providing intelligent motion-hints to the robot by observing an unknown environment through robot cameras. And we also have a specific network security agreement with the agent operating company. If the agent gets camera image as input (usually UDP streams) and provides motion-hints (TCP packets) as output, the network traffic pattern, CPU use, connection time etc. can easily be defined. If we use these settings as our sample security-policies, we can definitely build our knowledge-based system to sense and react on the violation of these policies. Furthermore, if the system can remember past histories of policy violation/reaction sequences, it can also infer new system reaction in an unknown violation case.

Our knowledge-based approach is highly adaptable on dynamic environments as it can decide autonomously using its security-senses. Consequently, there is no huge administrative overhead if the system is properly trained and tuned. Moreover, the system can also prove its robustness and efficient decision-making capabilities using its past experiences from knowledge-base.

4 Security-Aware Robot Control Model

In this section, we have described our knowledge-based security model, its architecture and mechanisms for policy and knowledge base generation and reuse.

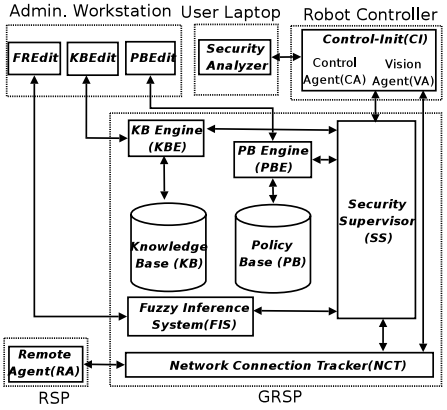

Fig. 2. Knowledge-based security architecture for a networked robot

4.1 Model Overview

The main role-players of our model, as shown in Fig. 1, are the *Networked Intelligent Robot*, *User* (a privileged user is termed as *superuser*), *Gateway to Robot Service Provider (GRSP)*, *Robot Service Provider (RSP)*. Here, GRSP have a central role for providing access to the services offered by the robot. RSPs are the network servers which provide various services to the users and robots via the GRSP. This model has two major distinct features.

1) Gateway Protected Security: Our model clearly introduces a new robot control architecture based on a mandatory gateway-layer for ensuring the maximum security and reliability. We choose this approach for two main reasons: firstly, a gateway-based security management is a well-known approach used by almost every secure network and secondly, the system resources (e.g. battery power) of robot are too expensive to utilize it in security handling..

2) Knowledge-Based Architecture: In our model, another important enhancement is the provision for knowledge development and dynamic security management using the acquired knowledge. The following subsections describe our knowledge-based architecture, policy and knowledge-development processes with necessary examples.

4.2 Knowledge-Based Security Architecture

The security system architecture of our model, as shown in Fig. 2, consists of three implicit layers: planning and decision, supervisory control and routine job management. The members of the planning and decision-making layer is a strategic layer and is only concerned about the global security policy and knowledge of robot and they help to maintain system security by a high-level application interface to user. For example, Policy-Base (PB) has a front-end server PB Engine (PBE) which can be accessed by *Policy-Base Editor (PBEdit)*. Similarly, *Knowledge-Base Editor (KBEdit)* and*Fuzzy Rule Editor (FREdit)* can

access *Knowledge-Base Engine (KBE)* and *Fuzzy Inference System (FIS)* respectively. The middle layer of the security architecture is responsible for managing the lower level security agents by a main *Security Supervisor (SS)* program which may have different submodules for access control, damage detection, intrusion detection etc. This layer plays a significant role in knowledge development process by utilizing the bottom layer assistants, e.g. *Network Connection Tracker (NCT)*, *Emergency System Tracker (EST)* etc. through a close interaction with the chief executive robot control program, *Control-Init (CI)*.

4.3 Security Policy Implementation and Policy-Base Development

The security policy is mainly a set of compliance guidelines provided by the robot manufacturer, robot service provider (e.g., agent developer and/or operator) and user/owner of the robot. At the system development stage, several domain-experts gather these guidelines, e.g., the regular working patterns of a remote software agent like usage of network bandwidth, port, protocol etc. (by a network engineer). A knowledge engineer will take several interviews with these domain-experts to translate these high level system policies into the appropriate sentences of the Policy-Base (PB). These sentences will be fed to a deductive database engine, PBE, which is responsible for storing/retrieving the policy sentences to/from PB and answering queries from SS. PB is a relational database where raw data is stored. A knowledge-engineer or superuser will interact with the PBE via PBEdit (which is actually a client to the deductive database system).

From the sketch of our running system, as shown in Fig. 2, the PB development process is outlined as follows:

- When NCT initializes (at GRSP), it asks SS for a fresh copy of robot's current network security policies which include the remote agent interaction (port and protocol (TCP/UDP) usage, bandwidth characteristics, CPU use, time of interaction etc).
- SS immediately asks the ConrolInit(CI) for a set of current relevant settings of the robot, e.g., operational mode (6 operational modes are defined: Emergency Stop (ES), Graceful Stop (GS), Maintenance/Safe (MS), Limited Human Interaction (LI), Full Human Interaction (FI)), current motion state (standing/walking/running), human command under execution (if any) and so on.
- SS feeds these settings to the PBE as policy generating constraint (e.g., at MS mode, no remote agent connection can be alive).
- Consequently, PBE tries to find a set of policy which can satisfy these settings. If any prior policy data does not exist, it uses the inference rule-sets for generating a fresh copy of policy data.
- While serving the generated policy data to SS, PBE also saves a local copy to the PB for future use.

Thus PB can be filled with necessary policies for different run-time settings of the robot and they can also be reviewed by the superuser through PBEdit.

4.4 Knowledge-Base Development and Knowledge Reuse

Making a safe decision in an unknown situation is a big challenge for the robot. If any security policy is violated by a remote agent the robot needs to find out the anomalous behavior patterns. Here the Fuzzy Inference System (FIS) becomes handy to decide on a remote agent's suspicious behavior. If FIS reports that the agent's behavior is highly anomalous, SS can notify CI with an appropriate caution alarm (Unsafe/Critical/Emergency). Consequently, CI can take emergency decision to switch to a different mode where remote agent interaction will be limited or ceased. In such a case, SS again needs to trigger an update to NCT about the current security policy. These sequences contribute to create dynamic knowledge which is briefly mentioned below.
- Suppose a Remote Agent (RA) is maintaining a connectivity with the motion control agent CA which is transparent to NCT. NCT periodically collects the connection data and checks against the security policy (specific to that RA). In course of time, if NCT notices policy violation of RA it immediately reports to SS.
- SS then asks CI about its current settings and feeds the violation data to FIS and gets a decision on RA's behavior (degree of anomalousness).
- SS submits all these information to KBE to match a similar past experience (e.g. what was done at past in this kind of situation). If no past experience is found KBE generates a new experience.
- While serving the new experience to SS, KBE also saves a local copy to KB for future use.
Thus KB can be filled with necessary experiences for different run-time settings of the robot and they can also be modified by the superuser through KBEdit.

5 Implementation

The feasibility of our model is primarily verified by a prototype implementation as demonstrated in Fig. 2. For the sake of simplicity, a remote-agent (RA) from a RSP system has connected with two local agents: control-agent (CA) and vision-agent (VA) through the security-controller, GRSP. Both CA and VA have been implemented as the core modules of CI. All connections from remote-agents are transparently passed through (and intercepted by) NCT. SS is the main intelligent agent which actively collaborates with CI, as described in Section 4.3 and 4.4, for overall security implementation. It is also interfaced with the PB and KB through PBE and KBE. PB and KB are implemented by MySQL 4.1 relational database [5] and PBE and KBE are CORAL 1.5.2 deductive database system which supports a rich declarative language, and an interface to C++ [4]. Here, the CORAL database engine has two workspaces: relational database workspace and default workspace. Default workspace serves the CORAL query from SS (and also from PBEdit/KBEdit) and RDB workspace maps CORAL query into SQL query for MySQL database. A superuser can manipulate the PB and KB through the PBEdit and KBEdit respectively. All security related reports are periodically sent to SecurityAnalyzer for superuser's review.

Table 1. A KB-term and *iptables* match gives robot a security-sense

Fuzzy Term	Sets	Possible Feelings (Sense)	*iptables* Match
State	LOW	Someone tries to talk	NEW
	MEDIUM	Talk already started	ESTABLISHED
	HIGH	Talk has expired	INVALID
Defense	LOW	Let them talk	ACCEPT
	MEDIUM	Force to stop talking	REJECT
	HIGH	Force to stop talking	DROP
Logging	HIGH	Keep their details	LOG
	MEDIUM	Mark them	MARK
	LOW	Keep them awaiting	QUEUE

In our work, most of the software modules are developed using C++ and Linux. We have also used Matlab Fuzzy Logic Toolbox for initial development and verification of FIS. A user interface, FREdit, is used to tune the FIS by an expert knowledge engineer. Linux-based open-source security framework Netfilter's [9] packet filtering package *iptables* (version 1.2) module is integrated with our NCT. We have used our humanoid platform *Network Based Humanoid (NBH-1)* [10] with Xenomai real-time OS [11](former fusion branch of Real Time Application Interface RTAI/fusion) interfaced with our robot control program, CI.

6 A Case Study on Netfilter/Iptables

In this section, we have presented a simple case study to demonstrate the reactions of our system under some malicious activity of a remote agent. For this purpose, we have exploited the security-features offered by Netfilter's *iptables* package. From Section 4.2 we have already known that our NCT will report SS if it finds any security policy violation caused by remote agent RA, as outlined in Fig 2. For operating our FIS, we have used the similar techniques discussed in [6]. Three fuzzy sets: low, medium and high are defined to represent the degree of anomalousness of RA's behavior. So when the SS gets the anomalous behavior data of RA from NCT, it feeds them to FIS and gets a decision, i.e., the degree of anomalousness of the RA's behavior. Now we may examine the iptables options and combine them with fuzzy rules to get sample system reaction. Let us consider a simple case where our system has sensed that agent RA tries to talk and it has some anomalousness in its behavior. So KBE will read its rule base like this:

IF *someone tries to talk* AND his *behavior* is *anomalous*,
THEN *defense* is high.

Here, *someone tries to talk* is, in fact, the sense obtained from the iptables *state* match options NEW (iptables has its internal mechanism to identify a new packet by looking SYN types in TCP packet header).This is mapped to KB term state

as `LOW` (some possible robot senses, iptables options, fuzzy terms and sets are listed in Table 1). Similarly in output space, *`defense` is high* corresponds to the iptables target jump option `DROP`. A detail description on various iptables matches and jumps can be found at [8]. Thus, if we correctly map the iptables option to robot sense then it is also possible to define the desired system reaction on various anomalous behaviors of any remote agent.

7 Summary and Outlook

The networked intelligent robots are expected to detect quickly any violation of security policy of the environment as well as to defend the potential security threats according to its preprogrammed knowledge or previously developed security-sense without waiting for any user-interaction. The security defense approach described in this paper is dynamic and more robust comparing with the traditional security models. This behavior is very interesting and highly desired in a decentralized environment like sensor networks.

Although the merits of this knowledge development of robot may not be instantly visible to user, but after a long time, the developed intelligence will play a significant role to save human and all other elements of the environment. The improved behavior patterns of the robots will greatly increase the feelings of reliability and confidence of the users and thus it can create many new possibilities for implementing the networked intelligent robots in a challenging environment.

References

1. Brazier, F.M.T., Jonker, C. M.,Treur, J.: Principles of ←Component-Based Design of Intelligent Agents. `http://www.few.vu.nl/ wai/Papers/DKE02.princ.pdf`
2. Jansen, W., Karygiannis, T,: NIST Special Publication 800-19 Mobile Agent Security. `http://www.mirrors.wiretapped.net/security/info/reference/nist/special-publications/sp-800-19.pdf`
3. Yamada, S., Kamioka, E.: Access Control for Security and Privacy in Ubiquitous Computing Environments. IEICE Trans. Commun., Vol.E88-B, No.3. March 2005.
4. Ramakrishnan, R., Srivastava, D., Sudarshan, S., and Seshadri, P.: The CORAL deductive system. The VLDB Journal. 3 (1994) 161-21.
5. Widenjus, M., Axmark, D.: MySQL Reference Manual. O'Reilly & Associates, Inc. June 2002.
6. Botha, M. and von Solms, R.: Utilizing Fuzzy Logic and Trend Analysis for Effective Intrusion Detection. Computers and Security. Vol 22, No 5, (2003) 423-434.
7. Bridges, S.M. and Vaughn, R.B.: Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection. Proc. 23rd National Information Systems Security Conference (NISSC), October 2000, Baltimore, MD.
8. Andreasson Oskar: Iptables Tutorial 1.2.0. Available Online `http://iptables-tutorial.frozentux.net/iptables-tutorial.html`.
9. `http://www.netfilter.org/`
10. `http://humanoid.kist.re.kr/`
11. `http://www.xenomai.org/`