A prolog based HIPAA online compliance auditor

Anthony Ho   Sharada Sundaram

March 20, 2008

1 Introduction

With the passing of HIPAA in 1996, multiple health-organizations have become increasingly interested in technological methods of safeguarding health information in order to reduce their legal liability and manage costs. With the primary goal of enforcing medical privacy, the Privacy Rule of HIPAA deals specifically with how health information can be disclosed by covered entities.

2 Background

The idea of modeling HIPAA using Prolog was inspired by Adam Barth during collaborations between Vanderbilt hospital and researchers at Stanford with Prof John Mitchell. Vanderbilt hospital required an easy to maintain system that would enable them to verify whether or not a messages passing through a portal enabling patients to access their health information online was legal under HIPAA. A small protype on this line was also built by Steve Neuyen and Nicole Taheri as CURIS project.

3 Objectives

The purpose of this project was first to determine if we could use such a Prolog representation of HIPAA to discover weaknesses in the law, which allow unauthorized entities to gain access to information they should not be able to access. For instance, we wanted to determine if an insider to a covered entity can obtain unauthorized access to protected health information. Similarly, could an outsider gain unauthorized access?

Our secondary objective was to analyze the feasibility of using Prolog to model laws and its applicability as a HIPAA verifier. One is as a monitor that checks if messages being sent are legal under HIPAA. The second is as a way for us to analyze security properties of HIPAA.

4 Model

Using XSB, an open-source Prolog implementation, we developed a model of HIPAA sections 164.502 and 164.506, and some referenced sections, which describe the circumstances under which disclosures of protected health information is allowed. Our choices when designing the model in Prolog is a result of the two purposes the model will have. One is as a monitor that checks if messages being sent are legal under HIPAA. The second is as a way for us to analyze security properties of HIPAA.

All disclosures of protected health information were represented as a message containing seven fields. Those fields were To, From, About, Type, Purpose, In Reply To, and Consented By. The To and From fields indicate the recipient and sender of the message. The About field identifies who the information contained in the message refers to. The Type field indicates what kind of information is contained in the message. The Purpose field indicates the reason for which the message is being sent. The In Reply To field was added to model a disclosure where the message is sent as a result of being requested. The Consented By field indicates the information that is being sent in the message was allowed to be released by the information's owner.

Firstly we translated parts of HIPAA into its corresponding prolog interpretation making reasonable assumptions and maintaining the structure of the law. A hospital environment was also simulated in prolog to verify the working and regulation of HIPAA. Our Prolog program is run by invoking an overarching rule as message transfer in between the agents of hospital that checks whether a message is legal under HIPAA. The rule works by checking other rules which represented subsections of HIPAA in the program. Each rule is then subsequently checked until the all sub-goals could be terminated at a number of facts.

5 Analysis

In order to analyze HIPAA using our Prolog program, each message would be passed in a query to the overarching rule that checked all of HIPAA. However, our Prolog model is also flexible in that it allows us to check if a message is allowed by only a specific subsection by querying the rule that represents that section directly.

Unlike other analysis tools, Prolog does not have a built-in invariant mechanism that monitors whether or not a property is violated. Thus in order to test security properties of HIPAA using Prolog, potential attacks must be explicitly tested by executing queries for each potential problem.

We developed two types of tests to enable the testing of security properties. The first test contained a separate set of Prolog rules and facts. These test rules each individually test a particular clause of the HIPAA law implemented in our Prolog model. This test helped us verify that the implementation of HIPAA in

Prolog is correct as we interpret it. The second test was a script which iterated through all combinations of values for each field. We analyzed the results output from the script to determine if anything seemed to be unusual.

6 Results

The test cases are effective for verifying whether each implemented clause is working as expected. We were able to correct problems that were discovered after the results of some of the verification test cases were incongruent with our understanding of the HIPAA rules.

By analyzing the results of our bash script test, we were able to discover weaknesses in HIPAA. In a test run of the script, we discovered that many unexpected messages were being allowed. We found that this was due to the following clause in HIPAA allows the covered entity to disclose information for its own operations.

6.1 Insider gaining access

164.506 Uses and disclosures to carry out treatment, payment, or health care operations. (c) Implementation specifications: Treatment, payment, or health care operations. (1) A covered entity may use or disclose protected health information for its own treatment, payment, or health care operations.

HIPAA only regulates the use of protected health information by a covered entity and does not specify what should become of the information once it leaves the entity. If the recipient is not a covered entity, he or she could potentially disclose protected health information. For instance, what happens if a contractor working on renovations in the vicinity of protected health information disclosure happens to overhear some of the details of the information? He does not seem to be legally prevented by HIPAA to disclose the information later.

Furthermore, it does not seem that HIPAA has specified any measure to prevent a doctor who obtains protected health information because he needs to treat patients from revealing that information later once he is no longer part of the covered entity. If the doctor quits his job or retires, he may no longer be obligated to maintain the confidentiality of the information he has obtained.

6.2 Outsider gaining access

Another flaw we discovered is that there are some rules that allow a covered entity to disclose protected health information to outside individuals. For example, consider the following clause:

164.502 Uses and disclosures of protected health information: general rules. (a) Standard. A covered entity may not use or disclose protected health information, except as permitted or required by this subpart or by subpart C of part

160 of this subchapter. (2) Required disclosures. A covered entity is required to disclose protected health information: (ii) When required by the Secretary under subpart C of part 160 of this subchapter to investigate or determine the covered entity's compliance with this subpart.

In this case, once the Secretary receives the protected health information, can the Secretary then disclose the protected health information? There is nothing we could find in HIPAA that prevents the Secretary from then disclosing the information. However, it is possible that other laws can prevent this from happening.

Since HIPAA does not explicitly regulate these disclosures, we cannot be sure the disclosures outlined above are legal or not. This problem can be further looked into by those with legal backgrounds. However, since the primary purpose of the Privacy law is to protect health information, we believe these two weaknesses to be a violation of the goals of the Privacy Rule.

7 Conclusions

After working with Prolog for the past several weeks, we have come up with some opinions on the utility of Prolog as a security analysis tool and a law verifier. It is important to make a distinction between the two uses. A security analysis tool should contain capabilities that enhance the analyst's ability to discover new attacks, while a law verifier should be easy to use and contain conventions that facilitate the translation of the law to the language of the verifier. Prolog exhibits both positive and negative characteristics when judged in consideration with those two slightly different requirement sets. As a security analysis tool, Prolog is useful in that it allows us to quickly check using a query whether any of our pre-conceived attacks work. A security analyst with a Prolog model of the rules governing the protocol can potentially execute queries that probe the vulnerabilities of the protocol. XSB also contains a debugger for Prolog that allows the user to describe why a given query is satisfied or how it failed. However, it would be useful if a visualization of the satisfied goals and facts could be automatically generated rather than having to walk through the debugger a step by step to determine the reason a query passes.

The main drawback in using Prolog for security analysis is that a Prolog program does not seem to be able to compute potential security weaknesses on its own, at least in the way we have decided to model the law. All security weaknesses must be thought of in advance and only then can it be quickly verified by a query. For example, in order to find an attack, we must set up the facts and queries in such a way that we believe may yield an unexpected acceptance of a message. If we did not setup the environment correctly, the attack will not be discovered. Finite-state model checkers such as Murphi seem to be better suited to finding unknown and unexpected weaknesses, but the problem with modeling HIPAA with Murphi is that the law will be difficult to implement in Murphi.

As a law verifier, Prolog exhibits many positives traits. For instance, the logic in legal clauses often maps directly to the logic of Prolog rules. In addition, these Prolog rules are subsequently easy to understand. However, one of the main problems with using a language to model law is that law is open to interpretation. Small differences in the translation of the law can yield significant differences in the behavior of the model. Fortunately, this problem is mitigated by the fact that Prolog is easy to read and therefore easily validated by auditors.

Another potential problem with using Prolog as a message legality verifier is that it may be prone to DOS attacks. This is because when the Prolog verifier fails a query, it must search all subgoals to see if any subgoal can potentially allow the query to pass. If the HIPAA law were implemented in its entirety, this could take a significant amount of time. However, this problem only applies if the Prolog program is used to monitor messages.

Although there are problems with using Prolog as a security analysis tool and law verifier, there are also benefits to using it. As a security analysis tool, the fact that Prolog cannot search for potential attacks is a significant limitation. Unfortunately, it is one of the few choices available for modeling laws. Given the size of most legal documents, analyzing laws using other tools would be require an extraordinary amount of work. Therefore, we believe that the positives of Prolog such as the readability and efficient translation from law to rules warrant further investigation to see if Prolog can be better used to model law.