Enable Knowledge Sharing in Intrusion Detection

Ji Li JLI@MIT.EDU
Dah-Yoh Lim DYLIM@MIT.EDU
MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, Cambridge MA, 02139 USA

1. Introduction

Increasingly frequently and cleverly, intrusions are invading our hosts on the Internet. To address this, we see increasingly creative intrusion detection systems applying distinctive techniques, using different data structures for their underlying data, and reporting different kinds of intrusion status. Some are based on traffic patterns, others are based on signatures; some are centralized, others are distributed. Each of them has its own strengths and weaknesses. Despite the fact that attacks often affect multiple domains or organizations simultaneously, the tools used locally may not support effective collaboration across those boundaries. To date we do not have a way to integrate those techniques and create a comprehensive intrusion detection framework, so that intrusion detection systems widely distributed in the Internet can share information easily and worms be detected quickly. To unify those systems and make intrusion detection more effective in the Internet, we propose a unified, knowledge-based framework inspired by work on the knowledge plane (Clark et al., 2003).

In order to meet the needs of individual clients and allow for an adaptable, composable approach, our framework posits that a user contacts one or more detection engines, based on a set of criteria including goals, levels of trust, etc. Each detection engine, in turn will call on a set of knowledge agents, with various kinds of expertise, ranging from particular detection techniques to traffic monitoring capabilities. This framework provides the ability to share and discover existing and new knowledge between entities.

An important issue is for an entity to only expose as much or as little of its knowledge as it chooses, which leads to a mechanism for trustworthy private knowledge retrieval. In this mechanism, we face three intertwined problems. First, a knowledge agent may want to provide partial access to its possible capabilities. Second, the source of a query may want to disguise its own particular interests from the source of knowledge. Third, the parties want to have reasonable trust in the veracity of their exchange. In order to address the first two, we base our work on prior work on Private Information Retrieval, and for trust, on a protocol for developing a trust model.

2. Related Work

In (Allman et al., 2006) Allman et al. proposed a distributed architecture with cross-organizational information sharing to fight coordinated attackers. Their system consists of "detectives" and "witnesses". The detectives are savvy network monitors equipped with sophisticated intrusion detection techniques, while witnesses widely distributed in the Internet provide simple observations to detectives. Information sharing between detectives and witnesses is through loose private matching. We want to improve on two aspects. First, our framework is more general. Information is generalized and unified as knowledge, and most knowledge comes from the existing IDSs. Second, information sharing is done by secure knowledge sharing.

3. Intrusion Detection Framework

3.1 Overview

The main purpose of our work is to design a general framework so that existing as well as new knowledge can be integrated into it effectively. The key idea is to treat the result of any detection technique or other information as a piece of knowledge to be input to the detection engine. As a general term, knowledge in this framework refers to any useful information to intrusion detection in the Internet, including that about individual objects in the network, and the relationships between objects, etc. Figure 1 demonstrates this basic framework.

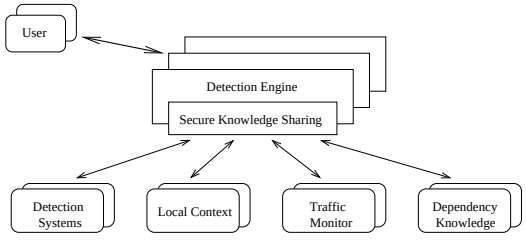

Figure 1. Knowledge-based intrusion detection framework.

There are three parties in this framework: users, detection engines, and knowledge agents. A *user* issues a re-

quest on intrusion detection to a number of *detection engines*. Each engine analyzes the request, coordinates the knowledge sharing between *knowledge agents*, and collects necessary knowledge from them. The knowledge agents provide processed knowledge according to their local policies. In Figure 1, we identify four categories of knowledge agents: existing intrusion detection systems, local context, traffic monitor, and dependency knowledge. After collecting enough knowledge, the engine builds a dependency graph of the agents, and then runs inference algorithms on it and reports the result to the user. All the parties use the same ontology language to describe their requests and capabilities, similar to (Lee, 2007).

3.2 Request Resolution

We demonstrate how the components interact with each other to resolve requests using a simple example. The example request is to detect whether Code Red intruded the network 1.2.3.4/24 in the past seven days, under the scope constraint that the knowledge agents must be within the local ISP. The prior knowledge is that the operating system running on most hosts within the network is Windows. A request is sent by a user to a detection engine. The engine parses the request and does the following.

1. As the request is about a specific worm, the engine checks whether any knowledge agent knows the signature or some properties of Code Red. If not, it has no way to resolve the request, and will return a failure to the user, together with the reason. If the request does not specify any worm, then this step is skipped.

2. If the signature and some traffic pattern are available, the engine collects such knowledge, and chooses a number of knowledge agents based on trust, privacy, and the scope constraint specified in the request. Suppose that at this point in time, two agents happen to be chosen, one using a signature-based technique, and the other using a traffic pattern based technique. Then the engine hands over the knowledge about the worm to the agents, respectively.

3. The two trustful agents analyze some hosts and the recent traffic in the network using their own techniques, respectively, and return the results. Note that the data analyzed may come from a third traffic-monitor agent.

4. After receiving the agents' results, the engine builds a dependency graph of the results, and runs some inference algorithm, for instance, Bayesian inference.

   As a concrete example, suppose the engine employs the following rule to integrate the results from the agents: the engine will report to the user the probability that both results are "No intrusion". Since there is no dependency knowledge about the results (yet), the engine will assume independence between them.

5. At this point in time, another relevant agent happens to join the system, giving dependency knowledge relating the signature and the traffic pattern. In this case the result will be revised to consider the new knowledge, and get a more accurate result.

6. The engine reports a final result to the user.

This example demonstrates how an engine resolves a request with the collaboration of multiple independent knowledge agents, while following the privacy and other constraints. It also shows how a new piece of knowledge helps the engine obtain a better result. Note that a technique itself is just a piece of knowledge in this framework, and this is especially useful when an engine could not let the agent with that technique analyze the data directly due to privacy constraints.

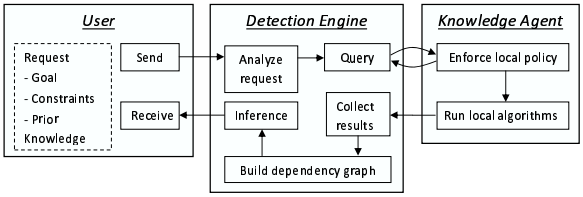

Figure 2. The resolution process of a request.

4. Secure Knowledge Sharing

Different parties may be under different administrations, and would not disclose sensitive information to others. Agents also have different capabilities and credibilities. Therefore, we need a mechanism for secure knowledge sharing that protects sensitive information for both sides (knowledge provider or receiver) and helps establish a trust system to represent capabilities and prevent cheating.

4.1 Private Knowledge Retrieval

To allow for secure information sharing, we propose to use Private Information Retrieval (PIR) and policy enforcers. First, PIR enables detection engines to encode a query in such a way that knowledge agents can answer the query but do not know the query itself. Second, policy enforcers on the agents make sure only proper knowledge will be sent to the engines. Therefore, little sensitive information is revealed to either side.

Private Information Retrieval has been extensively studied in theoretical computer science (Chor et al., 1998). For this work we choose a computationally bounded PIR approach because the alternative requires either complete copies of

the database at both ends or transmission of the complete database, both infeasible.

Each agent needs a policy enforcer that enforces a local policy regarding its own knowledge. The policy enforcer implements the security policy to prevent the exposure of sensitive local information, but to allow for the report of valuable non-sensitive knowledge to the engines.

We will use the private keyword search (Chor et al., 1997) as a black box and demonstrate how to use it in our protocol. For concreteness we focus on a particular example in which database entries are of the form "(IP, Port, Protocol, Traffic, Time)". Say that an engine E wishes to query the agents $KE_1, KE_2, ..., KE_m$. The protocol is as follows.

1. *User Input*. The user provides the engine E with the goal, constraints, and prior knowledge. E contacts each $KE_i, i \in 1, ..., m$ that it thinks it would need knowledge from, to inform them to start the knowledge handshake.

2. *Knowledge Handshake*. Each KE_i checks its local policy regarding information exchange with E, then KE_i computes some function f_i on its database D_i based on the policy, to end up with $f_i(D_i)$. Then KE_i tells E the form of queries allowed.

3. *Knowledge Query*. E sends each KE_i a query that it is interested in, as a function of the user's inputs, conforming to the form that KE_i deemed as valid. Here we use the PIR to obfuscate the queries.

4. *Knowledge Answer*. KE_i computes and sends the results, and E extracts the answer, using the PIR.

Note that in the step *Knowledge Handshake*, if KE_i has no local policy that restricts information exchange with E, then f_i is the identity function, and KE_i would tell E that the valid queries are of the form "(IP, Port, Protocol, Traffic, Time)"; if there is a local policy that only allows aggregate information exchange with E without disclosing the IP address, port number, or protocol type, then KE_i would tell E that the valid queries are of the form "(TotalTraffic, StartTime, EndTime)".

4.2 Dealing with Trust

Privacy is not the only issue we need to deal with. Our system consists of many parties with different interests, and adversaries may intentionally join the system and provide false information. In this section, we demonstrate how to integrate trust into our framework, independently of any underlying trust model.

A user chooses detection engines based on their rankings, an engine accepts or rejects a user's request based on the user's ranking, and similarly knowledge agents accept or reject an engine's request based on the engine's rating.

All the parties rate the others based on their performance. A user rates the engines based on the quality of the returned results using some out-of-band method and sends feedback to the engines. Based on the user's feedback, each engine rates the knowledge agents involved in this process, and differentiates them based on the quality of the knowledge they provided. The engine also forwards the rating information to the agents. The agents rate the engine based on the receiving rating information and other available information such as the request rate. The rating process can be interactive by designing an interactive protocol for two parties to argue about the feedback, either between the user and the engines or between the engine and the agents. All the parties periodically exchange their ratings.

5. Conclusions and Future Work

The strength of this framework comes from its generality and extensibility to support intrusion detection using a wide range of detection techniques and knowledge in a secure way. There are many issues to explore, such as the ontology language to describe the knowledge and agents' capabilities, a practical PIR method, etc. Although it is proposed for intrusion detection, the design is general enough for many other large-scale systems that involve different parties to collaborate and share information.

6. Acknowledgments

We thank Karen Sollins for the help on the initial draft. Ji Li is supported by a research fund from the Intel Corporation, for which we are grateful.

References

Allman, M., Blanton, E., Paxson, V., & Shenker, S. (2006). Fighting coordinated attackers with cross-organizational information sharing. *ACM HotNets Workshop*.

Chor, B., Gilboa, N., & Naor, M. (1997). Private information retrieval by keywords. *TR CS0917, Technion*.

Chor, B., Kushilevitz, E., Goldreich, O., & Sudan, M. (1998). Private information retrieval. *Journal of the ACM, 45*.

Clark, D., Partridge, C., Ramming, J. C., & Wroclawski, J. (2003). A knowledge plane for the internet. *Proceedings of ACM SIGCOMM 2003*. Karlsruhe, Germany.

Lee, G. (2007). *Capri: A common architecture for distributed probabilistic internet fault diagnosis.* Doctoral dissertation, MIT.