Sentry App and Sentry Technology Add-ons

A proactive privacy monitoring application to audit for appropriate use

Table of Contents

Intent

The Sentry application is a privacy monitoring tool that incorporates third party application access logs and expands them with demographic data into consumable views that enable a user to visually interpret details and drive investigations for inappropriate use.

Prerequisites

You should be familiar with the Splunk search language (SPL) and know how to install Splunk apps before attempting to install the Sentry app and Sentry add-ons. Additionally, if you do not utilize the DB Connect method of retrieving data for employee, customer and access logs, you will need to provide the data in the form defined by the Sentry common information model. Developing a Sentry add-on will require some knowledge of the Splunk search language, HTML/CSS, Django, and Javascript.

How it works (out-of-box)

The Sentry framework allows developers to create a Sentry technology add-on and incorporate additional dashboards, views, or pages into the existing app seamlessly. This is accomplished by utilizing Django template inheritance to reuse base code throughout your application.

On a scheduled basis searches are run against the sentry index for the previous day's access which produces reports that will drive individual view. These reports are stored within the sentry_summary index.

There are static lookups that define expected column values that must be adhered to. Please see "Lookups" for more information.

Additionally, auxiliary lookups are derived from the sentry index on a scheduled basis. These lookups are required for some reports to function. Please see appendix "saved search schedule" for timing requirements.

Out of the box, Sentry uses three indexes and numerous csv lookup tables to augment and enhance access and demographic data.

Indexes

sentry_stage:

The sentry _stage index is optional and is used to store the narrow access logs in its native format. This data is subsequently enhanced using employee and customer lookups. This index can be ignored if the access log is inserted into the sentry index and complies with the Sentry common information model.

sentry:

The sentry index is used to store the access data and available demographic data for both customer and employee.

sentry_summary:
The sentry_summary index is used to store the report data or the subset of data that drives each view. This is what most of the searches reference to populate a view.

Lookups

App.csv:
This csv is used to populate the "Apps" dropdown in the search controls panel. If you create your own technology add-on, you will need to update this file to include your app to be available as a selection choice.

> For information on maintaining this lookup see the following:
> "Configure Sentry to use your new application data"

Common Last Name:
This lookup is populated by getting the top 1,000 common last names from the customer data. If you do not choose to generate a customer lookup, this lookup will need to be populated in some other fashion.

Customer is Employee:
This lookup is populated by comparing the customer and employee demographic lookups to infer the customer is an employee. If you do not choose to generate a customer lookup, this lookup will need to be populated in some other fashion.

Customer.csv (optional):
This csv contains customer demographic data.

Employee.csv (optional):
This csv contains employee demographic data.

Emp Types:
This lookup defines a set of employee types. If your demographic data does not differentiate, please use "EMP" as the default.

Access Types:
This lookup defines a set of access severities. If your demographic data does not differentiate, please use "Major Event" as the default.

Triage Status:
This lookup defines the available triage status options. This can be customized, however, a non-trivial amount of changes are required in the controls Django template (sentry_base_with_*).

Installation

Install the Sentry app

1. Download and install the Sentry app from Splunk's app store and any applicable Sentry technology add-ons available.
2. Restart the Splunk instance to complete the installation.
    a. $SPLUNK_HOME/bin/splunk restart

Setup Sentry

If you initially index a backlog of access data and would like to backfill reports, you will need to run each saved search beginning with "Report: *" for the appropriate time range. Otherwise, you will not receive reports until the first scheduled run.

Sentry Technology Add-on (TA) Setup Requirements

Any TA built for the Sentry framework will need to modify the framework installation in the following ways:

1. Only if TA contains views see the following section:
    a. "Configure Sentry to use your newly created "sentry_ta_exampleApp"
2. Only if TA defines data for a new application (re: "app" column) see the following section:
    a. "Configure Sentry to use your new application data"

Use Cases

The Sentry add-on for Epic incorporates Epic specific access logs into the existing Sentry framework utilizing Sentry's existing scenarios and pages to view Epic data. The Sentry add-on for Epic can be found through Epic and will not be hosted on Splunkbase.

Create a Sentry Technology Add-on app (sentry_ta_exampleApp)

1. Open a command prompt and navigate to $SPLUNK_HOME/etc/apps/framework.
2. If you're using Mac OS X or Unix, enter the following:

   ./splunkdj createapp sentry_ta_exampleApp

   If you're using Windows, enter:

   splunkdj createapp sentry_ta_exampleApp

```
:~/etc/apps/framework$ ./splunkdj createapp sentry_ta_exampleApp
```

3. You'll need to provide your Splunk username and password to create the app—enter your credentials when prompted.

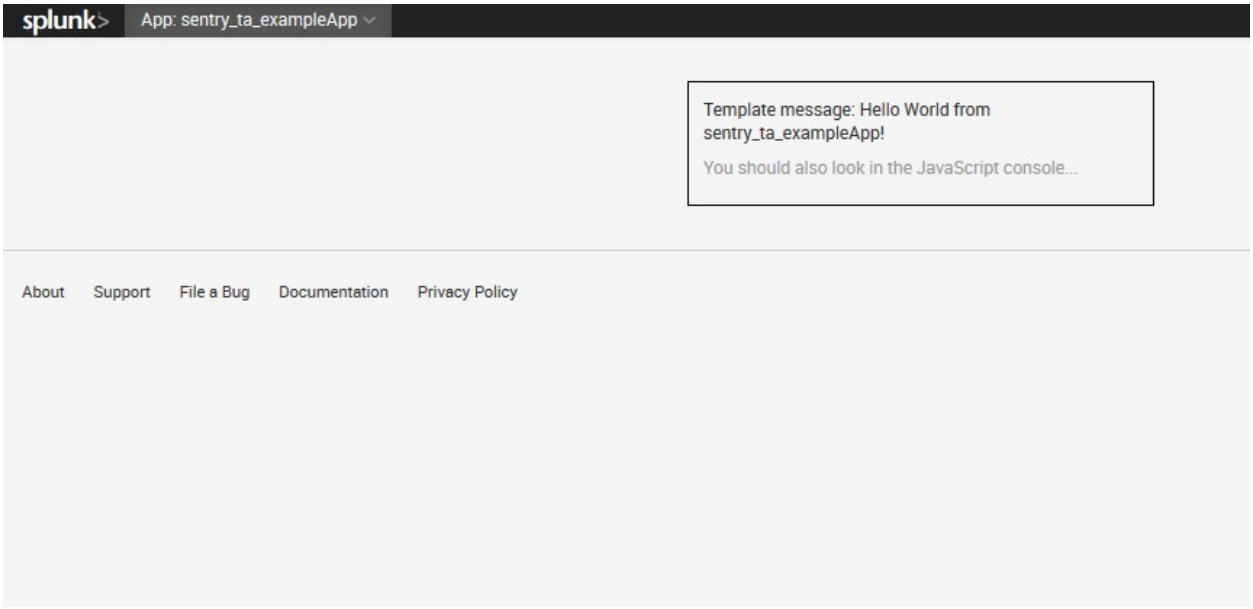
   The $SPLUNK_HOME/etc/apps/sentry_ta_exampleApp directory is created with auto-generated files for the new "sentry_ta_exampleApp " app.

4. Restart Splunk.

   $SPLUNK_HOME/bin/splunk restart

```
Stopping splunkweb...
                                                          [  OK  ]
Stopping splunkd...
Shutting down.  Please wait, as this may take a few minutes.
..
```

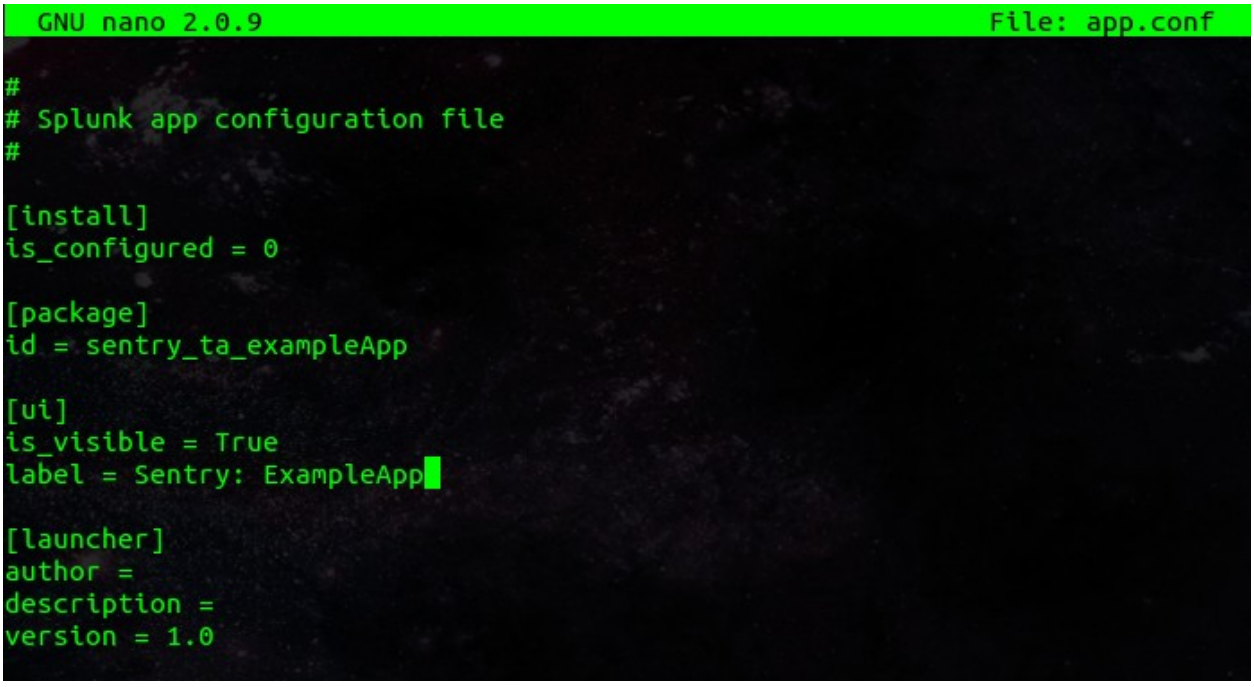
5. Open Splunk Web and find your new app's home.html page:



Modify the App.conf

1. Navigate to the default directory within your app.
2. Open the *app.conf* in a text editor of your choosing.
3. Under the **[ui]** stanza, modify the label section to a user friendly app name.
   I.e. “Sentry: ExampleApp”.

4. Save and close app.conf.



Copy Sentry components into your newly created "sentry_ta_exampleApp"

1. Copy the following from Sentry's app into your new app's directory.

   From: etc/apps/sentry/sentry_do_not_remove.txt
   To: etc/apps/sentry_ta_exampleApp/sentry_do_not_remove.txt

   From: etc/apps/sentry/metadata/
   To: etc/apps/sentry_ta_exampleApp /metadata/
   Note: Contains configuration for default access permissions.

Configure Sentry to use your newly created "sentry_ta_exampleApp"

Note: These instructions are to serve as an example; we will create a page within our sentry_ta_exampleApp called examplePage.

1. Within the Sentry app, navigate to the local/data/ui/nav directory.
2. In a text editor, open the default.xml file.
3. Locate the **Application** collection label and add your own app's page to the file.

   For example, the highlighted green section is where you need to insert your collection. The highlighted light blue code is the new collection that is referenced in our example. And the highlighted grey section is the applicable **Epic** technology add-on that can be installed if you use the Epic application.

```
<collection label="Application">
        <!-- Add app menu collections with <collection> tags-->
            <collection label="Epic">
                <!-- Add menu items with <a> links -->
                <a href="/dj/sentry_ta_epic/break_the_glass">Break the Glass</a>
            </collection>

        <!—My new technology app -->
            <collection label="ExampleApp">
                <!-- Add menu items with <a> links -->
                <a href="/dj/sentry_ta_exampleApp/examplePage">Example Page</a>
            </collection>
</collection>
```

4. Save and close the default.xml file.
5. You can either wait 5 minutes for the navigation bar to sync automatically or you can trigger a manual synchronization via the Diagnostic page.

Configure Sentry to use your new application data

An addition to the app.csv within the lookups directory is required when adding Sentry data from a new application. The label column is what is displayed in the App dropdown selectors. The value column must match exactly the "app" field value in the data itself.

Example (../lookups/app.csv):

```
label,value
Epic,Epic
"Application Two",app2
```

Configure your new app's navigation (urls.py and views.py)

1. Within your newly created app, navigate to "django/sentry_ta_exampleApp/".
2. Open the urls.py conf within your text editor. The urls.py conf is a mapping between the url and the view/page to display. In our example, we will add an entry for our examplePage.

```
urlpatterns = patterns('',
  url(r'^home/$', 'sentry_ta_exampleApp.views.home', name='home'),
  url(r'^examplePage/$', 'sentry_ta_exampleApp.views.examplePage', name='examplePage'),
)
```

3. Save and close the urls.py file.

4. Next, open the views.py conf within your text editor. At a high level, the views.py conf serves up the actual page and can be used to pass variables or data to the template (the actual HTML file).
5. Copy and paste the "home" page's syntax and modify (the emphasized font denotes required changes).

```
@render_to('sentry_ta_exampleApp:examplePage.html')
@login_required
def examplePage(request):
  return {
        "message": "Hello World from sentry_ta_exampleApp - example page!",
        "app_name": "sentry_ta_exampleApp"
  }
```

6. Save and close the views.py file.

Create the examplePage template

The Sentry app comes with example code to help facilitate developers to create their own Sentry technology add-ons and incorporate it into the existing Sentry framework.

The example code is contained within the **appserver** directory in the Sentry app's file system.

1. Using the example code, locate the *templatePage.html* file and copy/paste the entire contents into your *examplePage.html* file located in your app's django/<your_app_name>/templates/ directory.

I.e. ***etc/apps/sentry_ta_exampleApp/django/sentry_ta_exampleApp/templates/examplePage.html***

2. Save the examplePage.html file.
3. With examplePage.html still open in your text editor, you can review the code and look for the "INSERT YOUR" placeholder comments denoting where you can make modifications.
4. Additionally, there is an example table element (tableTopTenUserAccounts-Example) that is used to reference how the relation between the HTML, Javascript and Django code functions.
    a. For example, looking at the search manager, the **id** referenced is "searchTopTenUserAccounts-Example". This **id** is then used as the **managerid** for the javascript TableElement view. At the end of the javascript object is the **el** property, which is set to "#tableTopTenUserAccounts-Example". And this is where the HTML tag is used to render the table element.
    b. Below the **Views** comment in the javascript block, there is an **Event Handlers** section that describes how events can be handled. One example shows how mouse clicks on a chart element can start a button click.

```
chartTopEmployee.on("click:chart", function(e) {
        e.preventDefault();
        tokens.set({"selUserID": e.value});
        tokens.set({"selTriageUserID": e.value});
        $("#buttonSearch").click();
});
```

    c. The other example shows how a mouse click on a table element can call the *redirector* javascript function. Note: the redirector function parameters have been truncated to make it readable.

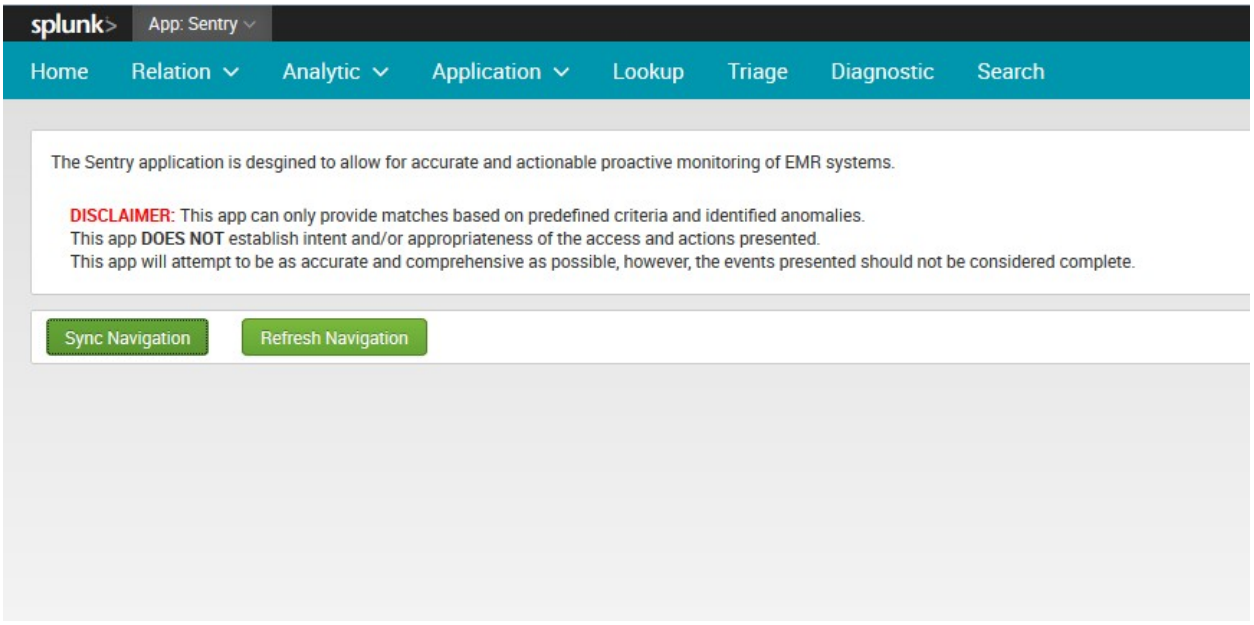
```
TopTenUserAccounts-Example.on("click", function(e) {
        e.preventDefault();
        var earlyTime = tokens.get("earlyTime");
        var lateTime = tokens.get("lateTime");
        var selApp = tokens.get("selApp");
        redirector({…. 'earlyTime': earlyTime, 'lateTime': lateTime}, 'lookup');
});
```

5. Save and close the examplePage.html file.

Sync navigation bar from your newly created “sentry_ta_exampleApp” into Sentry

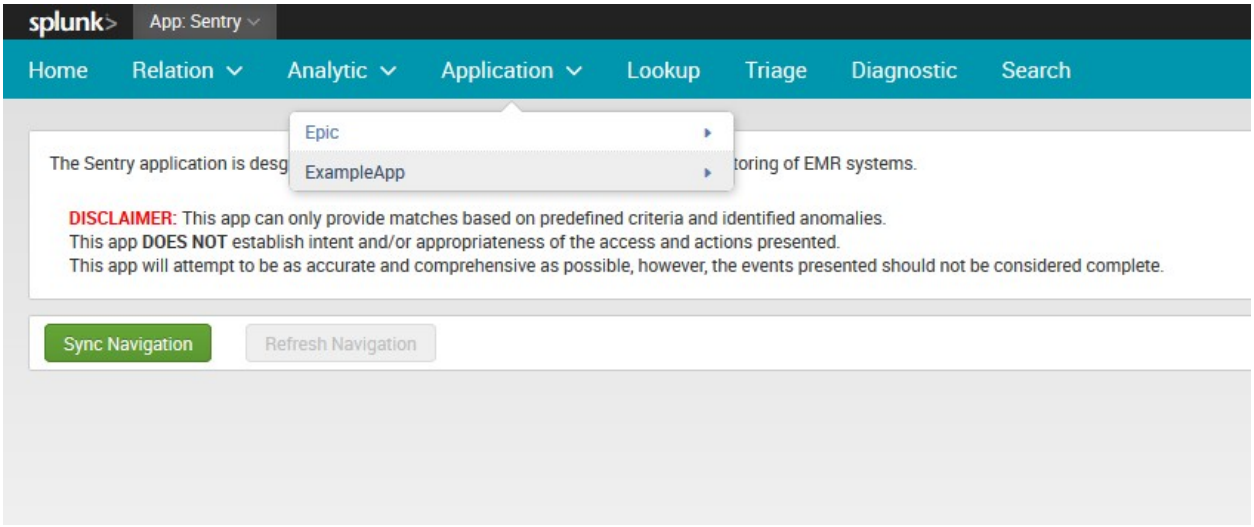
Note: The navigation bar (default.xml) is configured to replicate itself into any Sentry technology add-ons on a regular interval. The following Sentry page and instructions are to manual initial that replication.

1. To sync the navigation bar, open the Sentry app and navigate to the **Diagnostic** page.
2. Click the **Sync Navigation** button.
3. The **Refresh Navigation** button will become enabled (changes into a green button); click the **Refresh Navigation** button now.




```
Entity refresh control page
===========================
'''
Forces a refresh on splunkd resources

        This method calls a splunkd refresh on all registered EAI handlers that
        advertise a reload function.  Alternate entities can be specified by appending
        them via URI parameters.  For example,

            http://localhost:8000/debug/refresh?entity=admin/conf-times&entity=data/ui/manager

        will request a refresh on only 'admin/conf-times' and 'data/ui/manager'.

        1) not all splunkd endpoints support refreshing.
        2) auth-services is excluded from the default set, as refreshing that system will
           logout the current user; use the 'entity' param to force it
'''

Refreshing /data/ui/nav                 OK
DONE
```

4. After the newly opened tab shows **DONE** as the previous screenshot depicts, close the tab and refresh the Sentry page.
5. You should now be able to select the **Application** navigation bar dropdown to view your newly created app's navigation page.



Appendix

12:00-1:00pm - Log Indexing & Demo Update	
dbmon-tail://ClarityPrd/access_audit_epic	12:10:00
dbmon-tail://ClarityPrd/access_audit_epic_btg	12:20:00
Lookup: Customer Demographic Update	12:25:00
Lookup: Employee Demographic Update	
1:00-2:00pm - Auxiliary Lookup Updating	
Lookup: Common Last Name Update	13:00:00
Lookup: Customer is Employee Update	13:10:00
5:00-6:00pm - Enrichment	
Enrich: Access Audit Epic	17:00:00
Enrich: Misc Epic Break the Glass	17:00:00
6:00-7:00pm - Auxiliary Enrichment	
Enrich: Access Audit Customer is Employee	18:15:00
7:00-8:00pm (Previous Day's access available)	
Stat: Non-Employee	19:00:00
Report: Employee Viewing Address	19:00:00
Report: Epic BTG Failure without Success	19:05:00
Report: Employee Viewing Self	19:10:00
Report: Direct Report Viewing Manager	19:20:00
Report: Employee Viewing Employee	19:25:00
Report: Customer Walk Count	19:30:00
Report: Non-Employee Access	19:35:00
Report: Manager Viewing Direct Report	14:40:00
Report: Non-Employee Access No Demographics	19:40:00
Report: Non-Employee Viewing Lastname	19:45:00
Report: Employee Viewing Last Name	19:50:00

Sentry Common Information Model (CIM)

This application will not cover the nuances of collecting and properly extracting fields from data. For this information please reference the extensive Splunk documentation.

The Sentry framework application is designed to interact with data placed into the "sentry" index. Saved searches and scenarios will summarize into the "sentry_summary" and is referenced by their respective views. One way to implement Sentry is by creating an employee and customer lookup and staging the narrow access log into "sentry_stage", then subsequently enriching the data and collecting into the "sentry" index. This method reduces indexing volume and complexity of demographic data enrichment for subsequent applications that you may want to incorporate into the Sentry Framework. That said, this method of data collection is not a requirement as long as the data ends up in the "sentry" index with the following columns of data.

Note: Not all data elements are a hard requirement, however, if data that is integral to a scenario and it is not populated, that scenario will not function.

Note: Not all fields need to have 100% coverage, however, a single hyphen "-" is expected instead of Null.

Note: Some fields must be populated with a value at all times. These fields are in red below.

Access Log Fields

Note: For "access_type", if there is no differentiation in the severity of the access, a value of "Major Event" should be used for access_type.

Note: "app" must be populated consistently with the desired name of the application.

access_instant, access_time, access_type, app ,metric_id ,metric_name, workstation_id

Customer Fields

cust_addr_line1, cust_addr_line2, cust_birth_date, cust_city, cust_first_name, cust_home_phone, cust_id, cust_last_name, cust_pat_id, cust_ssn_last4, cust_state, cust_work_phone, cust_zip, cust_pcp

Employee Fields

user_id, emp_addr_line1, emp_addr_line2, emp_birthdate, emp_city, emp_dept, emp_facility, emp_first_name, emp_hr_dept, emp_id, emp_job_title, emp_last_name, emp_mailstop, emp_middle_name, emp_org, emp_phone, emp_ssn_last4, emp_state, emp_type, emp_zip, mgr_first_name, mgr_last_name

Customer is Employee Fields

Note: this data is utilized via lookups and enriched on the fly, it is not meant to be in the "sentry" index data. The population of this lookup can by dynamic or static but must be done in a Sentry TA.

cust_id, cust_user_id, cust_dept, cust_facility, cust_hr_dept, cust_job_title, cust_mailstop, cust_org, cust_type, cust_mgr_first_name, cust_mgr_last_name