Modifying Without a Trace: General Audit Guidelines are Inadequate for Electronic Health Record Audit Mechanisms

Jason King, Ben Smith, Laurie Williams
North Carolina State University
890 Oval Drive, Campus Box 8206
Raleigh, NC 27695-8206
{jtking, ben_smith, laurie_williams}@ncsu.edu

ABSTRACT
Without adequate audit mechanisms, electronic health record (EHR) systems remain vulnerable to undetected misuse. Users could modify or delete protected health information without these actions being traceable. *The objective of this paper is to assess electronic health record audit mechanisms to determine the current degree of auditing for non-repudiation and to assess whether general audit guidelines adequately address non-repudiation.* We derived 16 general auditable event types that affect non-repudiation based upon four publications. We qualitatively assess three open-source EHR systems to determine if the systems log these 16 event types. We find that the systems log an average of 12.5% of these event types. We also generated 58 black-box test cases based on specific auditable events derived from Certification Commission for Health Information Technology criteria. We find that only 4.02% of these tests pass. Additionally, 20% of tests fail in *all three* EHR systems. As a result, actions including the modification of patient demographics and assignment of user privileges can be executed without a trace of the user performing the action. The ambiguous nature of general auditable events may explain the inadequacy of auditing for non-repudiation. EHR system developers should focus on *specific* auditable events for managing protected health information instead of general events derived from guidelines.

Categories and Subject Descriptors
J.3 [**Computer Applications**]: Life and Medical Sciences – *Medical information systems*

General Terms
Design, Security, Standardization

Keywords
audit, log, user-based non-repudiation, accountability, security, healthcare, privacy

1. INTRODUCTION
Without adequate audit systems to ensure accountability, electronic health record (EHR) systems remain vulnerable to undetected misuse, both malicious and accidental. Users could modify or delete protected health information without these actions being traceable to the modifier. According to Chuvakin and Peterson [3], "If [an organization's information technology] isn't accountable, the organization probably isn't either." Patients need to trust the privacy practices and accountability of healthcare organizations. Administering software audit mechanisms forms a basis for privacy-driven and accountability-driven policy and regulations, including government regulations [8]. The United States Health Insurance Portability and Accountability Act of 1996 (HIPAA) Security and Privacy Rule states that one must implement, "mechanisms that record and examine activity in information systems that contain or use electronic protected health information" [5].

Storing an accurate history of user interaction with a software application and its underlying data helps build a sense of accountability, since a user cannot expressly deny performing certain actions that were recorded by the audit mechanism. In the case of a medical mistake, audit mechanisms can provide a record by which healthcare practitioners can exonerate themselves from legal action by demonstrating that they prescribed the correct drug at a certain time, or that a certain test result was, in fact, what they claim it was. The health informatics field needs standards that address the implementation of software audit mechanisms to monitor access and information disclosure, including details of *what* should be logged, *how* it should be logged, and *when* logged information should be monitored.

The objective of this paper is to assess electronic health record audit mechanisms to determine the current degree of auditing for non-repudiation and to assess whether general audit guidelines adequately address non-repudiation. In performing this study, we investigate the following questions:

R1: What events should be included in an EHR log file for non-repudiation?

R2: What are the strengths and weaknesses of software auditing mechanisms in current open-source EHR systems?

Software audit log files may include system logs and server logs that assist with debugging and troubleshooting. For this paper, we focus on human-readable, semantic user activity logs that contain data related to user actions that may be monitored for the purpose of audit and user accountability. In this study, we first perform an analysis of EHR audit mechanisms by deriving a set of 16 general assessment criteria, derived from four academic and professional sources of *general* auditable events (such as "view data" and "create data"). Next, we perform an analysis by deriving 58 audit-related black-box test cases to assess *specific* user actions (such as "view diagnosis data" and "view patient demographics") in an EHR system. We analyze three open-source EHR systems:

- Open Electronic Medical Records (OpenEMR)[1]
- Open Medical Record System (OpenMRS)[2], with added Access Logging Module[3]
- Tolven Healthcare Innovations's Electronic Clinician Health Record (eCHR)[4] system, with added Performance Plugin[5] module

By evaluating each EHR's audit mechanism with both our general and specific analyses, our goal is to compare and contrast the results and suggest techniques for healthcare software developers to strengthen EHR audit mechanisms.

The remainder of this paper is organized as follows. Section 2 briefly discusses background information related to this study and some key terms and definitions. Section 3 discusses related work with audit mechanisms. Section 4 describes the formulation of our general auditable events and specific auditable events assessment criteria for analyzing non-repudiation in EHR systems. Section 5 presents the open-source EHR systems studied and presents our case studies of evaluating the open-source EHR audit mechanisms. Section 6 discusses the implications and significance of our evaluations. Section 7 presents limitations of our work. Section 8 presents future work in the field of EHR audit mechanisms. Finally, Section 9 summarizes our findings and concludes the paper.

2. BACKGROUND

The United States Department of Justice's Global Justice Information Sharing Initiative defines:

> *non-repudiation* -- a technique used to ensure that someone performing an action on a computer cannot falsely deny that they performed that action. Non-repudiation provides undeniable proof that a user took a specific action [10].

With software systems that manage protected, sensitive data (including EHR systems), a more-specific definition of non-repudiation is needed. We further define the following term based on the definition of non-repudiation above:

> *user-based non-repudiation* – a techniques used to ensure that an authenticated user accountholder performing an action within a software system cannot falsely deny that they performed that action.

Böck, et al., identify four primary concerns regarding software audit mechanism reliability [1]:

- *storage confidentiality* – malicious users should not be able to access log entries
- *machine-based non-repudiation* – log files can be traced to a specific machine to identify the source of the audit entries
- *application-based non-repudiation* – log entries can be traced to trusted software applications such that malicious users cannot manually create fake log entries
- *transmission confidentiality* – accuracy and integrity of log file data is preserved during transmission

Satisfying these concerns is not a simple task, especially for software developers who may implement software audit mechanisms without proactively considering the protection and reliability of the data contained within the log files. Böck, et al., suggest that these four concerns should be considered as a core set of requirements for any software audit mechanism [1]. Yet actually implementing the software and hardware infrastructure to fulfill these requirements may prove challenging.

One motivation for implementing EHR audit mechanisms for user-based non-repudiation involves the mitigation of insider attack. An *insider attack* occurs when employees of an organization with legitimate access to their organizations' information systems use these systems to sabotage their organizations' IT infrastructure or commit fraud [9]. Researchers at the Software Engineering Institute at Carnegie Mellon University released a comprehensive study on insider threats that reviewed 49 cases of Insider IT Sabotage between 1996 and 2002 [9]. According to the study:

- 90% of insider attackers were given administrative or high-level privileges to the target system.
- 81% of the incidents involved losses to the organization, with dollar amounts estimated between "five hundred dollars" and "tens of millions of dollars."
- The majority of attacks occurred after the employees were terminated from the organization.
- Lack of access controls facilitated IT sabotage.

Although federal laws, such as HIPAA, provide legal sanction against tampering with or stealing medical records, we cannot assume that employees working within a medical organization will always follow the rules.

3. RELATED WORK

Related literature has identified several challenges and limitations with software audit mechanisms. Here, we discuss challenges in technology and challenges with policy, regulations, and compliance.

3.1 Challenges in Technology

Audit mechanisms in EHR systems face several challenges and limitations because of technology. We group these challenges into

[1] http://www.oemr.org

[2] http://openmrs.org

[3] https://wiki.openmrs.org/display/docs/Access+Logging+module

[4] http://www.tolven.org/echr.html

[5] http://wiki.tolven.org/doc/index.php/Plugin:org.tolven.performance

two categories: limited infrastructure resources and log file reliability.

3.1.1 Limited Infrastructure Resources

Behind every piece of software lies some sort of hardware configuration. Hardware, itself, provides limitations that affect software. For example, information storage may be restricted to a single hard drive with a limited storage capacity. As a result, EHR systems must manage storage resources carefully.

Another challenge involves distributed software systems. Chuvakin and Peterson suggest that the biggest technological challenge of audit mechanisms involves determining the location at which generating, storing, and managing the log files will be most beneficial for the subject domain and intent of the software application [3]. In these systems, software components may run on separate host machines. For example, one machine may host a database server while a separate machine hosts a web server. In this situation, software audit mechanisms are not as centralized or easy to implement with the physically distributed nature of the overall software application. Here, the actual site of the audit logging functionality is not easy to define [3]. Should software generate audit trails at the web server level, at the database server level, both, or at some third-party location? Software architects must determine the ideal location of user-based non-repudiation audit mechanisms to ensure *all* user accountholder actions are recorded and monitored.

3.1.2 Log File Reliability

Another technological challenge facing software audit mechanisms involves reliability of the audit mechanism, itself. The National Institute of Standards and Technology (NIST) highlights the issue of breach of audit mechanism log data [8]. Audit mechanism log files need protection to ensure that the data contained within the log files is unmodified, accurate, and reliable. Engineering this protection of the audit mechanism log files may be challenging; it may also be overlooked by system developers who are unaware or indifferent to the implications of unprotected log files and inaccurate data that may result from modified logs. In this unprotected situation, log files are no longer trustworthy, the audit mechanism is no longer effective for monitoring user-based non-repudiation, and the accountability of the system is weakened.

3.2 Challenges in Policy, Regulations, and Compliance

In this section, we group policy and regulatory challenges into two categories: ill-defined standards, policies, and regulations; and ineffective log analysis.

3.2.1 Ill-defined Standards, Policies, and Regulations

Standards provide a foundation for consistency and quality. With software systems, coding standards provide a set of guidelines and suggestions for making program code style consistent across software applications. Software developers may choose to ignore standards if they wish, but overall quality and understandability may be sacrificed.

Log file content, timestamps, and formats may vary externally over software companies and internally over software applications of the same company [8]. Distributed web services, for example, may have different policies based on the host machines [3]; the database server may have one set of auditing policies, while the web server may have a completely different set of auditing policies. In addition, the physical location of the distributed systems may cause concern. The organization (or country) that hosts the database server likely has different policies and regulations compared to the organization (or country) that hosts the web server. Furthermore, the transmission of data between these servers may pass through additional organizational authority, which likely introduces an additional degree of varying policies and regulations. Chuvakin and Peterson [3] state that administrators of such complicated distributed systems may not currently enable security features (such as software audit mechanisms) by default; instead, software organizations must actively enable auditing features by choice. Without a default auditing system enabled, user-based non-repudiation and enforcement of accountability would likely decline.

Even if software audit mechanisms are enabled, these mechanisms still face other challenges, such as ambiguous logging requirements. When implementing audit mechanisms, software developers may focus on recording only additions, deletions, and modifications of data. The developers tend to overlook logging the viewing or reading of data, however [11]. In healthcare [5], viewing and reading data in EHR systems is a vital concern when managing protected health information.

Without well-defined standards and regulations by a central governing body, the industry has no widely accepted standard for software audit mechanisms [3], including audit mechanisms in EHR systems. This leaves the responsibility of interpreting and complying with vague regulatory verbiage to individual software development teams who may be unprepared, untrained, or unaware of policies and regulations that govern the software systems upon which they work.

3.2.2 Ineffective Log Analysis

With respect to software audit mechanisms, accountability and non-repudiation implies that the stored log files should be analyzed to monitor compliance. Without log analysis, the audit trail remains unseen, compliance remains unchecked, and accountability remains unmonitored for non-repudiation. Log file analysis seems to fall into three categories: manual, automated, or a combination of both. However, a current lack of efficient automated log file analysis policies and tools often leads to manual log file review [11].

Software companies tend to inadequately prepare, support, and maintain human log file analyzers [8]. Preparation, support, and maintenance of effective human analyzers should include two activities: initial training in current regulations, and continued training in evolving policy, regulation, and case law. The current ineffective training practices in industry likely results in diminished control of accountability and non-repudiation [8].

Schneider [13] compares accountability to defensive strategy: unacceptable actions (such as a receptionist viewing protected health data without authorization) may be capable of being prevented, but must instead be identified to reprimand the given user who performed the unacceptable actions. Schneider suggests analysis methods must be mature enough to identify these users based on digital evidence (such as audit mechanism data), just as law enforcement investigators collect fingerprints from a crime scene. Dixon [4] also suggests this notion of computer forensics – computer data must be preserved, identified, extracted, documented, and interpreted when legal or compliance issues

transpire. Likewise, effective software audit mechanism analysis must preserve, identify, extract, document, and interpret log files entries for user-based non-repudiation.

4. ASSESSMENT METHODOLOGY

Section 4.1 describes our user-based non-repudiation assessment criteria for EHR audit mechanisms, based on *general* auditable events (such as "view data" and "create data"). Section 4.2 describes the development and execution of our black-box test plan to help evaluate the logging of *specific* auditable events (such as "view diagnosis data" and "view patient demographics data") for user-based non-repudiation.

4.1 Assessment using Audit Guidelines and Checklists

Section 4.1.1 describes the derivation of our assessment criteria for user-based non-repudiation based on general auditable event types. Section 4.1.2 describes our methodology for assessing EHR system audit mechanisms.

4.1.1 Derivation of General Auditable Events

Our assessment of user-based non-repudiation first involves compiling a list of *general* events that should be logged in software audit mechanisms, according to other researchers and standards organizations. General events include basic actions such as "viewing" and "updating", but these events do not specify *what information* is viewed or updated. Our goal is to compile a set of common general auditable event types for user-based non-repudiation based on the general guidelines and checklists from four academic and professional sources:

- Chuvakin and Peterson [3] provide a general checklist of items that should be logged in web-based software applications. We collect 17 auditable events from this source.
- The Certification Commission for Health Information Technology (CCHIT)[6] specifies an appendix of auditable events specific to EHR systems. CCHIT is a certification body authorized by the United States Department of Health & Human Services for the purpose of certifying EHR systems based on satisfactory compliance with government-developed criteria for meaningful use [2]. We collect 17 auditable events from this source.
- The SysAdmin, Audit, Network, Security (SANS) Institute provides a checklist of information system audit logging requirements to help advocate appropriate and consistent audit logs in software information systems [7]. We collect 18 auditable events from this source.
- The "IEEE Standard for Information Technology: Hardcopy Device and System Security" presents a section on best practices for logging and auditability, including a listing of suggested auditable events [6]. We collect 8 auditable events from this source.

Combining all four sets of data, we collect 60 total general auditable events and event types. After combining duplicates, our set contains 28 unique auditable events and event types. The only item appearing in all four suggested auditable events sets is "security administration event". Out of the 28 unique events, 18 (64.3%) are contained in at least two of the source sets. Ten events (35.7%) are only contained in one source set. The overlap among the four sources suggests some common understanding and agreement of general events that should be logged, yet the disparity seems to indicate disagreement about the scope and breadth of auditable events. Table 1 provides a comparison of the four source sets of general auditable events and event types.

Next, we categorize each individual auditable event or event type from Table 1 into one of two categories: events that *affect* user-based non-repudiation, and events that *do not affect* user-based non-repudiation. Our categorization is denoted in Table 1 under the "Affects User-based Non-repudiation" column. When categorizing these events, we determine if the given event should be traced to a specific user accountholder in an EHR system. If so, we categorize this event as one that affects user-based non-repudiation. If the event need not be traced to a specific user accountholder, we categorize the event as one that does not affect user-based non-repudiation. For example, the "view data" event suggests a user accountholder (such as a physician) has authenticated into an EHR system and is viewing protected patient health information. The action of viewing this protected data should be traceable to the physician's user account. Therefore, this event is categorized as one that does affect user-based non-repudiation. On the other hand, an "application process failure" does not suggest any intervention by a user accountholder. Instead, this event suggests an internal EHR system state change. Therefore, we categorize this event as not affecting user-based non-repudiation.

Of the 28 total unique auditable events and event types, we identify 16 unique events that affect user-based non-repudiation. Of these 16 actions, only 9 events (56.25%) are suggested by two or more of the sources. The remaining 7 events (43.75%) are contained in only one source set.

4.1.2 General Auditable Events Assessment Methodology

For each EHR system, we deploy the software on a local web server following the deployment instructions provided by each EHR's community website. Next, we consult official documentation typically provided on the website for each of the EHR systems. In the documentation (typically user guides, development guides, or community wiki pages) we search for sections on auditing and logging to understand how to access these mechanisms in the actual application. Once we understand how to access the auditing mechanism, we open our locally-deployed EHR system and attempt to access these features to continue our analysis. We document all of our observations or difficulties during this analysis process for reflection after the analysis is complete.

Once we have either physical access to or a general understanding of the given application's auditing mechanism, we record the following information:

1. A flag (satisfied or unsatisfied) for each of the assessment criteria listed in the "Logging Actions" column of Table 2.
2. Any observations or important findings that may influence the results or provide justifications for results

[6] http://www.cchit.org

Table 1. A comparison of auditable events by source, with a categorization of events affecting user-based non-repudiation

Auditable Events	**Source of Software Audit mechanism Checklist**				**Affects User-based Non-repudiation**
Log Entry Item	*Chuvakin and Peterson [3]*	*CCHIT [2]*	*SANS [7]*	*IEEE [6]*	*(Yes or No)*
System startup	X	X	X		N
System shutdown	X	X	X		N
System restart			X		N
User login/logout	**X**	**X**	**X**		**Y**
Session timeout		**X**			**Y**
Account lockout		**X**			**Y**
Create data	**X**	**X**	**X**		**Y**
Update data	**X**	**X**	**X**		**Y**
Delete data	**X**	**X**	**X**		**Y**
View data	**X**	**X**	**X**		**Y**
Query data		**X**			**Y**
Node-authentication failure	X	X	X		N
Signature created/validated		**X**			**Y**
Export data		**X**			**Y**
Import data		**X**			**Y**
Security administration event	X	X	X	X	N
Scheduling		X			N
System backup	**X**	**X**			**Y**
System restore		**X**			**Y**
Initiate a network connection	X		X	X	N
Accept a network connection			X	X	N
Grant access rights	**X**		**X**	**X**	**Y**
Modify access rights	**X**		**X**	**X**	**Y**
Revoke access rights	**X**		**X**	**X**	**Y**
System, network, or services changes	X		X	X	N
Application process abort/failure/abnormal end	X		X		N
Detection of malicious activity	X		X		N
Changes to audit log configuration				X	N

4.2 Black-box Audit Test Cases

Our assessment of user-based non-repudiation also involves constructing a black-box test plan for testing an EHR system's recording of *specific* auditable events (such as "view diagnosis data"). In this paper, we briefly describe the process for the audit test cases used to evaluate user-based non-repudiation audit functionality. We developed this methodology in earlier work [14].

In 2006, through a consensus-based process that engaged stakeholders, CCHIT defined certification criteria [2] focused on the functional capabilities that should be included in ambulatory (outpatient) and inpatient EHR systems. The requirements specifications contain 284 different functional descriptions of EHR behavior.

The CCHIT ambulatory certification criteria contain eight requirements related to audit. The audit requirements contain functionality such as "The system shall allow an authorized administrator to set the inclusion or exclusion of auditable events based on organizational policy & operating requirements/limits." One CCHIT audit criterion states that the set of auditable events in an EHR system should include the following fourteen items:

1. Application start/stop
2. User login/logout
3. Session timeout
4. Account lockout
5. Patient Record created/viewed/updated/deleted
6. Scheduling
7. Query
8. Order
9. Node-authentication failure
10. Signature created/validated
11. PHI Export (e.g. print)
12. PHI import
13. Security administration events
14. Backup and restore

The list provided here verbatim from the CCHIT ambulatory criteria. The criteria are vague. For example, the phrase "security administration events" is undefined and could relate to authentication attempts, deletion of log files, or assigning user privileges. Likewise the term "scheduling" could relate to scheduling patient appointments, scheduling system backups, or scheduling system down-time for maintenance. The interpretation of these phrases varies, and the intended meanings are ambiguous.

Due to the vagueness in these auditable events, we elected to approach the CCHIT certification criteria as a general functional requirements specification. The criteria describe functionality for EHR systems, such as editing a patient's health record, signing a note about a patient, and indicating advance directives (e.g. a do-not-resuscitate order). Using these functional CCHIT requirements [2], we develop a set of 58 black-box test cases[7] that assess the ability of an EHR system to audit the user actions specified by these CCHIT requirements. These test cases all involve a registered user performing a given action within the EHR system, therefore representing an assessment of user-based non-repudiation within each EHR system. The 58 test cases correspond to 58 individual CCHIT requirements statements. Our test plan covers the 20.4% of the CCHIT requirements that are relevant to personal or protected health information. The remaining 79.6% of the CCHIT requirements do not pertain to personal health information, and therefore do not necessitate an audit record for user-based non-repudiation.

We iterated through each of the 284 ambulatory CCHIT requirements, extracting keywords and applying a template (see Section 4.2.1) to produce a test case when necessary. We generate a test case from a specific requirement based on keywords within the requirements statement. Requirements that include key phrases like "problem list," "clinical documents," and "diagnostic test" all indicate the user's interaction with a piece of a patient's protected health information, and should therefore necessitate the application of our test cast template.

Additionally, we extract an action phrase (e.g. "edit") and an object phrase (e.g. "demographics") from each relevant requirement to construct the black-box test case. We present the template used for these black-box tests in Section 4.2.1, and present an example of a test case and its corresponding requirement in Section 4.2.2.

4.2.1 Audit Test Case Template

Test Procedure Template:

1. Authenticate as <*insert a registered user name*>.
2. Open the user interface for <*insert action phrase*>ing an <*insert object phrase*>.
3. *Verb* an <*insert object phrase*>with details.
4. Logout as <*insert a registered user name*>.
5. Authenticate as <*insert an administrator's user name*>.
6. Open the audit records for today's date.

Expected Results Template:

- The audit records should show that *registered user* <*insert action phrase*>ed an <*insert object phrase*>.
- The audit records should be clearly readable and easily accessible.

4.2.2 Audit Test Case Example

Example Natural Language Artifact:

- CCHIT Criteria: AM 03.08.01 – The system shall provide the ability to associate orders and medications with one or more codified problems/diagnoses.

Example Test Procedure:

1. Authenticate as Dr. Robert Alexander.
2. Remove the association between Theodore S. Smith's Hypertension diagnosis and Zantac.
3. Add the association back between Theodore S. Smith's Hypertension diagnosis and Zantac.
4. Logout as Dr. Robert Alexander.
5. Authenticate as Denny Hudzinger.
6. Open the audit records for today's date. If necessary, focus on patient Theodore S. Smith.

Example Expected Results:

- The audit records should show adding and removing the association of Theodore S. Smith's Hypertension diagnosis and Zantac, both linked to Dr. Robert Alexander, and with today's date.
- The audit records should be clearly readable and easily accessible

5. CASE STUDIES

We conduct a case study of three open-source EHR systems using our assessment methodology described in Section 4. Section 5.1 describes the EHR systems we used in this case study. Section 5.2 describes our EHR audit mechanism assessment based on the general auditable events assessment criteria from Section 4.1. Then, Section 5.3 describes our black-box test case evaluation of three open-source EHR systems.

5.1 Open-source EHR Systems Studied

In this study, we compare and contrast audit mechanisms from three open-source EHR systems. The criteria for inclusion in this study involved (1) being open-source for ease-of-access, and (2) having a fully-functional default demo deployment available online. For this study, we assess the following EHR systems:

- Open Electronic Medical Records (OpenEMR) system,
- Open Medical Record System (OpenMRS), with added Access Logging Module
- Tolven Healthcare Innovations's Electronic Clinician Health Record (eCHR) system, with added Performance Plugin[8] module

A summary of some of the facts about these software applications appears in Table 2.

[7] http://healthcare.zapto.org/doku.php?id=public:cchit_black_box_security_test_plan#audit_test_scripts

[8] http://wiki.tolven.org/doc/index.php/Plugin:org.tolven.performance

Table 2. Summary of open-source EHRs studied

	Version / Release Date	License	Clientele	Added Modules
OpenEMR	3.2.0 / February 16, 2010	General Gnu Public License	>30 million clients	None
OpenMRS	1.6.1 / March 28, 2010	OpenMRS Public License	International client base	Access Logging Module
Tolven eCHR	RC1 / May 28, 2010	Lesser General Public License	US, Europe, Asia-Pacific	Performance Plugin

5.2 User-based Non-repudiation Assessment

The objective of our user-based non-repudiation assessment of the three EHR systems is to identify a percentage of satisfaction for *user-based non-repudiation* events in Table 1. Since these auditable events are general (for example, "view data" is a general form of the event "view diagnosis data"), this assessment evaluates the effectiveness of following such general auditable events guidelines when implementing audit mechanisms in EHR systems. From Table 1, we use the set of auditable events that affect user-based non-repudiation as the basis for our analysis. As we assess each open-source EHR system, we mark each auditable user-based non-repudiation event as being satisfied or unsatisfied in the given EHR's audit mechanism. To calculate the percentage of user-based non-repudiation satisfaction, we divide the number of satisfied actions by the total number of user-based non-repudiation actions.

A summary of our assessment criteria observations appears in Table 3. The Access Logging Module within OpenMRS satisfies 18.75% (3 out of 16) of our general auditable events user-based non-repudiation criteria. The OpenMRS audit mechanism seems to focus on creating, updating, and viewing patient demographics and encounters. The auditing mechanisms of OpenEMR and Tolven eCHR both satisfied fewer auditable events than the OpenMRS audit mechanism. In OpenEMR, for example, the audit mechanism only addresses user logins/logouts and viewing of data. Likewise, Tolven eCHR's audit mechanism only addresses user logins/logouts. The purpose of the Tolven eCHR Performance Plugin mechanism involves system performance logging, not specifically user-access logging. Therefore, the Tolven eCHR logs are not easily parsed by humans for monitoring user-based non-repudiation.

We also find that OpenEMR allows unrestricted access for administrative users to both view and modify the audit log database tables via a default installation of phpMyAdmin[9]. The running copy of phpMyAdmin is enabled and configured by default in OpenEMR, and is accessible through an administrator's login to the application. This administrative access to the audit log file contents effectively renders the OpenEMR log files untrustworthy and unreliable, as any malicious administrative user could alter the log table entries to cover wrongdoings or hide unauthorized accesses to protected health information. For this study, we did not factor audit log file immutability into our analysis.

[9] http://www.phpmyadmin.net

Table 3. Satisfaction of user-based non-repudiation criteria

EHR System	Criteria Met	Criteria Not Met	Satisfaction Percent
OpenEMR	2	14	12.5%
OpenMRS	3	13	18.75%
Tolven eCHR[a]	1	15	6.5%

5.3 User-based Non-repudiation Assessment with Black-box Test Cases

We executed the 58 black-box test cases on OpenEMR, OpenMRS, and Tolven eCHR. We present the results of our black-box test plan in Table 4. We use the following system to classify the results of executing the test cases on these two electronic health record systems:

- **Pass**: The system met the test case's specified preconditions, and the actual results matched the expected results. The test case did not reveal any audit issue.
- **Fail**: The system met the test case's specified preconditions, but one or more results did not match the expected results. The test case revealed an audit issue.
- **PNM (**Precondition not met): We could not execute the test case due to constraints in the system's configuration or setup, or perhaps because the test case makes an assumption about the system that simply is not true.
- **N/A**: The test case could not be executed because we could not find the functionality specified in the requirements. These systems are not CCHIT-certified, so a missing implementation for a requirement is understandable.

Of our 174 test cases (58 test cases applied on each of the three systems), 50% failed, meaning our test plan revealed an event that should be logged by the system but was not. Additionally, we analyzed the number of test cases that failed for all three of the systems and found 12 (or 20.7%) examples of system functionality that was implemented by the systems, yet produced no audit record when performed. Examples of user actions that failed in both systems include:

- Assigning privileges or restrictions to users and groups
- Session timeout
- Changing passwords
- Maintaining the diagnoses associated with a patient
- Recording the prescribing of medications
- Displaying and maintaining an allergy list
- Managing diagnostic tests and test results

As a result, any of these actions may take place without a recorded trace of the identity of the user who performed the action in these systems

Table 4. Results of user-based non-repudiation black-box test cases for auditing in OpenEMR and Tolven eCHR

System	Pass	Fail	PNM	N/A	Total
OpenEMR	3	37	0	18	58
OpenMRS	4	23	1	30	58
Tolven eCHR	0	27	2	29	58
Total	7	87	3	77	174
Percent	4.02%	50.00%	1.72%	44.25%	

6. MODIFYING WITHOUT A TRACE

Our user-based non-repudiation black-box test results reveal several scenarios in which user behavior is not properly audited, based on CCHIT criteria requirements. The audit log of user actions within a medical records system is essential. Doctors and other healthcare practitioners depend on the accuracy and availability of the data in the healthcare system to make life or death decisions about patient care. In the case of a medical mistake, audit mechanisms can provide a record by which healthcare practitioners can exonerate themselves from legal action by demonstrating that they prescribed the correct drug at a certain time, or that a certain test result was, in fact, what they claim it was. Further, with no audit mechanisms in place for user-based non-repudiation, patients and doctors, alike, could forge medical records with no chance of getting caught. For example, a doctor could retroactively create a record of the completion of a certain test to exonerate herself from a medical malpractice charge.

In both Tolven eCHR and OpenEMR, the modification of patient demographic data is not recorded to the audit log file. Neither application's audit mechanism records log entries concerning the user modifying the demographics, the patient whose demographics data are modified, the timestamp, or other relevant information. A patient's demographic data is considered protected data under HIPAA, yet the audit mechanisms fail to track changes to this data. Medical records contain personal and sensitive information about what procedures and tests a patient has had, as well as diagnoses that a patient has received from doctors. For example, some medical diagnoses are stigmatized, like a sexually transmitted disease diagnosis. Other information can be life threatening, such as allergies. Insurance companies as well as employers are interested in knowing a patient's health record to make unethical decisions about whether to cover a patient or whether to hire a patient, respectively.

Additionally, in an insider threat scenario, a rogue administrative user could assign special privileges to a collaborating malicious user for creating prescription orders. This privilege would not otherwise be associated with the given user. None of the OpenEMR, OpenMRS, or Tolven eCHR audit mechanisms record the assignment of privileges to users or groups of users. The assignment of this privilege would go undetected. Further complicating the scenario, neither OpenEMR, OpenMRS, nor Tolven eCHR record the creation of prescription orders. In this case, not only will the assignment of the privilege go unrecorded, but the actual creation of prescription orders would go unrecorded, as well. The combination of unaudited events would greatly benefit the malicious insider threat. With proper auditing and monitoring polices in place, however, such a threat would be mitigated.

Both our general auditable events and (specific auditable events assessments indicate inadequacies in EHR system audit mechanisms. These two approaches to evaluating the EHR audit mechanisms highlight a key concern for developers of EHR software. If developers of EHR audit mechanisms rely only on generalized checklists of general auditable events such as CCHIT's "view data" and "update data", they may unintentionally overlook some of the EHR-specific auditable events for certain types of protected data.

In Section 3, we discuss a lack of industry-wide standards, policies, and regulations for audit mechanisms. Considering R1, we find auditable events guidelines defined by organizations such as CCHIT too general to ensure adequate auditing in an EHR system. Whereas our assessment based on these general auditable events checklists from Chuvakin and Peterson, CCHIT, SANS, and IEEE finds OpenEMR satisfying 12.5% of the generalized user-based non-repudiation events, the fine-grained black-box assessment finds OpenEMR really satisfies only 5.2% of specific auditable events. Likewise, our assessment based on general auditable event types finds OpenMRS satisfying 18.75%, compared to 6.9% of our black-box test cases. Furthermore, Tolven eCHR satisfies 6.5% of our general auditable event type criteria and 0% of black-box test cases. The specific auditable events from our assessment represent actual healthcare-specific user actions within an EHR system. Therefore, the black-box audit assessment approach in Section 4.2 provides a more fine-grained, accurate assessment of user-based non-repudiation compared to the 16 general auditable events compiled as our general auditable events assessment criteria. The healthcare field could benefit from mature, well-defined standards and regulations to ensure consistency, adequacy, and widespread adoption of adequate audit mechanisms for ensuring strong user-based non-repudiation in EHR systems.

With respect to R2, we observed some weaknesses of the three open-source EHR audit mechanisms. Above all, the lack of audited events by the three EHR systems is concerning for user-based non-repudiation in systems that manage protected health information. Additionally, related research attempts to enhance audit reliability by proposing fair and irrefutable auditing techniques [12]. In terms of audit mechanism reliability and user-based non-repudiation, however, a software audit mechanism's effectiveness depends on the accuracy of events that are logged. If any person accesses the stored audit log files (such as an administrative user using the included phpMyAdmin installation in OpenEMR to access log entry contents), these files should be considered untrustworthy, inaccurate, and tainted. Steps must be taken to ensure log files cannot be altered, fabricated, or destroyed. If log files are unprotected, the software audit mechanism, as a whole, is effectively useless in terms of accurate recording, enforcement of accountability, and user-based non-repudiation.

7. LIMITATIONS

One limitation of this study involves our manual parsing and interpretation of audit log entries through each EHR's user interface. We based our criteria satisfaction and black-box test plan results on the information provided by only the audit mechanism's user interface. We did not consider system logs, server logs, or debug logs for this assessment. Additionally, we only evaluated three open-source EHR systems. These three systems may not be representative of the level of auditing that other EHRs may provide. Furthermore, proprietary systems, in-particular, may contain a more complete level of auditing for user-based non-repudiation to help detect malicious user behavior.

Likewise, our derivation of general auditable events in Section 4 only relies upon four academic and professional sources, only one of which (CCHIT) is healthcare-specific and relates directly to EHR systems. There may be additional healthcare-specific audit guidelines and checklists for user-based non-repudiation in related literature.

With respect to the user-based non-repudiation black box test plan, the CCHIT requirements may not be representative of every piece of functionality present in an EHR, and our test plan is based only on the CCHIT requirements. Also, we may have misjudged which CCHIT requirements relate to protected health information for our black-box assessment, and human error may have resulted in a missing test case that should assess that a specific user action is auditable. Likewise, we manually derived the assessment criteria for our general auditable event types assessment. Varying perspectives and interpretations of the meaning of some of the ambiguous auditable events (such as "security administration event" and "scheduling") may alter whether on considers the event as affecting user-based non-repudiation or not.

In terms of our local EHR system deployments, we may have overlooked some system configurations that may affect auditing for user-based non-repudiation. We based our assessments on default installations of the three EHR systems, and a more fine-grained configuration may affect the system's auditing for user-based non-repudiation.

8. FUTURE WORK

First, additional research could investigate and propose procedures for designing, implementing, and maintaining tamper-proof, accurate, and reliable software audit mechanisms. We found OpenEMR's inclusion of unrestricted phpMyAdmin access to all database table contents detrimental to the reliability of the EHR system's auditing mechanism. Even with adequate auditing mechanisms for user-based non-repudiation, the contents of the audit log files are useful and effective only if the log files can be trusted and immutable.

As mentioned in Section 7, one limitation of this study involves manually interpreting data presented in each audit mechanism's user interfaces. Additional research could lead to assessment criteria concerning manual or automated monitoring techniques for user-based non-repudiation. We found the Tolven eCHR audit mechanism interface not easily human-readable, since it is primarily a system performance auditing mechanism. For log analysis and monitoring for user-based non-repudiation, log monitoring should not be a complicated, tedious task.

Third, having audit mechanisms defined as functional requirements in a software requirement specification (with accompanying test cases) may lead to better user-based non-repudiation in EHR audit mechanisms. In Section 5, we constructed our black-box test plan for user-based non-repudiation by parsing the CCHIT certification criteria, which are presented similar to functional software requirements. Parsing these CCHIT criteria as functional requirements led to the extraction of more *specific* auditable events for our assessment. Similarly, if audit mechanism implementations were based on functional requirements, increased user-based non-repudiation might be achieved.

9. CONCLUSION

Storing an accurate history of user interaction with a software application helps build a sense of accountability and non-repudiation, since a user cannot expressly deny performing certain actions that were recorded by the audit mechanism. According to Chuvakin and Peterson [3], "If [an organization's information technology] isn't accountable, the organization probably isn't either."

Current software audit mechanisms face challenges and limitations associated with ensuring adequate user-based non-repudiation. Our general auditable events assessment criteria and specific auditable-event black-box test cases both show major weaknesses with user-based non-repudiation. With an average of 12.5% of our general auditable events assessment criteria met, our specific auditable events black-box test evaluation reveals only 4.02% of audit-related test cases pass for all three EHR systems. This disparity highlights a key problem with following generalized guidelines for auditing in EHR systems handing protected health information. Instead of assessing general events such as "view data" in our general auditable event types assessment, our specific auditable events approach assesses specific, EHR-related actions such as "view diagnosis data", "view patient demographics data", and "view prescription data" as separate auditable events. The more-specific nature of our specific auditable events assessment provides a more accurate evaluation of user-based non-repudiation. Software developers for EHR systems should focus on specific auditable events for managing protected health information, instead of basing their audit mechanisms on guidelines or checklists that contain generalized auditable event types.

Even so, both assessments reveal severe inadequacies in EHR audit mechanisms for user-based non-repudiation. Even though HIPAA mandates the implementation of an audit mechanism in health information systems software, it fails to explicitly define any guidelines or standards to ensure adequate audit mechanisms to help ensure accountability of users who have access to protected health information. Without strong audit mechanisms to ensure accountability and responsibility, healthcare software remains vulnerable to undetected misuse, both malicious and accidental, including insider threat.

10. ACKNOWLEDGMENTS

The National Science Foundation under CAREER Grant No. 0346903 supports this work. Any opinions expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

11. REFERENCES

[1] Böck, B., Huemer, D., and Tjoa, A.M., "Towards more trustable log files for digital forensics by means of 'Trusted computing'," in *AINA '10, Proceedings of the 24h IEEE International Conference on Advanced Information Networking and Applications*. Perth, Australia: IEEE Press, 2010, pp.1020-1027.

[2] "CCHIT Certified 2011 Ambulatory EHR," Certification Commission for Health Information Technology, 2011, Available: http://www.cchit.org/certify/2011/cchit-certified-2011-ambulatory-ehr.

[3] Chuvakin, A., and Peterson, G., "Logging in the age of web services," *IEEE Security and Privacy*, vol. 7, no. 3, May 2009, pp. 82-85.

[4] Dixon, P., "Overview of Computer Forensics," *IEEE Potentials*, vol. 24, 2005, pp.7-10.

[5] HIPAA § 164.312(b), "Technical Safeguards," 2007, Available: http://edocket.access.gpo.gov/cfr_2007/octqtr/pdf/45cfr164.312.pdf.

[6] "IEEE standard for information technology: Hardcopy device and system security," *IEEE Standard*, 2008, pp.1-177.

[7] "Information system audit logging requirements," SANS Institute, 2007, Available: http://www.sans.org/security-resources/policies/info_sys_audit.pdf.

[8] Kent, K., and Souppaya, M., "Guide to Computer Security Log Management," National Institute of Standards and Technology, Gaithersburg, Maryland, USA: 2006.

[9] Moore, A.P., Cappelli, D.M., and Trzeciak, R.F., *The "Big Picture" of Insider IT Sabotage Across U.S. Critical Infrastructures*, Carnegie Mellon Software Engineering Institute. CERT Program, 2008.

[10] "Privacy Technology Focus Group: Final Report and Recommendations," United States Department of Justice Global Justice Information Sharing Initiative. September 2006, p.57.

[11] "Revolutionizing health care through information technology," National Coordination Office for Information Technology Research and Development, Arlington, Virginia, USA: 2004, Available: http://www.nitrd.gov/Pitac/meetings/2004/20040617/20040615_hit.pdf.

[12] Robinson, P., Cook, N., and Shrivastava, S., "Implementing fair non-repudiable interactions with Web services," in *EDOC '05, Proceedings of the 9th IEEE International Enterprise Computing Conference*. 2005, pp. 195- 206.

[13] Schneider, F., "Accountability for perfection," *IEEE Security & Privacy*, vol. 7, no. 2, 2009, pp. 3-4.

[14] Smith, B., and Williams, L., "Systematizing Security Test Planning Using Functional Requirements Phrases". North Carolina State University, Technical Report #2011-5.

12. APPENDIX

Track: Systems

Focus: Computing, Information Science, Security, Software Engineering

Topics Covered:

- Evaluation of health information systems
- Privacy in healthcare
- Security in healthcare