# OpenEMR Auditing and ATNA
# Design and Implementation

Prepared by ViSolve

**Contact:** 408.666.4320

**Email:** vicareplus_support@visolve.com

www.vicareplus.com

18th January 2010

Table of Contents

**Revision History**

| Version | Date | Author | Reviewed By |
|---|---|---|---|
| 1.0 | 01/08/2010 | ViCarePlus Team | Team |
| 2.0 | 01/18/2010 | ViCarePlus Team | Team |

# 1. Introduction

The purpose of this document is to describe the design and high-level implementation of the auditing requirements in OpenEMR. There are two main standards that define auditing requirements for EHR systems:

CCHIT Ambulatory Audit Requirements
http://www.cchit.org/sites/all/files/
CCHIT%20Certified%202011%20Ambulatory%20EHR%20Criteria%2020091006.pdf

ATNA (Audit Trail and Node Authentication)
http://www.ihe.net/Participation/upload/IHE-ITIATNA-CT-08.ppt

# 2. Existing Audit Framework in OpenEMR 3.1.0

OpenEMR 3.1.0 contains an existing audit framework. The audit table is the 'log' table, which has the following columns:

| | | |
|---|---|---|
| id | bigint(20) | The primary key |
| date | datetime | The date and time of the event |
| event | varchar(255) | The possible events are "login", "logout", "view", "delete", "authorize" |
| user | varchar(255) | The user logged in. |
| groupname | varchar(255) | The group the user belongs to. |
| comments | longtext | For login/logout, the comments include the IP address of the user, and whether the login was successfull.<br>For view, the comments include the name of the patient. |
| user_notes | longtext | Not used. |

The function

```
function newEvent($event, $user, $groupname, $comments="")
```

in file openemr-3.1.0/library/log.inc is used to add an entry to the audit 'log' table.

OpenEMR currently logs the following events:

- login - File auth.inc. An audit entry is made for both successfull and failed logins.
- logout - File auth.inc. An audit entry is made for both direct logouts, and logouts due to timeouts.
- view - File auth.inc. Once a patient is selected, every time a page is visited, a 'view' event is generated with that patient's name.

# 3. Audit Requirements Review

Both CCHIT and ATNA provide a list of events to audit. The table below lists both the CCHIT and ATNA audit events, and a brief description of each event, as per our understanding.

**Table 1 – Audit Events Description**

| CCHIT Event | ATNA Event | Description |
|---|---|---|
| start/stop | actor-start-stop | The application (Apache and MySQL) is started or stopped.<br>Apache: /usr/local/apache/log/error_log<br>MySQL: /usr/local/mysql/data/<host>.err |
| user login/logout | | A user performs a successful or failed login/logout. These events are already audited in the file openemr/library/auth.inc |
| session timeout | | A user attempts to access a page using an expired session. This event is already audited in auth.inc |
| account lockout | | A user account has the 'active' field set to yes/no by the administrator. OpenEMR currently does not have any automatic lockout after a number of failed logins. |
| patient record created, viewed, udpated, deleted | procedure-record-event<br>patient-record-event | A patient record is created, modified, viewed, or deleted. |
| scheduling | patient-care-assignment | A patient appointment is scheduled or updated. |
| query | images-availability-query<br>query-information | Some persistent data (data in the database) is queried. For example, hospital or pharmacy information. This suggests that all SQL SELECT queries must be audited. |
| order | order-record-event | An order is placed for a medical service or medical item (like a prescription). |
| node-authentication-failure | node-authentication-failure | A node fails to authenticate by using an incorrect SSL client certificate. SSL client certificate failures are logged to the Apache errorlog. |
| signature created/validated | | Also, a document or piece of data is signed with an RSA/DSA key. Currently, OpenEMR does not create/validate digital signatures on documents. |
| PHI export | PHI export | Patient information is exported. This also overlaps with a patient-record view event. |
| PHI import | PHI import | Patient information is imported. This also overlaps with a patient-record create/update event. |
| security administration | security administration | Events such as creating/updating users and groups that login to the system. |
| backup and restore | | Backing up or restoring the OpenEMR data. |
| | audit-log-used | The audit log was read or modified. |
| | begin-storing-instances | |
| | health-service-event | Any misc health related auditable events. |
| | instances-deleted | A DELETE is performed on data in the database. |
| | instances-stored | An INSERT or UPDATE of data is performed on the database |
| | medication | Medication is prescribed or delivered. |
| | mobile-machine-event | Mobile equipment is relocated, leaves the network, and rejoins the network. |
| | patient-care-episode | Patient information other than medical records (such as an appointment). |
| | study-object-event | A study report is created, modified, or deleted. |

| | | |
|---|---|---|
| | study-used | A study report is viewed. |

The audit entries should contain the following information:

- The date and time of the entry
- The type of event
- The patient identifier (where relevant)
- The user identity
- The client certificate name
- The group the user belongs to
- The outcome (success or failure)
- The checksum for the current record (for ‘update’,’insert’ and ‘replace’ statements)
- Details of the event [The entire sql query is stored]

When transmitting audit entries to a separate ATNA repository machine, the following additional information is needed:

- Hostname or IP address of machine that generated the audit entry
- The application which generated the entry (OpenEMR)
- The IP address of the destination ATNA audit repository.

# 4. Auditing the new Events (Implementation)

The events are audited only if the ‘enable_auditlog’ is set to 1 in globals.php.

```
$GLOBALS["enable_auditlog"] = 1;
$GLOBALS["audit_events"] = array("patient-record"=>1,
                                 "scheduling"=>1,
                                 "query"=>1,
                                 "order"=>1,
                                 "security-administration"=>1,
                                 "backup"=>1,
                                );
```

We can also enable/disable the logging of following configurations through globals.php
patient-record
scheduling
query
order
security-administration
backup

The audit log table includes three additional columns: patient_id, success and client certificate name
The database.sql file will be modified to include these additional columns.

Next, we would like to audit the events above without having to modifying every PHP page. Almost all the events above involve reading or writing data to the database. We can therefore capture all these audit events by intercepting each database call, determining the event type, and writing an audit entry. For example, the sqlStatement() function would be modified as follows:

```
function sqlStatement($statement)
{
  $query = mysql_query($statement, $GLOBALS['dbh']);
  if ($query === FALSE) {
    auditSQLEvent($statement, FALSE);
    HelpfulDie("query failed: $statement", mysql_error($GLOBALS['dbh']));
  }
  auditSQLEvent($statement, TRUE);
  return $query;
}
```

The sqlStatement() is just one place where database calls are made. All locations where database calls are made must be modified to also call auditSQLEvent().

The function auditSQLEvent() will create an audit entry based on the SQL query made. The outcome of the query (success or fail) is also passed to the function. The auditSQLEvent() function will work similar to the following pseudocode:

```
function auditSQLEvent($statement, $outcome) {
  $user = $_SESSION['authUser'];
  $group = $_SESSION['authGroup'];
  $comments = $statement;
  $success = ($outcome === FALSE) ? 0 : 1;
  $event = get_audit_event_type($statement);
  $patient_id = 0;
  if (array_key_exists('pid', $_SESSION) && $_SESSION['pid'] != '') {
    $patient_id = $_SESSION['pid'];
  }
  $sql = "insert into log (date, event, user, groupname, comments, patient_id, success) "
   . "values ( NOW(), " . qstr($event) . ", " . qstr($user) . "," . qstr($group) . ","
   . qstr($comments) . "," . qstr($patient_id) . "," .qstr($success) . ")";
}
```

We can determine the event type based on the tables viewed/modified in the SQL statement.
Below is a summary of which tables correspond to which audit events:

**Table 2 - Audit Events Implementation**

| **CCHIT/ATNA Event** | **Implementation** |
|---|---|
| start/stop<br>actor-start-stop | No OpenEMR changes needed.<br>Use the Apache/MySQL log files. |
| user login/logout | Use existing audit calls in auth.inc. |
| session timeout | Use existing audit calls in auth.inc |
| account lockout | The query string contains "UPDATE users" and "active = 0" |
| patient record created, viewed, udpated, deleted<br>patient-record-event<br>procedure-record-event | The query string contains one of the following tables:<br><br>billing<br>claims<br>employer_data<br>forms<br>form_encounter<br>form_dictation<br>form_misc_billing_options<br>form_review_ofs<br>form_ros<br>form_soap<br>form_vitals<br>history_data<br>immunizations<br>insurance_data<br>issue_encounter<br>patient_data<br>payments<br>pnotes<br>prescriptions<br>transactions<br>lists<br>onotes |

| | |
|---|---|
| scheduling<br>patient-care-assignment | The query string contains the table 'openemr_postcalendar_events' |
| query<br>query-information<br>images-availability-query | Logging all the “select” statements |
| order<br>order-record-event | The query string contains the table 'drugs'. |
| node-authentication-failure | No OpenEMR changes needed.<br>Use the Apache error log. |
| signature created/validated | OpenEMR currently doesn't use digital signatures for documents. |
| PHI export | Currently, we are unable to distinguish between a PHI export and a patient-record-view event. |
| PHI import | Currently, we are unable to distinguish between a PHI import and a patient-record-update event. |
| security administration | The query string contains one of the following tables:<br>users<br>groups<br>facility<br>pharmacies<br>addresses<br>x12_partners<br>insurance_companies<br>codes<br>registry<br>phone_numbers<br>gacl_acl<br>gacl_acl_sections<br>gacl_acl_seq<br>gacl_aco<br>gacl_aco_map<br>gacl_aco_sections<br>gacl_aco_sections_seq<br>gacl_aco_seq<br>gacl_aro<br>gacl_aro_groups<br>gacl_aro_groups_id_seq<br>gacl_aro_groups_map<br>gacl_aro_map<br>gacl_aro_sections<br>gacl_aro_sections_seq<br>gacl_aro_seq<br>gacl_axo<br>gacl_axo_groups<br>gacl_axo_groups_map<br>gacl_axo_map<br>gacl_axo_sections<br>gacl_groups_aro_map<br>gacl_groups_axo_map<br>gacl_phpgacl |
| backup and restore | Logging happens through backup.php.  No restore activity present in openemr |
| **Events specific to ATNA** | |
| audit-log-used | The query contains the table "log". |
| begin-storing-instances | |
| health-service-event | |
| instances-deleted | A DELETE query is performed that doesn't match any other events.. |
| instances-stored | An INSERT or UPDATE query is performed that doesn't match any other events. |
| medication | The query contains the table 'prescriptions'.  This can also be classified as a patient-record event, since medication is prescribed to a specific patient. |
| mobile-machine-event | |
| patient-care-episode | |

| | |
|---|---|
| study-object-event | . |
| study-used | |

# 5. Sending Audit Events to an ATNA Repository

In order to send audit events to an ATNA repository:
- We need the hostname/IP of the repository server
- We need the port of the server
- We need to generate an RFC 3881 XML Audit message
- We need to connect to the audit repository server, and send the message.

In globals.php, we can add two entries to specify the ATNA repository server:

```
// File globals.php
// Uncomment to forward audit entries to an ATNA audit repository server
// $GLOBALS['atna_audit_repository'] = 'host.com'
// $GLOBALS['atna_audit_repository_port'] = 6514;
```

The other variables used are:
+// atna_audit_localcert - Certificate to send to RFC 5425 TLS syslog server
+// atna_audit_cacert - CA Certificate for verifying the RFC 5425 TLS syslog server

If the variable $GLOBALS['atna_audit_repository'] is set, then the auditSQLEvent() function will call an additional function to send the audit entry to the ATNA repository.

The function create_rfc3881_msg() will create the XML message given the audit data:

```
function create_rfc3881_msg($date, $user, $group, $event, $patient_id,
                            $outcome, $comments);
```

The message will have the following format:

```
--------------------------------------------------------------------
<?xml version="1.0" encoding="ASCII"?>

<AuditMessage
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="healthcare-security-audit.xsd"
>

<EventIdentification
  EventActionCode="$eventActionCode"
  EventDateTime="$eventDateTime"
  EventOutcomeIndicator="$outcome"
>
  <EventID
    code="$eventIDcode"
    displayName="$eventIDdisplay"
    codeSystemName="DCM"
  />

</EventIdentification>

<ActiveParticipant
  UserID="$srcUserID"
  UserIsRequestor="true"
  NetworkAccessPointID="$srcIP"
  NetworkAccessPointTypeCode="2"
>
  <RoleIDCode
    code="110153"
    displayName="Source"
    codeSystemName="DCM"
  />
</ActiveParticipant>

<ActiveParticipant
  UserID="$destUserID"
  UserIsRequestor="false"
  NetworkAccessPointID="$destIP"
  NetworkAccessPointTypeCode="2"
>
  <RoleIDCode
    code="110152"
    displayName="Destination"
    codeSystemName="DCM"
  />

</ActiveParticipant>

<AuditSourceIdentification AuditSourceID="OTHER_SOURCEID" />

<ParticipantObjectIdentification
  ParticipantObjectID="$userID"
  ParticipantObjectTypeCode="$userTypeCode"
  ParticipantObjectTypeCodeRole="$userRole"
>

  <ParticipantObjectIDTypeCode
    code="$userCode"
    displayName="$userDisplay"
    codeSystemName="RFC-3881"
   />

  <ParticipantObjectDetail
    type="SQL Query"
    value="Base64-encoded SQL statement"
   />

</ParticipantObjectIdentification>
</AuditMessage>
--------------------------------------------------------------------
```

The main variables are described below:

- $eventActionCode - This value is based on the type of SQL statement being audited.  It will have the value 'C' for create, 'R' for view/select, 'U' for update, 'D' for delete, and 'E' for login/logout entries.
- $eventDateTime - The current date/time, using the PHP format string DATE_ATOM.
- $eventIDcode - The choice of event codes is up to us.  We will copy the event codes used by https://iheprofiles.projects.openhealthtools.org/, which contains a Java implementation of ATNA auditing.  They use the following codes:
  110101 (Audit Log Used)
  110106 (PHI Export)
  110107 (PHI Import)
  110110 (Patient Record)
  110111 (Procedure Record)
  110122 (Login), 110123 (Logout)
- $eventIDdisplay - The event value of the audit record
- $srcUserID - The hostname of OpenEMR plus the application name.

$_SERVER['SERVER_NAME']  . "|OpenEMR"

- $srcIP - The IP address of OpenEMR, $_SERVER['SERVER_ADDR']
- $destUserID - The hostname of the audit repository $GLOBALS['atna_audit_repository']
- $destIP - The IP address of the audit repository $GLOBALS['atna_audit_repository']
- $userID - The username for Login/Logout events, or the patient id for patient-record events
- $userTypeCode - 1 (for person)
- $userRole - 1 (for patient) or 6 (for user)
- $userCode - 2 (for patient number) or 11 (for user identifier)
- $userDisplay - Either "Patient Number" or "User Identifier"

# 6. Creating an ITI-19 (SSL) connection

When transmitting the audit records to an audit repository, OpenEMR must establish a secure connection according to ITI-19.  That is, OpenEMR must make an SSL/TLS connection with the remote machine, and perform bi-directional authentication, sending its own client certificate, and verifying the remote machine's certificate.

An ITI-19 SSL connection is also required for other features, such as transmitting PHI (patient health information) to another HIE.  A generic function create_tls_conn() will be used to make an ITI-19 secure connection:

```
/* Create a TLS (SSLv3) connection to the given host/port.
 * $localcert is the path to a PEM file with a client certificate and private key.
 * $cafile is the path to the CA certificate file, for
 *  authenticating the remote machine's certificate.
 * If $cafile is "", the remote machine's certificate is not verified.
 * If $localcert is "", we don't pass a client certificate in the connection.
 *
 * Return a stream resource that can be used with fwrite(), fread(), etc.
 * Returns FALSE on error.
 */
function create_tls_conn($host, $port, $localcert, $cafile) {
    $sslopts = array();
    if ($cafile != "") {
        $sslopts['cafile'] = $cafile;
        $sslopts['verify_peer'] = TRUE;
        $sslopts['verify_depth'] = 10;
    }
    if ($localcert != "") {
        $sslopts['local_cert'] = $localcert;
    }
    $opts = array('tls' => $sslopts, 'ssl' => $sslopts);
    $ctx = stream_context_create($opts);
    $timeout = 60;
    $flags = STREAM_CLIENT_CONNECT;

    $conn = stream_socket_client('tls://' . $host . ":" . $port, $errno,
                                 $errstr, $timeout, $flags, $ctx);
    return $conn;
}
```