elysium
ANALYTICS

OPEN DATA MODEL

HIGH FIDELITY THROUGH LAZY NORMALIZATION

Due to the rise in security breaches more companies are turning to another type of security product to discover unknown threats – *the security data lake.*

This enables customers to build transportable analytics: third-party vendors can create a source specific analytics (for instance MS Windows) that can be used by any Security Data Lake customer who has Windows.

Abstract

Any Data Lake takes muscle, know how and thoughtfulness to get the Data Model, Analytics and Total Cost of Ownership aligned with the key milestones necessary to hit production dates and value on a solution lifecycle basis. This white paper reviews capabilities that streamline and scale the ability to build a modern Security Data Lake for both Open Source environments. Highlighted in the paper are unique capabilities that optimize key considerations for configuring a lasting high value solution. Included in the paper are considerations that touch on Enrichment, Source Mapping Views, and Integration with legacy Security SIEMS and devices, Context Mapping, Context Extensions and more. A high level Schema Diagram is included in Appendix A as well as a diagram highlighting a Simplified Version of Login Events.

What's unique about Elysium's Open Data Model

- ***Data Model Hierarchy with normalized-event definitions***
- ***Preserve data source fidelity through raw base tables***
- ***Context table: user, network, endpoint, VP***
- ***Extensible with third party data models***
- ***Abstraction layer for seamless flow of new sources into analytics***

Elysium Core Set of Layers

Raw Data Tables

At the root of all the layers of views are the raw source tables.

Structured and Unstructured Storage

In order to get the best of all worlds, Elysium's solution allows customers to store all data in either structured or unstructured formats. Unstructured storage is better for human interactive queries and investigations, while structured storage is better for automated advanced analytics such as machine learning, artificial intelligence, and advanced reporting, and can be more efficient at scale for compliance reporting. Elysium customers can easily select both where data flows (structured, unstructured, or both) and how long each the data is retained in each of these formats with simple configuration options.

This paper focuses on the structured data. Unstructured data is simply enriched and sent to the indexer (Elastic Search or Solr) without any normalization or extensive parsing.

Enriched, Parsed, Not Normalized

Structured data sent to Hive tables is enriched and parsed as completely as possible to accommodate a wide variety of reporting and analytical needs. However it is not normalized, it is kept in its "natural" form. This is very different than most other SIEM and Log Management products that store data in structured formats. Other products often have a predefined schema that all sources must fit into (for instance CEF format), forcing a normalization process on all content. This works acceptably for sources of types close to original intent of the product (for instance networking and OS data in CEF), however it is limiting when trying to extend functionality to different types of security such as complex new high-level applications. Further, its not suited to where the market is now taking advantage of Machine Learning (a core competency for Elysium).

By storing all fields without normalization, retaining all the original data in its "natural" form, all possible analytics can be accommodated including future needs. This also enables the most thorough forensic investigations since all the data is present without alteration.

This does mean a separate Hive table is needed for each source type. As an example a first set of base tables may look something like this:

- Nix_syslog_su
- Nix_syslog_ftp
- MS_Windows_2007 or 2010
- BlueCoat
- Watchguard
- MacOS
- MS_Exchange_2010
- Cisco_ASA
- Cisco_IOS

As you see there is a different table for each different product, even if there are two different products that do the same thing. In some cases when a product has significant changes in a new release, a separate table is needed for separate versions. Each table will have a wide variety of event types depending on the nature of the product. In some cases, like MS Windows, it makes sense to further divide the table into different event type-categories to keep the total number of columns down to a reasonable number.

Source Mapping Views

Source Mapping Views are the second layer in the schema architecture. These views are specific to each source (product): they are developed and maintained along with the parser for each source.
Now that we have all the raw “natural” data preserved, we can begin normalizing the data for a variety of analytics. As mentioned each table will have a variety of event types depending on the nature of the product. Firewalls may have just a handful of types of events, such as connection granted, connection denied, administrative activity, etc. Other products such as MS Windows will have dozens of different event types, everything from user logins, account creation, account modification, file access, process startups, etc.

Mapping Views can add information about the source such as the vendor name, product name, version number, etc. In addition to handle subtle variations between event types unique to a product that does not warrant a completely new Master View, Elysium has included an Event_Type field with a hierarchy of values that can be used in analytics. Therefore, as an example, in the ‘login” master view in the field Event_Type we may have values “login-successful” and “login-failed”.

Mapping Views can also be used to handle source schema changes. Schema changes can occur when vendors upgrade their products or when configuration changes are made. The mapping view can include logic based on the cut-over date/time. Queries requesting data before the cut-over can be directed to the old schema, and queries after the cut-over can be directed to the new schema, and results are combined with a union function.

Event Master Views

The Event Master Views are the third layer in the schema architecture. These views are specific to each type of event. Typically they are developed independently from any specific source. However, as mentioned, sometimes a new source will contain a new type of event, and in this case a new Master View is created while working on the new source.

Each Event Master View should have specific business meaning that is relevant to the business or to cybersecurity. It is this meaning and any related analytics that should define the Event Master View schema. There should be no (or very little) information that is specific to a specific source or vendor product. It is best practice to design the Master Views with at least two different vendor’s sources in mind.

Additional Sets of Layers

While the first set of layers of tables and views are organized for specific purpose, this lays the foundation for several points of integration. Depending on the nature of integration it may make sense to integrate to any layer. Following are two example integrations to show the possibilities. The first is an integration with ArcSight and the second is for an integration with Apache Spot.

ArcSight Integration Using CEF

The goal of the ArcSight integration is to enable the Elysium solution to collect data and forward to ArcSight in CEF format for event correlation. The Common Event Format (CEF) is defined by ArcSight, and using this format is the easiest way to load data into ArcSight.

Although the CEF schema is a “one size fits all” approach, there are many fields that need to be set based on the event type. This means we can easily leverage the top-level Master Views which are based on event type. The result is one more layer on top of the master views we termed the CEF Master Views. These views are written directly on top of the base Master Views so none of that logic needs to be repeated. The only logic needed in the CEF Master Views is column mapping.

Optionally one can add filtering logic to the CEF Master Views. This can result in great benefits to ArcSight performance. By letting the Security Data Lake be the long-term storage for all log data, one only needs to send data to ArcSight if it is an event type that might actually result in an alert from a correlation rule. This can reduce the data sent to ArcSight by an order of magnitude which both reduces license requirements and increases performance.

Apache Spot Integration Using ODM

The Spot program has defined three context schemas, one for networks (domains), one for endpoints, and one for users. Each of these schemas includes timestamps, similar to event tables. The result is a detailed running history of all events from the viewpoint of the network, the viewpoint of the endpoint, and the viewpoint of the user.

This *context-based* design is a complete alternative to the Elysium *event-based* design. Therefore the point of integration is at the raw base tables, the first layer of the Elysium stack. We still only need to load the data once to provide both types of analytics. We simply add a layer of Context Mapping Views at the same level as the Source Mapping Views, and we add a layer of Context Master Views at the same level as the Event Master Views.

Although the raw data only needs to be loaded and stored once, there are at times a need to store additional data. ODM defines some fields with information that is not necessarily found in any particular raw event, but can be found in other IT systems or from other events previously loaded. For this Elysium has designed “Summary Tables” that contain the latest, best-known information about each context entity. As data is loaded information is both pushed and pulled from these Summary Tables. Therefore the integration includes more than just views, it also includes some extensions to the existing enrichment routines.

Context Mapping Views

Since the logic for determining which field should appear in each context view is different for each different source (and even for each field within each source), the context mapping views are written on top of the raw data tables. Any given source might have any number of user context mappings, network context mappings, and endpoint context mappings depending on how many usernames, IPs, hostnames, etc are in the event.

As an example MS Windows may have 20 different Context Mapping Views since there are many different ways a hostname can appear in different events, plus several different ways a user name may appear in events. A simple firewall may have only three views, one endpoint view for source IP, one endpoint view for destination IP, and one user view for a user ID.

Context Master Views

In a similar fashion to Elysium’s existing Event Master Views, three Context Master Views are created: one for networks, one for endpoints, and one for users. Each Context Master View pulls information from all the context mapping views of their own type, even if this means pulling the same raw event multiple times, once for each

field that relates to a context. Again this should be the only logic in the Master Views along with the chore of performing a union over all the data.

Now we have another set of views with specific context meanings, independent of the vendor sources they came from. If you want to see the complete history of any particular user, host, or domain, you can easily all the events across all types of products. In addition any analytics written to this layer will be future-proof to the addition of new sources. When a new source is added (the new raw table and the Context Mapping Views) the data from the new source will automatically show up in existing analytics without any change to the analytics.

Context Summary Tables

Many of the fields in the above context master views are not in the source raw data tables at all, but rather need to be looked up in reference tables. Therefore Elysium designed Summary Tables to keep the latest, best known information we have about each of the three asset types. These tables are unique by each of the three contexts. That is: a network (domain) table unique by domain, endpoint (IP or hostname) table unique by endpoint, and user table unique by userID. Note that none of these tables have timestamp as part of the unique key, so only aggregate information can be stored in these tables.

There are no fields of type timestamp that relate to the time of an event. The conceptual time for all information in these tables is “now”, these tables represent the latest and best known information about a given context. There are fields of type timestamp that refer to the “first” or “last” time something occurred, such as “created”, “updated”, “first seen”, “last seen”, etc.

These Summary Tables are updated in two ways, by the enrichment processes as data is loaded (described above), and by synchronization with other IT systems in the environment (e.g. Active Directory or an HR system for users).

In addition Elysium has designed three “children” Summary Tables that are unique by each of the combined keys of the parent Summary Tables. That is one table unique by user ID and hostname, one unique by user ID and domain, and one unique by hostname and domain. Again no timestamps includes in the unique keys. The purpose of these tables is to quickly see the first and last time the two contexts were seen together – for instance the first and last time a user has visited a certain domain, or the first and last time a user has touched a certain machine. This has been very helpful for customers with large data sets to quickly find the scope (time range) that an investigation may be needed, for instance if threat intelligence is received late about a specific domain.

Elysium Context Extensions

One of the key aspects of an open Security Data Lake is that a variety of analytics can be built all leveraging the same source data. Where appropriate, new information from these analytics can be stored back into the Security Data Lake for use again by all analytics.

Elysium has created some advanced analytics based on context information that stores additional summary information back into the security data lake. For example the Insider Threat Detection algorithm produces a daily summary or “roll-up” table for various aspects of user behavior. Another analytic keeps track of the “first seen” and “last seen” for each process name on a particular host.

Appendix A – High-level Schema Diagram

Both are promising new standards for defining schemas to create new communities for developing transportable analytics.

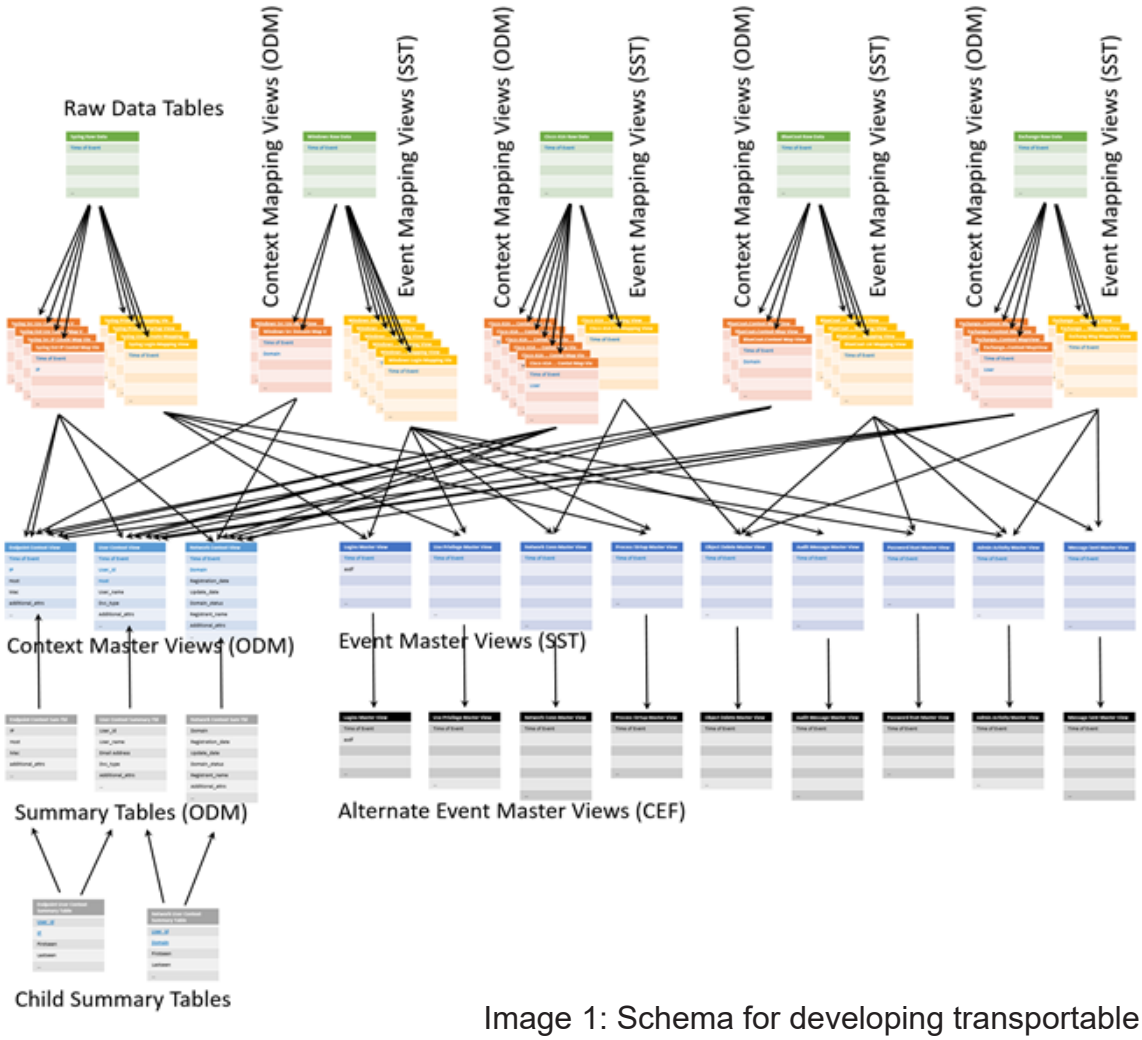

Image 1: Schema for developing transportable analytics

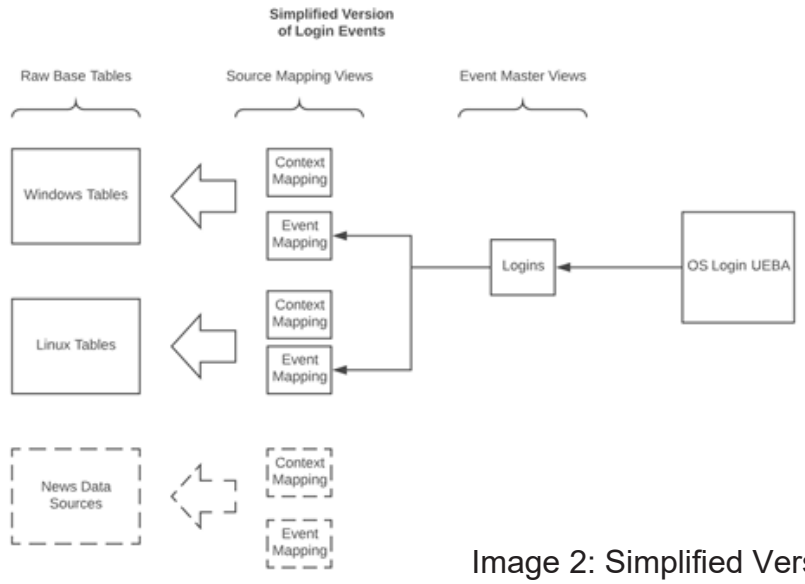

Image 2: Simplified Version of Login Events

For more information contact System Soft Technologies at elysium@sstech.us