Machine Readable Security Incident Notification

PROJECT REPORT 1.0

Hossein Fani

University of New Brunswick  
Fredericton, NB, Canada

hosseinfani@gmail.com

**ABSTRACT**

Mining security incidents helps with better precautionary planning for community safety. However, incident records are expressed in diverse and application dependent formats which impedes common comprehension for automatic knowledge extraction and reasoning. In this project, we provide an infrastructure to transform textual notification of security incidents to machine readable representation in Semantic Web knowledge modeling paradigm. With this regard, our project develop two main components, *Security Incident Service* and *Security Incident Data Mart*. The former consists of I) Natural LanguageLayer (NLL) which extracts temporal and factual information from incident notification text to data objects, II) Semantic Web Layer (SWL) which serializes objects to Web Ontology Language individuals, sio:SecurityIncident in particular, according to Security Incident Ontology[[1]](#footnote-1) (SIO), and the last but not the least, III) Triple Access Layer (TAL) which stores the individuals to the well-known triple store, Virtuoso Universal Server[[2]](#footnote-2). All layers are in compliance with layering architecture and are working within a passive run on service. The later component incorporates a statistical and visual data mart of security incidents distribution to support police and government with decisions, and criminologists with suggestions. This layer plays as the user interface to the system. Java JDK 1.8 and PHP 5.5 are the programming languages for the service and data mart components respectively, NetBeans[[3]](#footnote-3) 8.0 is the Integrated Development Environment (IDE), and Protégé[[4]](#footnote-4) 5.0 is the ontology editor. We use Apache Jena[[5]](#footnote-5) in our SWL and Virtuoso Jena Provider[[6]](#footnote-6) for TAL.

**Categories and Subject Descriptors**

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods - *Representations (procedural and rule-based)*;

**General Terms**

Design, Documentation, Standardization, Security.

**Keywords**

Ontology, Semantic Web, OWL, Security Incident, Machine Readable, Event.

# INTRODUCTION

Environmental health and safety is the utmost priority of any municipal governor. In order to enhance the continual security of a community nearly real-time software systems are developed to monitor the area, record the events and send alarm notifications [1]. For an ounce of prevention is worth a pound of cure, lots of efforts have been put to find security incidents’ root cause, prediction, and prevention by mining huge datasets of records [2]. Unfortunately, these records do not comply with a widely accepted standard in representation which impede the automatic knowledge extraction and reasoning. Local and official security guards or crime detector CCTV cameras [3] express same event with different schemes. Besides, no such schemas are represented explicitly through ontologies. In this project, we devised a novel light-weight domain ontology, Security Incident Ontology (SIO), which is able to describe any kind of security risk from indecent behavior to assault to more adverse crime. Temporal aspects of security incidents are indispensable and SIO links to OWL-Time[[7]](#footnote-7),[[8]](#footnote-8). OWL-Time together with Event [4] ontology are able to address any general events. However, in SIO we specify the Event to security incident types and Agent class to Victim and Subject. SIO is developed in OWL2 with Protégé 5.0 ontology editor. Then, we decide to apply SIO as OWL model for security notifications of Integrated Risk Management (IRM) system at Ryerson University. This way, we provoke SIO capability in security incident representation, incorporate in Linked Data [5], and at a very least, obtain an in depth experience of theory and practice of Semantic Web technologies and engineering.

The remainder of this report is organized as follows. Section 2, constructs event oriented security incident ontology, SIO. Section 3 presents the system architecture for automatically populating security incident instances from Ryerson IRM notifications. Finally, the conclusions are given in section.

# SECURITY INCIDENT ONTOLOGY

## Ontology Engineering

Our ontology editor is Protégé 5.0. The environment not only facilitates reusing other ontologies by importing ontologies as well as its ontology library [6], but also has visualization tools which help an engineer to have big clear picture of ontology compartments. We choose OWL2 for our ontology. Moreover, we either can create a new upper l

evel core ontology for event or reuse the most cited, yet suitable one for our work. We prefer the latter one and adopt Event event:<http://purl.org/NET/c4dm/event.owl>.

## Event Ontology Adoption

Event ontology is based on the view expressed by James F. Allen and George Ferguson in [7] which states that events are primarily linguistic or cognitive in nature and the world contains no events. Events are just certain useful and relevant patterns of world changes. Nonetheless, this ontology has already been proven useful in a wide range of context, due to its simplicity and usability. The ontology has event:Eventat the heart and reuses OWL-Time time:<http://www.w3.org/2006/time#> for temporal predicate event:time to express instantaneous or extended time object along with a temporal algebra, Geo RDF vocabulary geo:<http://www.w3.org/2003/01/geo/wgs84_pos#> for spatial predicate event:place, and foaf:<http://xmlns.com/foaf/0.1/> for event:agents of an event. event:factor is everything used in an event, event:productis whatever produced by an event and event:sub\_event provides a way to split a complex event into simpler ones. Figure 1. These properties are appropriate for modeling any type of event, though they are two general for security incidents and we think further specifications are needed, particularly for agents. For instance, in security incidents we have specific types of incidents e.g. assault, robbery, and so on or agents such as subject, victim, and observer.

Security incident is an event. That simply means:

<owl:Class rdf:ID="SecurityIncident">

<rdfs:subClassOf rdf:resource="#Event" />

</owl:Class>

However, security incident instances investigation reveal other types of incidents. Table 1 are lists of new concepts included in our SIO and Figure 2 shows its inter-relations among ontologies. SIO leverage event:Event class predicate event:*agent* for modeling the sio:Victim and sio:Subject of sio:SecurityIncident. We assume that security incident should have at least one event:agent of type sio:Subject which is owl:subClassOf foaf:Agent indirectly. This entity is the most enriched and has predicates to describe and track the subject in an incident such as. sio:height, sio:weight, sio:hairColor, sio:image, sio:preState, sio:postState. sio:preState explain the subject how he/she start to make a security concern and sio:postState shows the his/her final destiny e.g. flee, arrest, killed. sio:Victim, likewise, has these predicates. It is worth mentioning that on the one hand sio:Subject or sio:Victim may be sio:CommunityMember, and on the other hand there is no is-a or kind-of relationship between them. SIO leave it to the time when we instantiate a security incident. By then, we can add rdf:type to manage the security incident types.

## Reification

The SIO ontology specification and its Turtle version are available at <http://s2-ls3.rnet.ryerson.ca/ontologies/sio.owl>.

# SOFTWARE SYSTEM

While developing SIO, we define software system *Machine Readable Security Incident Notification* to put all the theory of Sematic Web technology into practice and examine our SIO in reality. The academic vision for the final product can be summarized into:

1. Automatic information extraction from text of security incidents report or notification and represent it as way to be able to query it
2. Follow Open Linked Data idea to provide public free access to security incidents happens in Ryerson University urban campus
3. Evolve an ontology for security incidents and provide its competency proof
4. Practice Semantic Web software engineering process
5. Mastering Semantic Web technologies, languages, tools, Api’s, and triple stores

From the business point of view, the product fulfils the following functional requirements:

1. Web based near online dashboard for security incident distribution monitoring and management; Security Incident Management Data Mart
2. Automatic security incident information extraction through textual notifications

And non-functional requirements:

1. Reusable infrastructure for transforming any kind of textual security incident representation to OWL individuals for future automatic reasoning
2. Wise architecture design to separate different responsibilities through layering and components

In this report we explain the whole design and features of the first version. It is trivial that the software is in its early stages and some features are going to develop in the next iteration of software production line.

Table 1. A list of the major concepts and entities (second level) in the SIO ontology version 1.0

|  |  |
| --- | --- |
| **Term** | **Description** |
| Security Incident | The set of connected events which reflects an occurrence, unusual problem, incident, deviation from standard practice, or situation that requires follow-up action. |
| Sexual Assault | A form of sexual violence, is any involuntary sexual act in which a person is threatened, coerced, or forced to engage against their will, or any non-consensual sexual touching of a person |
| Assault with Weapon |  |
| Assault | The direct or indirect application of force to another person, or the attempt or threat to apply force to another person, without that person’s permission |
| Indecent Exposure | Indecent exposure is the deliberate exposure in public or in view of the general public by a person of a portion or portions of his or her body, in circumstances where the exposure is contrary to local moral or other standards of appropriate behavior |
| Voyeurism | Voyeurism is the sexual interest in or practice of spying on people engaged in intimate behaviors, such as undressing, sexual activity, or other actions usually considered to be of a private nature |
| Robbery | The act of taking or another person's property (including attempts) |
| Victim | A victim of a security incident is an identifiable person who has been harmed individually and directly by the suspect, rather than by society as a whole |
| Subject | A subject is a known person who initiate the security incident usually by committing crime. |
| Community Member | Staff, Student, Faculty Member |



Figure 1. A brief snapshot concepts reuse from foaf, geo, wn, time, event, and our sio. Red arrows imply inheritance or owl:subClassOf and the blues are associations.

# Architecture

Machine Readable Security Incident Notification software system has two main component, passive service which does the polling for fetching new security notifications, and web based dashboard which provide a management console to the end user. Figure 2 pictures gracefully the whole initial architecture sketch. Although these two component are completely independent both in build time and runtime, in future they may lose independence in build time due to Triple Access Layer.

## Security Incident Service

Security Incident Service is the core and passive component of the software system. It has a polling part along with three processing layers. In this part we explain the polling structure and further discussion of each layer will be done in their specific sections. All Ryerson Students, Faculty & Staff receive Security Incident notices directly via email. We create automatic forward features of gmail to forward any such notification to a named account [seurityincidentsemanticweb@gmail.com](mailto:seurityincidentsemanticweb@gmail.com). Then, we poll in time intervals (e.g. 120 sec) the account’s email box through IMAP protocol and fetch all the unseen messages. Each message has a link to a webpage which describe the incident fully. We read that page and send it to the Natural Language Layer (NLL) for text analysis. All the settings for gmail account and polling interval are in smtp.properties config file in the service jar file:

Table 2. Security Incident Service Project Properties

|  |  |
| --- | --- |
| **Property** | **Value** |
| Project distribution folder | /SecurityIncidentService/dist |
| Distributed jar file | /SecurityIncidentService/dist/SecurityIncidentService.jar |
| Configuration file path | /SecurityIncidentService/dist/SecurityIncidentService.jar/SecurityIncidentService/smtp.properties |
| Class File | /SecurityIncidentService/dist/SecurityIncidentService.jar/SecurityIncidentService/SecurityIncidentService.class |
| Source Path | /SecurityIncidentService/src/securityincidentservice |

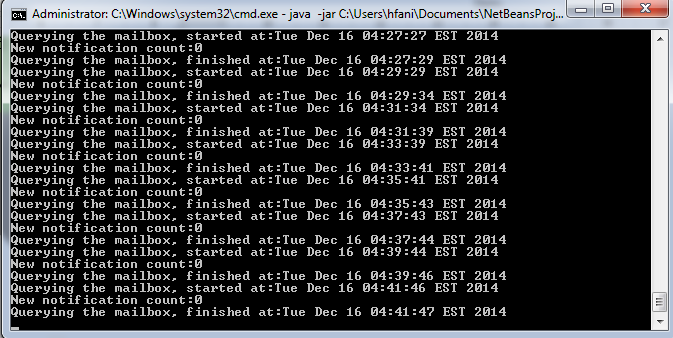


Figure 2. Polling gmail account for new notifications by the service.

### Natural Language Layer[future]

Natural Language Layer is supposed to parse the textual content of the security incident report and extract entities and their property-values along with the co-references. Extracting temporal expression such as incident report date and event date, finding the victim and subject of the event and their associated property-values are the major tasks in this layer. There are api’s for this kind of task among which the Apache OpenNLP[[9]](#footnote-9) and Stanford University CoreNLP[[10]](#footnote-10) are the most cited. However, these libraries working properly in general text corpus and will fail in our domain specific context. Therefore, we should train them (the model) to learn our textual paradigm. For it requires much time, we omit developing this layer for the first version. For now, it populates different security incident objects with randomly generated data. Besides, this layer has reference to the Semantic Web Layer only.



Figure 3. Machine Readable Security Incident Notification software system architecture. The left hemisphere is the software service which polls each 2 minutes a gmail account, [seurityincidentsemanticweb@gmail.com](mailto:seurityincidentsemanticweb@gmail.com), which is subscribed to be notified for any security incident by Ryerson University Integrated Risk Management. After fetching new incidents, the Natural Language Layer parse the text and extract information to an object of type SecurityIncident, defined in Common, and hand it to next layer. The Semantic Web Layer serialize the populated object to OWL individuals and pass them to Triple Access Layer to store them in the Virtuoso triple store. The right hemisphere is the primitive dashboard which shows the incidents geographical distribution filtered by the date of event.

Table 3 Natural Language Layer Project Properties

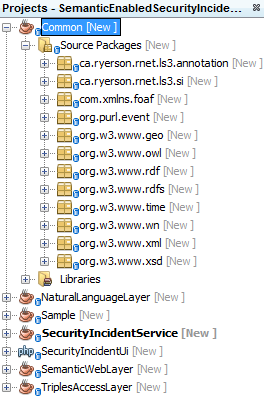
|  |  |
| --- | --- |
| **Property** | **Value** |
| Project distribution folder | /NaturalLanguageLayer/dist |
| Distributed jar file | /NaturalLanguageLayer/dist/NaturalLanguageLayer.jar |
| Configuration file path | n/a |
| Class file | /NaturalLanguageLayer/dist/NaturalLanguageLayer.jar/ca/Ryerson/rnet/ls3/si/nnl/NaturalLanguageLayer.class |
| Source path | NaturalLanguageLayer/src/ca/Ryerson/rnet/ls3/si/nnl |

### Semantic Web Layer

Semantic Web Layer serialize the security incident object from upper layer, Natural Language Layer, to OWL individuals. We, instead of doing serialization by a singleton serializer object, give the responsibility of rendering OWL triple of each object part to the associated object itself. With this regard, we first create a class for each concept of our SIO with the class Thing the Godfather. Then put a method getRdfResource() to render the fundamental RDF triple parts. All the children override this method appropriately to render their customized RDF triples. We put all the class definition for the whole ontology in a separate layer Common and create namespaces according to their OWL IRI’s. For the namespace naming convention in java is different from IRI naming, we use customized annotation for each class to find its IRI prefix. The only thing remains for this layer is only calling the security incident object’s getRdfResource(). It is interesting that we add a field of type List<Class> to simulate the runtime dynamics in multiple type specification of OWL individuals. Then we consider this field values as the types for the object not the this.getClass(). This layer and the Common layer use Apache Jena library extensively for OWL individual generating.

Table 4. Semantic Web Layer Project Properties

|  |  |
| --- | --- |
| **Property** | **Value** |
| Project distribution folder | /SemanticWebLayer/dist |
| Distributed jar file | /SemanticWebLayer/dist/SemanticWebLayer.jar |
| Configuration file path | /SemanticWebLayer/dist/SemanticWebLayer.jar/ca/ryerson/rnet/ls3/si/swl/configuration.properties |
| Class file | /SemanticWebLayer/dist/SemanticWebLayer.jar/ca/ryerson/rnet/ls3/si/swl/SemanticWebLayer.class |
| Source path | /SemanticWebLayer/src/ca/ryerson/rnet/ls3/si/swl/ |



|  |
| --- |
| PREFIX sio: <http://ls3.rnet.ryerson.ca/ontologies/sio/>  PREFIX event: <http://purl.org/NET/c4dm/event.owl#>  PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84\_pos#>  PREFIX time: <http://www.w3.org/2006/time#>  SELECT distinct ?type ?lat ?lng ?year ?month ?day  WHERE  {  GRAPH <http://ls3.rnet.ryerson.ca/SecurityIncident/test>  {  {?securityIncidentId rdf:type sio:Assault} UNION  {?securityIncidentId rdf:type sio:AssaultWithWeapon} UNION  {?securityIncidentId rdf:type sio:SexualAssault} UNION  {?securityIncidentId rdf:type sio:Robbery} UNION  {?securityIncidentId rdf:type sio:Voyeurism} UNION  {?securityIncidentId rdf:type sio:IndecentExposure} UNION  {<p> <p> <o>}  ?securityIncidentId rdf:type ?type.  ?securityIncidentId event:place ?placeId.  ?placeId geo:lat ?lat.  ?placeId geo:long ?lng.  ?securityIncidentId event:time ?timeId.  ?timeId time:inDateTime ?dateTime.  ?dateTime time:year ?year.  ?dateTime time:month ?month.  ?dateTime time:day ?day.  FILTER  (  (?type != owl:NamedIndividual) &&  (  1 = 1  )  )    }  } |

Figure 4. Base query to search different types of security incidents in a date range

Figure 4. Class and namespace definition of SIO

### Triple Access Layer

We choose Virtuoso Universal Server for storing our OWL individuals, for it also provide SPARQL end point. Triple Access Layer use Virtuoso Jena Provider to connect to Virtuoso server endpoint through jdbc and store the generated OWL individuals. This layer should be the only layer which can access our security incidents OWL individuals, not only for saving, but also for querying. Hence, we will not use other endpoints such as the standard SPARQL endpoint for our software system.

Table 5. Triple Access Layer Project Properties

|  |  |
| --- | --- |
| **Property** | **Value** |
| Project distribution folder | /TripleAccessLayer/dist |
| Distributed jar file | /TripleAccessLayer/dist/TripleAccessLayer.jar |
| Configuration file path | /TripleAccessLayer/dist/TripleAccessLayer.jar/ca/Ryerson/rnet/ls3/si/gal/configuration.properties |
| Class file | /TripleAccessLayer/dist/TripleAccessLayer.jar/ca/Ryerson/rnet/ls3/si/gal/TripleAccessLayer.class |
| Source path | /TripleAccessLayer/src/ca/Ryerson/rnet/ls3/si/gal/ |
| Triple store endpoint | jdbc:virtuoso://s2-ls3.rnet.ryerson.ca:1111/charset=UTF-8/log\_enable=2 |
| Graph IRI | <http://ls3.rnet.ryerson.ca/SecurityIncident/test> |
| Virtuoso SPARQL endpoint | http://s2-ls3.rnet.ryerson.ca:8890/sparql |

## Data Mart

We develop a very simple dashboard to our stored security incidents at <http://s2-ls3.rnet.ryerson.ca/sio/> . The page searches all the selected type of security incidents within the date range and locating them geographically in the google map.

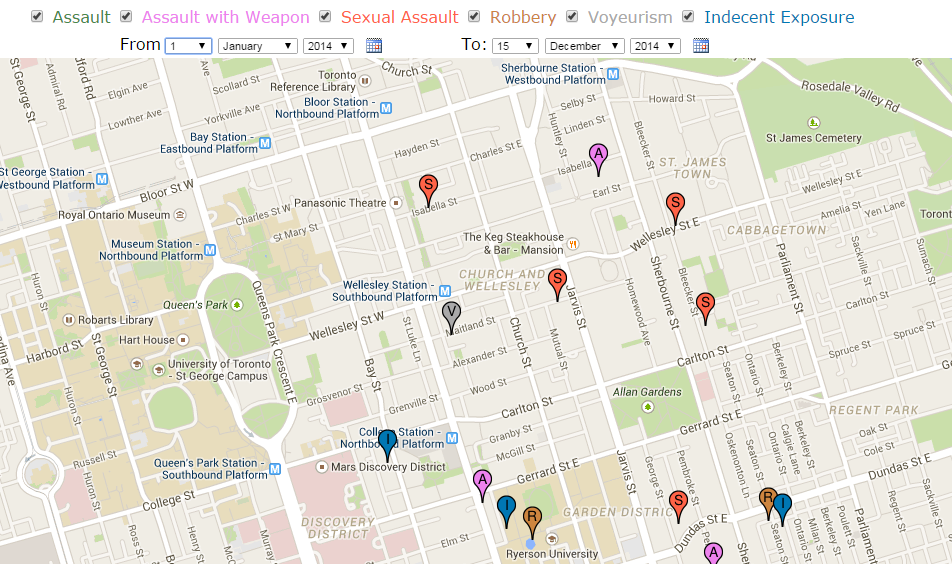


Figure 5. The snapshot of system dashboard at http://s2-ls3.rnet.ryerson.ca/sio/. It locates the incidents geographically with its type color and capital letter. The shown icons are incidents within the 2014 year.

# AKNOLEDGMENT

This work is done as part of “Sematic Web Technologies” course’s project in Laboratory for System, Software, and Semantics (LS3)[[11]](#footnote-11) at Electrical and Computer Engineering department of Ryerson University, ON, Canada. We highly appreciate the course instructor, Dr. Bagheri[[12]](#footnote-12), and the lab members for their precious comments.

REFERENCES

|  |  |
| --- | --- |
| [1] | Ryerson University, "Integrated Risk Management (IRM)," [Online]. Available: http://www.ryerson.ca/irm. [Accessed 17 10 2014]. |
| [2] | Canadian Broadcasting Corporation, "Toronto Crime by Neighbourhood," [Online]. Available: http://www.cbc.ca/toronto/features/crimemap/. [Accessed 17 10 2014]. |
| [3] | BRS Labs, "AI Sight," BRS Labs, [Online]. Available: http://www.brslabs.com/index.html#aisight. [Accessed 17 10 2014]. |
| [4] | Y. Raimond and S. Abdallah, "The Event Ontology," Centre for Digital Music, Queen Mary, University of London, [Online]. Available: http://motools.sourceforge.net/event/event.html. [Accessed 17 10 2014]. |
| [5] | T. Heath, "Linked Data - Connect Distributed Data across the Web," Linked Data community, [Online]. Available: http://linkeddata.org/. [Accessed 17 10 2014]. |
| [6] | Stanford Center for Biomedical Informatics Research, "Protege Ontology Library," Stanford Center for Biomedical Informatics Research, [Online]. Available: http://protegewiki.stanford.edu/wiki/Protege\_Ontology\_Library. [Accessed 17 10 2014]. |
| [7] | J. F. Allen and G. Ferguson, "Actions and Events in Interval Temporal Logic," *Journal of Logic and Computation,* vol. 4, pp. 531--579, 1994. |

1. http://s2-ls3.rnet.ryerson.ca/ontologies/sio.owl [↑](#footnote-ref-1)
2. http://virtuoso.openlinksw.com/ [↑](#footnote-ref-2)
3. https://netbeans.org/ [↑](#footnote-ref-3)
4. http://protege.stanford.edu/products.php#desktop-protege [↑](#footnote-ref-4)
5. https://jena.apache.org/ [↑](#footnote-ref-5)
6. http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtJenaProvider [↑](#footnote-ref-6)
7. http://www.w3.org/TR/owl-time/ [↑](#footnote-ref-7)
8. OWL-Time is still in W3C First Public Working Draft (FPWD) state since 27 September 2006 and is not expected to become a W3C Recommendation. [↑](#footnote-ref-8)
9. http://opennlp.apache.org [↑](#footnote-ref-9)
10. http://nlp.stanford.edu [↑](#footnote-ref-10)
11. ls3.rnet.ryerson.ca/ [↑](#footnote-ref-11)
12. http://www.ee.ryerson.ca/~bagheri/ [↑](#footnote-ref-12)