Requirements and Design for an Extensible Toolkit for Analyzing EMR Audit Logs

Eric Duffy, Steve Nyemba, Carl A. Gunter, David Liebovitz, and Bradley Malin*

April 2013

Abstract

Over the past decade, various regulations have been proposed and promulgated to support the auditing of accesses to Electronic Medical Record (EMRs). Current tools to support this process can improve their use of statistical and machine learning techniques and auditor interfaces. We sketch requirements and design for an *Extensible Medical Open Audit Toolkit (EMOAT)* to enable progress in these areas. A key objective is to provide interfaces that support three types of stakeholders: (1) expert analysts, (2) privacy and security officers, and (3) patients. Our system design provides for an application programming interface that enables officers and patients to access both simple and complex analytic systems. We illustrate how EMOAT has been adapted to support certain audit functionalities with data from the EMR systems of several large hospital systems.

1 Introduction

Complex workflows and the high risk of denying access to records make it impossible to define strict least privileges for access to patient records in electronic medical record (EMR) systems used by Healthcare organizations (HCOs). HCO privacy and security officers therefore tend to rely on employee training and after-the-fact reviews of EMR audit logs to inspire accountability in employees and mitigate risks of continuing abuses. These logs contain many entries (on the order of millions per day), so technologies and commercial products (e.g., FairWarning [12] and Veriphyr [29]) have emerged to automate the review process. In current HCO practices, reviews mainly rely on simple rules and manual inspections based on well-known types of abuse, such as the accessing of "very important persons" (VIPs) records without legitimate cause. However, recent research has explored ways to extend these techniques with new types of rules, such as explanations (e.g., [11]), that reduce manual review requirements and machine learning approaches that identify unusual patterns worthy of further inquiry (e.g., [3, 5, 19]). At the same time, there is an increasing push to adopt legislative rules that involve patients in the review of their records by expanding their current privileges to know who has accessed their EMR record. However, there are several major challenges associated with auditing in the EMR domain. First, the emerging class techniques are not yet integrated into practice due to a variety of impediments (See Section 2). Second, HCOs are often skeptical about showing audit log data directly to patients because it may be misunderstood and, frankly, because HCOs often do not understand these logs very well themselves.

The aim of this paper is to articulate a core set of the requirements for pushing this new tranche of advanced analytics and active patient reviews into practice and sketch design strategies that aid their implementation. We argue for the need of an extensible toolkit that is aimed at the analysis of EMR audit logs. We further claim that such a system should have, at a minimum, three interfaces to support its primary users. The first of these is an *analyst interface*, which enables researchers and vendors with new ideas for analysis to introduce and tune their strategies. The second is an *officer interface*, which is used by HCO personnel, such as the privacy official. This interface should offer access to assorted analytic tools without burdening the users with too much detail about how they work and how they are calibrated. Finally, the third is a *patient interface*, which provides the patient with access to their EMR data, as well as information about how HCO workflows have shared and utilized their record. We show how these last two interfaces can be supported by an application programming interface (API) that allows for the combination of information from underlying analytic modules in a readable form. There are two key issues to be understood about the three inter-

*Affiliations: Eric Duffy and Carl A. Gunter, University of Illinois; Steve Nyemba and Bradley Malin, Vanderbilt University; David Liebovitz, Northwestern University

faces: (1) their requirements both for their users and how they relate to each other and (2) their design as a way to add new functions over time and interface with essential systems like the EMR.

To support the realization of such interfaces, we introduce the *Extensible Medical Open Audit Toolkit (EMOAT)*, a prototype system of assorted tools that we have been developing and applying over the last few years. Various modules in the system have been validated on the audit logs of Vanderbilt University Medical Center (VUMC) and Northwestern Memorial Hospital (NMH). We refer to our overall methodology as Experience-Based Access Management (EBAM) [15], which comprises the general idea of using experience (especially from access logs) to tune access controls toward least privilege using a continuous process improvement model. While we believe the idea has general applicability independent of this sector, our focus has been on HCO audit logs and access by HCO personnel given the current regulatory requiremtns. Examples of work with EBAM and EMOAT include anomaly detection for EMR users and accesses using techniques like social networks models [6, 7] and algorithms from operations optimization [30], exploring ways to predict the roles of users based on their access histories [32] and seeking ways to adjust roles so that they better match user behavior [31]. We have also explored ideas for modeling and detecting access patterns such as open terminals and masquerading with techniques from machine learning [16], explaining logs in health information exchanges (HIEs), and as well as other completed and ongoing studies.

The remainder of this paper is organized as follows. Section 2 provides general background and related work, including a taste of the relevant legislation, standards, products, and emerging technologies. Section 3 describes some of the high-level requirements for the system including the needs for the backend system and the frontend system, whose designs are discussed in Sections 5 and 4 respectively. In Section 6 we illustrate how the system might work with examples. We conclude with a discussion of issues and tradeoffs in Section 7.

2 Background and Related Work

Audit reviews of access logs encourage accountability in system users and can catch problems before they escalate to massive breaches. Reviews by HCOs are primarily motivated by three driving forces. First, doing the right thing: ethical considerations and security best practices encourage adherence to the Hippocratic Oath and least privilege in responsible organizations. Second, there are recognized business costs entailed in losing a patient because of a privacy concern [4]. Third, there are a number of legal regulations, such as the U.S. Health Information Portability and Accountability Act (HIPAA) and the Health Information Technology for Economic and Clinical Health Act (HITECH), which impose specific privacy rules for patients and financial penalties for violating them. Because of the concreteness of this last driver and its demonstrated impact on publicity and the bottom line, many of the technologies and practices for audit are built around avoiding these penalties, ideally by deterrence.

Auditing as a Process Based on regulatory requirements, the auditing in the EMR domain should consist of three stages. In the first stage, audits should be directed at specific types of violations with recognized patterns. This level of audit has demonstrated its value in catching violators who illegitimately access records of celebrities [2, 24, 28] or those of friends and fellow employees [22, 18]. In the second stage, audits should leverage statistical information to detect anomalies, even when they may not arise from previously-recognized specific patterns. There is some progress in this area in practice, at least with the use of basic summary statistics like viewing the users who accessed the most records in a given period [13]. In the third stage, information from audit logs should be monitored in real time to detect and possibly prevent violations. We are not aware of any widespread use of such real time techniques at the current time.

Our focus in this work is what it will take to enable the second stage technology. Current vendor offerings such as those of FairWarning [12], P2Sentinel [25], and Veriphyr [29] provide a foundation to build upon, by collecting audit logs into SQL databases, where they can be put in normalized schemas and analyzed efficiently. However, current capabilities are limited to simple types of summary statistics. In order to truly move to the next level and begin to use advanced machine learning techniques, a more generalizable and flexible architecture is required. An example of the type of technology we have in mind is illustrated by algorithms that can form a social network out of a hospital log and use this social network to search for unusual users or accesses using a technique like k-nearest neighbors (k-NN) [5, 8]. This requires more than a fancy SQL query.

Accountability and Regulatory Requirement There is another level of accomplishment orthogonal to the three stages, which is based on a new round of legislative requirements that aim to increase the accountability impact of audit logs by sharing them with patients. Proposed modifications to Section 164.528 of the HIPAA Privacy Rule [9] would extend the existing right for patients to see an accounting of disclosures of their health data from a HIPAA covered entity to a third party. This extension would give patients the additional right to see the access logs for their records in a designated record set, which might include access for purposes of treatment, payment, and operations. This would allow patients to learn if specific users (i.e., HCO employees) have accessed their record.

This is considered a notable change, since the accounting of disclosures rule applied to a comparatively narrow collection of circumstances, such as giving information about a patient to law enforcement, whereas the new rule would provide information about a wide range of user accesses—even within the HCO and for routine purposes. There has been much debate in response to the proposed rule change about how feasible or desirable it is to make such an extension [23]. We have looked into some of the issues with this change from a technology and patient information perspective and consider how it might be accomplished as part of the EMOAT architecture. Our illustrative application is to show how the American Medical Society (AMS) Health Care Provider Taxonomy Code Set [21], which provides a classification for areas of specialization for employees of health care providers, could be used to help patients understand their access record.

Standards and Integration A major impediment to the development of advanced analytic techniques for HCO audits is the need to collect information from a large number of different systems and handle it in a standardized manner, so that techniques developed for one HCO can possibly be used in another, thus offering an economy of scale. Large HCOs manage patient information in as many as 100 or more systems [27]. We believe there is hope for improving the current situation by looking to efforts to address a still bigger one, namely the audit standards for Health Information Exchange (HIE) between HCOs. There are emerging standards in this area that could also help with audit standardization even within HCOs. The Digital Imaging and Communications in Medicine (DICOM) Standards Committee has advanced a draft standard [10]. Internet Engineering Task Force (IETF) Request for Comments (RFC) 3881 [20] consolidates the DICOM recommendation with other efforts in this area by providing a broad range of audit message formats and contents. RFC 3881 uses an XML schema that covers access events as well as other types of events such as those involved in security administration. This approach has been adopted by Integrating the Health Environment (IHE) for the purpose of auditing health information exchanges (HIEs) between HCOs, a problem which exhibits a certain degree of architectural similarity to auditing multiple systems within a single enterprise.

The IHE, through the Audit Trail and Node Authentication (ATNA) profile [17], recommends the use of the RFC-3881 format. Also in response to the need for auditing HIEs, the Nationwide Health Information Network (at first abbreviated as NHIN but later changed to NwHIN) specifies XML formats for audit log requests [1]. Furthermore, Health Level 7 (HL7) addresses a range of patient privacy audit scenarios in its Privacy, Access and Security Services (PASS) audit effort [26], which aims to situate DICOM, ATNA, and other audit standards with standards set by HL7. Unfortunately, there has been mixed success in transitioning the above-mentioned standards into practice. Few vendors support the IHE profiles [14] and most efforts have focused only on consolidating data into the audit logs. Yet, there is a keen need for meaningful analysis techniques to enable the audit logging to benefit its true purpose.

3 Requirements

In this section, we outline the design requirements for an extensible toolkit for analyzing EMR audit logs, which we will simply call *EMOAT requirements* below. These requirements are based on how auditing systems should enable the guidelines set forth in regulatory requirements (e.g., HIPAA and HITECH). Specifically, our requirements are oriented to address several types of stakeholders. As discussed earlier, we divide these into three general categories, namely patients, privacy and security officers, and analysts. The latter group is heterogeneous and may consist of EMR administrators, informatics researchers, and software developers.

Modern HCOs are distributed environments, as are the systems they utilize to conduct their operations. As such, EMOAT must be able to exhibit the properties of a distributed system: responsiveness, accessibility and integration supported by sound engineering principles. The *performance* of each individual module must have the appearance of real-time re-

sponsiveness to the end user, so that when she logs into the system all analysis on her records is instantaneously available. However, audit logs in an HCO belongs to the big data problem, where the volume of the data is a limiting factor to timely results. We have addressed this issue by defining an elaborate module lifecycle: a pre-processing stage, periodic updates, and the segmentation of computation. Since an HCO is a distributed system, every stakeholder must be have seamless *accessibility* access relevant resources provided by EMOAT from their wherever they may be in the HCO. HCOs are characterized by a wide variety of systems with specialized functions, so EMOAT must be able to *integrate* existing systems in order to deliver value.

Both the patient and the privacy officer often start from a view of the way in which a patient's record was accessed. Since officers are, at times, reacting to patient complaints, it is clear they should have this capability. However, beyond the patient view the officer should have the ability to look at the access history of a given user, perhaps because an access by that user seems unusual and needs to be considered in the context of other user accesses. Ideally, it will be possible to provide services through an API for both the patients and officers. Patients will need modules that help them understand why people accessed their records, such as the roles they played in terms that the patient will understand. Officers, by contrast, must be afforded with a broader view to gather information for (or against) an investigation of a policy violation.

There are many types of analysts with diverse goals, so it is too much to expect that a common API can be adequate for all of them. For instance, an EMR administrator may be interested in usability issues, such as discovering what causes certain types of users to make mistakes when they enter orders. A researcher may be interested in modeling hospital workflows to discover better procedures. A developer may be seeking performance figures to aid planning for new equipment and software. Also, EMOAT must be able to handle the heterogeneity of systems in an HCO and, in particular, the heterogeneity of the structure of logs. And, since auditing requirements and regulations change over time, EMOAT must allow for extensions to incorporate new methods. Auditing methods can be inter-dependent for a broader analysis, so we need a framework for building models that supports chaining modules, segmenting computations and provisioning seamless access to the HCO data that is authorized for auditing.

The requirements can be summarized in a system that implements advanced auditing in a cohesive suite while leveraging existing technologies. Technically the challenge is finding ways to implement a cohesive suite with components that are not always designed to interact with one another. Such a task would imply implementing integration patterns, architectural styles as well as protocols for the various components to communicate. In addition advanced auditing techniques require capabilities found in specialized tools (R; MATlab; Octave) that would have to be accounted for.

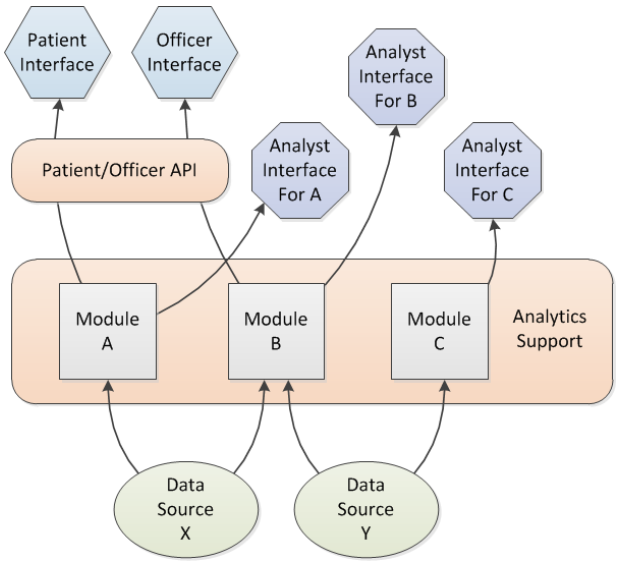

Figure 1: EMOAT Architecture

4 Back-end Design

A high-level architectural diagram for EMOAT is given in Figure 1. The interfaces at the top of the figure are the human interfaces to the system, which we have divided into patient, officer, and analysts. This is the EMOAT *front-end*. At the bottom of the figure are interfaces between EMOAT and its data and analytic capabilities, which we collectively call the EMOAT *back-end*. We consider each of these in turn, starting with the design of the back-end.

The key features of the back-end are its data store abstractions, the patient/officer API, and its analytic modules. The patient and officer user interfaces interact with EMOAT through the Patient/Officer API to enforce a restriction of the data presentation in an easily understandable format. The analyst interfaces require more detailed data and are, therefore, not restricted by the Patient/Officer API. Each module has access to one or more data sources, which can be heterogeneous in nature, but uniform in their presen-

tation to the modules.

Data Store Abstraction To standardize data access and integration as well, implementing accountability, and accounting for the demands of auditing (re-using results of methods) a high order of data abstraction is required. The nature of an HCO is characterized by the utilization of a wide variety of systems in order to conduct operations. This heterogeneity shows in the structure of access logs, as well as the various systems that store the logs. To address the needs of scalability and cohesion, EMOAT abstracts database access, data representation, and computation engine access. EMOAT is capable of connecting the semantics of an auditing model to that of an arbitrary relational database through a mapping mechanism framework, where an auditing module's inputs are mapped to a data source attribute. This capability addresses the diversity of databases within an HCO.

The abstraction of data representation enables scalability and integration in various architectural styles and protocols. Javascript Object Notation (JSON) is an implementation of abstract data representation that does not require serialization and is widely recommended as an industry standard in implementing distributed architectures. We address standardization by abstracting data stores, with JSON support it is possible to integrate data stores from various vendors (Postgresql, Oracle, Sql Server, Mysql) seamlessly. EMOAT abstracts data representation and computation engines (R, Matlab, Octave) providing the capability to implement supervised and unsupervised auditing methods and condition their output for re-use.

These capabilities leverage existing technologies, paradigms available within a given HCO or off the shelf. Auditing modules therefore inherently have these capabilities: leveraging abstract data stores, leverage computation engines and handle abstract data representation (JSON) .

Patient/Officer API The core of the Patient/Officer API is implemented as a series of Java classes and libraries running inside an Apache Tomcat Java container. All of the code for receiving and responding to HTTP requests, issuing queries, and launching modules is implemented in this component, referred to as the *middleware*. The middleware maintains a collection of the known modules and their configuration properties known as the *Module Registry* (MR). Module information is provided to the MR as a well-formatted extensible markup language (XML) file that provides the module's name, Java class location, and a series of parameters that control the module lifecycle. This lifecycle is divided into four phases: 1) registration, 2) preprocessing, 3) runtime queries, and 4) uninstallation. When a module is first registered, the Module Registry organizes all of the parameters required in the subsequent phases of the module's lifecycle. During preprocessing, the module is supplied with a database instance and notified to begin preprocessing. When the module declares that preprocessing is complete, the module buttons become visible to the front-end user and the module enters the run-time query (RTQ) phase.

Each time a user queries for a patient on the front-end, the module is given the type of data it declares as a parameter in the data given when it was initalized. The purpose of this data is to yield more efficient queries and computations which assure a more responsive user interface. Afterwards, the middleware supplies the module with the parameters it requests in the element, such as a patient identifier, user or role identifier, or patient diagnosis information. When a module is ready to be removed or refreshed, the uninstallation phase is called to clean up any persistent data. When a module capability is declared in the registration XML file, the accompanying Java interfaces must be implemented within the Java class for that module so that the middleware can call the appropriate methods. There is a unique interface for each of the data types given as parameters in the registration XML, as well as for the return types. For instance, if a module requires a patient identifier during either the initialization metadata or the runtime query metadata, it must implement the NeedsPatientId interface. This interface defines a method for receiving the identifier for a patient in the underlying data store, which would typically be a primary key to a patient table.

There are many strategies one might pursue for how to translate between the conclusions of an analytic module its graphical representation as we explain in the next section. In the current implementation, if a module is designed to color elements of the access log, it implements the ReturnsDouble interface, which returns a value between 0 and 1 representing the anomaly level of the element in question. A value of 0 denotes an assertion that the element is deemed not anomalous, a value of 1 means that the element is considered anomalous, and 0.5 means there is no assertion at all. Modules can also return a character string which provides a description of the element to be given to the user. The coloring and descriptions are applied to the elements specified as runtime query parameters.

Modules We describe three modules developed for use in the Patient/Officer View. These will be used to illustrate how the front-end can use analytic modules via a graphical interface.

Orders. The Orders module inspects the current state of the access log and searches for accesses for that appended a note to the patient's medical record or an order (e.g., medication start/stop, consult request, etc.) was issued. The core reasoning for this module is that if an EMR user, such as a nurse, examines a patient's record, they are unlikely to make a note or an order if the access is illicit in nature. Conceptually, this is similar to a simple form of the "explanation" proposed in [11]. When a user makes such a modification, the notes and orders are typically reviewed by at least one other member of the staff, therefore bringing attention to the access. Orders module requires no preprocessing or uninstallation time, but does need a database connection during the RTQ phase. In a strict sense, Notes and Orders only requires an access identifier as input, but it can be supplied with the patient identifier to enable more efficient data extraction from the underlying data source.

Position Explainer. To provide both patients and administrators with some intuition into why a user accesses a patient's medical record, we have developed the Position Explainer module. This module works by calculating the likelihood that a user will access a patient's record based on the patient's diagnosis information. Each patient has zero or more billing diagnoses (e.g., International Classification of Diseases, version 9) associated with them and it is anticipated that patients with similar diagnoses will follow similar workflows. For example, when a patient is admitted to a hospital with a broken leg, she should follow a routine series of tasks, such as check-in at reception, a physical examination, an x-ray, a cast placed on the leg, and finally a discharge from the hospital. It is unlikely (though not impossible) that a patient with a broken leg will need an abdominal ultrasound or a psychiatric evaluation, such as are common for expecting mothers and mental health patients, respectively. This principle suggests that many diagnoses have a high likelihood of being accessed by a particular user if that user has one of the common roles associated with the diagnosis. The Position Explainer module requests a patient identifier and user identifier as input and returns a description if that user's position has a sufficiently high likelihood of accessing the patient based on his or her diagnoses. For each diagnosis belonging to the patient, the module determines the likelihood a role accessing a patient with that diagnosis and repeats this for each role belonging to a user in the patient's access log. If the likelihood is higher than a tunable threshold, the user is given a message stating that the user's position is highly associated with his or her diagnosis. This module can also be leveraged for coloring users who have a very low probability of accessing the patient.

CADS. The community-based anomaly detection system (CADS) was developed to detect anomalous actors by leveraging the collaborative nature of healthcare workers [6]. It works by translating the user-patient relationships in the access log into a social network that connects users who access the same patient record during a certain timeframe. Once the social network has been constructed, CADS performs a principal component analysis on the network to extract the core communities within the network. Upon this decomposition, each user is compared to their nearest neighbors to assess how far they deviate with respect to their community of coworkers. CADS requires considerably preprocessing time compared with the other two modules as it performs a number of complex calculations on a large data set. CADS asks for a patient identifier during its initialization phase so that it can quickly return its results without needing to repeatedly look up results for each user when only a small number of users are anomalous. It returns both a double as an anomaly score as well as a description yielding information about why the user was detected as anomalous.

5 Front-End Design

The EMOAT user interface is implemented as a set of Hypertext Markup Language (HTML) pages which are dynamically generated by the middleware before being sent to the user for viewing. The user's browser also makes asynchronous calls via JavaScript to the middleware where Java servlets reply to the requests. We describe the three classes of views below. The Patient View is primarily a restricted version of the Administrator View, hiding information which is sensitive in nature, such as users or accesses which are detected as "anomalous" by the modules.

Officer View The Officer View initially consists of a page in which the user inputs a patient ID into a text field for querying. When the patient identifier is submitted to the form, the server responds with a new page consisting of a complete list of audit logs for that patient. This is divided into possibly several HTML tables, one for each of the patient's encounters. The user is then presented with buttons for each of the active modules currently available. If a module

has encountered an error or its preprocessing stage is incomplete, the button will not be shown.

When the user clicks a module button, an asynchronous call is made via JavaScript to the middleware by an HTTP GET request issued to the appropriate servlet running on the Tomcat server. There are two servlets currently utilized in EMOAT: one for handling coloring based on the anomaly scores returned from a module, and another for handling textual descriptions yielded by a module.

The HTTP GET request issued by the client-side JavaScript is sent a URL specifying the servlet to call with a module identifier and patient identifier as query parameters. The module identifier is a universally unique identifier (UUID) generated when the module is registered.

The servlet then calls the appropriate module and returns XML to the client containing the coloring or descriptions information. The user's browser then executes its JavaScript to parse the XML and either adds coloring to the access log or generates a icons notifying the user that there is a description available. The user can then hover over the icon to read the description.

Patient View The Patient View for the front-end is, currently, a restriction of the Administrator View. It is necessary to limit the amount of information presented to the patient, especially when there are potential HIPAA violations being considered. The Patient View is implemented using the same technology as the Officer View, but the set of modules exposed to the user are restricted. Currently, only the Position Explainer module is available to the patient because it only provides information explaining the roles which are present in the access log. The other two modules provide more detailed anomaly detection information and are therefore not provided to the patient.

Analyst Views EMOAT provides a specialized view for data analysts of various auditing modules. This view is effectively a configurable dashboard of various auditing modules which afford the analyst a broader view of a given auditing viewpoint. For instance, EMOAT has an interactive network analysis auditing module that computes support and confidence scores (i.e., joint and conditional probabilities) for the association roles between HCO departments and users of an HCO. The interactive network of scores (which, in this case, correspond to probabilities) is packaged as a searchable interactive dashboard with several views and details of either departments, users and the associated networks.

Code	Description
153.9	Malignant neoplasm of colon, unspecified site
197.0	Secondary malignant neoplasm of lung
197.7	Secondary malignant neoplasm of liver

Table 1: Summary of the diagnoses for patient 35347

The Analyst View does not require the Patient/Officer API because it does not need to reduce the analytic results to simpler formats like colored accesses. As such, the Analyst View directly connects to the underlying modules. Data representation in the analyst view abstracts access to various computation engines like R, Matlab and Octave using a pipe and filter architectural style and formatting their output into JavaScript Object Notation (JSON). The use of JSON gives the data high usability across layers, portable across layers, and integration within components. This enriches the capabilities of EMOAT and its ability to integrate components like rendering engines such as Highcharts, The JIT and gRaphaël which use JSON.

An example analyst view can be seen in Figure 2 of a social network auditing dashboard which displays support and confidence scores for departmental interactions.

6 Illustration

This section presents an example scenario in which EMOAT is utilized by both a patient and an administrator. We aim to demonstrate the power of EMOAT in satisfying the requirements of both classes of users alike and provide a taste of what can be accomplished after as additional modules are developed and integrated into the EMOAT system.

Patient View Imagine that you were recently a patient in a hospital who has undergone treatment for issues related to your internal organs. You were provided the patient identifier of 35347 and diagnoses in Table 1 during your five encounters at the hospital.

You remember there were many different employees of the hospital coming in and out of your room. You are curious who all these people were and why they required access to your sensitive personal information. You know you have the right to an access report, so you file a request with the provider, who is obligated by law to supply it to you.

The provider responds telling you to log into their EMOAT web page using your patient identifier. You log on to the page and are greeted with the welcome

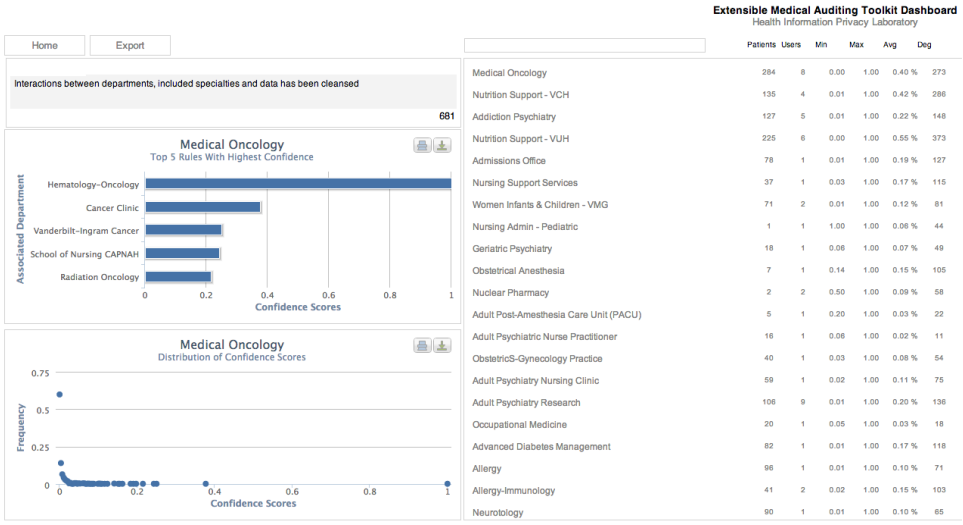

Figure 2: Sample EMOAT Analyst View

screen.

You click on patient view and enter your patient identifier in the search field, which produces a list of your accesses. Since find you were accessed by 319 distinct users over the course of 5 encounters. As a result, the access record is quite long and contains a large number of unfamiliar user positions, such as "UR/QA 1".

According to the requirements of the U.S. Department of Health and Human Services, your record displays the date and time of access, the name of the users who made the access, and the user's role which is used in lieu of a reason or action taken on the data.

Additionally there is a button reading "Explain Positions" at the top of the page. When you press it, each user has a small question mark icon next to their name. When you hover over the icon for the user with position UR/QA 1, a tool-tip text appears reading,

> This role has a 80.0% chance of accessing your diagnosis, 197.0
>
> This role has a 82.0% chance of accessing your diagnosis, 153.9
>
> This role has a 93.8% chance of accessing your diagnosis, 197.7

You begin to feel more comfortable about seeing strange positions that you previously were unaware of or unfamiliar with. However, since you had so many accesses to your record, you decide to call your healthcare provider and request they audit your record for any anomalous behavior.

Administrator View The security administrator at your provider receives your request for an audit and proceeds to check your record through the EMOAT system. When she logs in with your `patient_id`, she sees the same access log that you did when you logged in. However, she has several modules available to her that you did not have access to. In this case, these are the Notes and Orders module and the CADS module.

The administrator begins examining the record by selecting the CADS module, which colors a few users in various shades of red and places a question mark icon next to the names of those users. The administrator hovers her mouse over the icon for user 6679, who is shaded in a faint red. The tool-tip text appears, displaying

> User 6679 is possibly anomalous within his or her social network.

which indicates that CADS has deemed user 6679 suspicious based on its configuration parameters. The administrator decides that there is no need for alarm yet, since the CADS module was configured to begin shading users red when their deviations are outside of two standard deviations from the mean. She continues searching for users who have a higher alert level.

Before long, she discovers user 6736, who has a significantly deeper red color than the previous user. She begins to get suspicious of this user, so she enables the Notes and Orders module, which colors a large number of the accesses green, indicating that they are not likely to be suspicious.

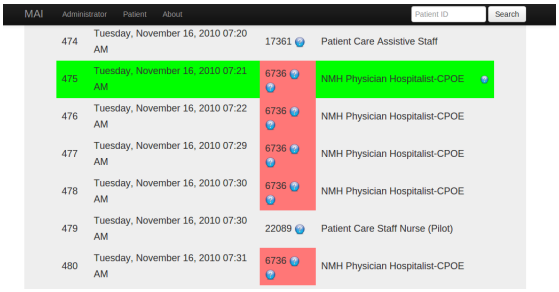

Figure 3: A combination of the CADS, Notes and Orders, and Position Explainer modules

User 6736 also has a number of his accesses turn green at that point, but the red color enabled by CADS remains. She also enables the Position Explainer module which gives statistics about the user-diagnosis relationship, as in Figure 3. The Position Explainer module says that the greatest chance of user 6736's role accessing a patient with this diagnosis is 75%. This means that users with the same role as user 6736 accessed patients having one of the same diagnoses as the patient 3 times out of 4. Since one of the accesses is colored green it is quite possible that the user is doing legitimate work, but he or she also makes several accesses that are not accompanied by notes or orders. The administrator decides that she should probably take a look into the accesses made by user 6736 to other patients to see if any other suspicious behavior sticks out.

Benefits This use case illustrates the following benefits. First, EMOAT automatically compiles an access record ready for presentation to the patient which eliminates time spent generating the data by specialists. Second, the patient is presented with options for explaining the accesses that are made to his or her record. This behavior is likely to prevent patients from becoming confused by, or concerned about, who makes accesses to their record. Third, a significant amount of manpower was saved in replying to a patient's request for an audit by leveraging existing anomaly detection techniques in a user-friendly way. Although providers are not required to respond to patient audit requests, doing so is good for keeping the patient satisfied and ensuring the patient returns to the same provider.

7 Discussion

We have demonstrated that the access logs of EMR systems can be harmonized and integrated into various modules for auditing purposes through an extensible software framework. Moreover, these modules can support interfaces for a range of stakeholders, including patients, administrators, and developers. In doing so, we have shown it is clear that there are certain baseline requirements for the interfaces. For instance, the analyst interface must allow for learning and calibration across as much of the hospital operations and human resources data as possible. It is intended for use by experts so there is a premium on flexibility and scalability to enable detailed reports on sometimes unanticipated attributes. Officers need an interface that demands less expertise, but is tuned to enterprise and regulatory protocols.

However, the establishment of such a software system leads to challenging questions regarding how and when to present information to these stakeholders. In particular, the notion of a patient interface is new territory. It is necessary that such an interface be as easy to understand as possible. Yet, beyond such baseline requirements, the issues are more complex. For instance, should patients be permitted to know how care providers have treated or worked with other patients? Even if such information is presented in aggregate, it begs the question of if the operations of the organization and the specific practices of certain employees should be provided to patients. This is information that reaches beyond the specific information about who accessed their records when and for what purpose.

There are also concerns around policies for other interfaces as well. Consider, should the officer interface be enabled also for EMR users (like doctors and nurses) so they can see their own profiles? There will be pressure to take information that can be seen in the officer view and make it available to patients, but there will be limits to how far it is appropriate to go in this direction.

EMOAT is designed to simplify the development, presentation, and interpretation of access logs in EMR systems. However, the detection of suspicious accesses is only one part of a much broader audit process. Such a process needs to allow administrative officials to open an investigation and systematically proceed until a suspicious event is either confirmed as a policy violation or is dismissed. However, at the present moment, there is no standardized audit process. Thus, while the users of EMOAT (or similar systems) may be able to receive alerts and navigate the system to perform an ad hoc investigation, they will be hampered in their ability to integrate such a tool into their daily administrative responsibilities. We believe that such a process can be refined, but this will require significant stakeholder engagement and field testing.

Acknowledgments

This research was supported by grants CCF-0424422 and CNS-0964063 from the NSF, R01-LM010207 from the NIH, and HHS-90TR0003/01 from the ONC but the views in the paper are those of the authors only

References

[1] T. Abdul-Basser and D. Riley. Audit log query. Trial Implementations Service Interface Specifications Ver. 1.3.1, Nationwide Health Information Network, 2009.

[2] H. Anderson. Prison terms in patient id theft case. Healthcare Information Security Articles, June 22 2011.

[3] A. Boxwala, J. Kim, J. Grillo, and L. Ohno-Machado. Using statistical and machine learning to help institutions detect suspicious access to electronic health records. *Journal the American Medical Informatics Association*, 18:498–505, 2011.

[4] Center for Health Solutions. Issue brief: Privacy and security in health care: A fresh look. Technical report, Deloitte, 2011.

[5] Y. Chen and B. Malin. Detection of anomalous insiders in collaborative environments via relational analysis of access logs. In *CODASPY*, pages 63–74, 2011.

[6] Y. Chen and B. Malin. Detection of anomalous insiders in collaborative environments via relational analysis of access logs. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 63–74. ACM, 2011.

[7] Y. Chen, S. Nyemba, W. Zhang, and B. Malin. Leveraging social networks to detect anomalous insider actions in collaborative environments. In *Intelligence and Security Informatics (ISI), 2011 IEEE International Conference on*, pages 119–124. IEEE, 2011.

[8] Y. Chen, S. Nyemba, W. Zhang, and B. Malin. Specializing network analysis to detect anomalous insider actions. 1:1, 2012.

[9] Department of Health and Human Services. Proposed Rule: HIPAA Privacy Rule Accounting of Disclosures under the Health Information Technology for Economic and Clinical Health Act. *Federal Register*, 76(104):31426– 31449, May 2011 RIN 0991-AB62.

[10] DICOM Standards Committee, Working Group 14. Digital imaging and communications in medicine (DICOM) audit trail messages. Trial Use Draft Supplement 95, 2004.

[11] D. Fabbri and K. Lefevre. Explaining accesses to electronic medical records using diagnosis information. *Journal of the American Medical Informatics Association*, 20:52–60, 2013.

[12] Fair warning. `http://www.fairwarning.com/`.

[13] R. Gallagher, S. Sengupta, G. Hripcsak, R. Barrows, and P. Clayton. An audit server for monitoring usage of clinical information systems. In *AMIA*, page 1002, 1998.

[14] B. Gregg, H. D'Agostino, and E. G. Toledo. Creating an IHE ATNA-based audit repository. *Journal of Digital Imaging*, 19(4):307–315, 2006.

[15] C. A. Gunter, D. M. Liebovitz, and B. Malin. Experience-based access management: A life-cycle framework for identity and access management systems. *IEEE Security & Privacy Magazine*, 9(5), 2011.

[16] S. Gupta, C. Hanson, C. A. Gunter, M. Frank, D. Liebovitz, and B. Malin. Modeling and detecting anomalous topic access. In *IEEE Intelligence and Security Informatics (ISI)*, page in press, 2013.

[17] IHE International. Ihe it infrastructure (iti) technical framework. Standard Volume 1 (ITI TF-1) Integration Profiles, August 2010.

[18] C. Kaufman. 2 fired at University of Iowa hospitals for peeking at records. De Moines Register, June 20 2011.

[19] J. Kim, J. Grillo, A. Boxwala, X. Jiang, R. Mandelbaum, B. Patel, D. Mikels, S. Vinterbo, and L. Ohno-Machado. Anomaly and signature filtering improve classifier performance for detection of suspicious access to ehrs. In *AMIA Annual Fall Symposium*, pages 723–731, 2011.

[20] G. Marshall. Security audit and access accountability message xml data definitions for healthcare applications. Informational Request for Comments 3881, IETF, 2004.

[21] National Uniform Claim Committee. Health provider taxonomy code set. Technical Report Version 11, American Medical Society, Jan 2011. Available at `http://www.wpc-edi.com/content/view/793/1`.

[22] L. A. News. Ex-UCLA researcher gets 4 months for snooping. `abclocal.go.com`, April 28 2010.

[23] H. Nissenbaum and J. L. Hall. Analysis and recommendations concerning hhs notice of proposed rulemaking covering changes to accountings of disclosure. Letter to the Office of the National Coordinator for Health Information Technology, December 2011. `http://sharps.org/wp-content/uploads/NISSENBAUM-ONC-LETTER.pdf`.

[24] C. Ornstein. Fawcett's cancer file breached. LA Times, April 3 2008.

[25] P2sentinel. `https://store.cerner.com/items/1552`.

[26] P. Pyette. Privacy, access and security services (PASS): Healthcare audit control service patient privacy capabilities. Technical report, Health Level Seven, 2010.

[27] W. Stead and H. Lin, editors. *Computational technology for effective health care: immediate steps and strategic directions*. National Research Council of the National Academies, Washington, DC, 2009.

[28] K. Thomson. Celebrity medical records in massive UCLA breach. HuffPost Entertainment, August 5 2008.

[29] Veriphyr. `http://www.veriphyr.com/`.

[30] H. Zhang, S. Mehrotra, D. Liebovitz, B. Malin, and C. A. Gunter. A patient flow model for scoring anomalies in access logs. Technical report, Northwestern University, 2011.

[31] W. Zhang, Y. Chen, C. A. Gunter, D. Liebovitz, and B. Malin. Evolving role definitions through permission invocation patterns. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, June 2013.

[32] W. Zhang, C. A. Gunter, D. Liebovitz, J. Tian, and B. Malin. Role prediction using electronic medical record system audits. In *AMIA Annual Fall Symposium*, pages 858–867, 2011.