Attack Pattern Ontology: A Common Language for Cyber Security Information Sharing

Yiwen Zhu

Faculty of Technology, Policy and Management, Delft University of Technology, Netherlands

August 2015

Abstract

Cyber attack nowadays is increasingly being reported. Defenders need a good understanding of attacker's perspective in order to accurately anticipate threats and effectively mitigate attacks. This understanding can be obtained through sharing attack pattern. However, in the existing researches the consideration about information sharing is not integrated into the attack pattern concept. In this paper, we propose an attack pattern ontology as a common language of information sharing; the goal is to demonstrate how this ontology may effectively support cyber security information sharing. Based on the existing theories about attack pattern, we developed an ontological model to present attack information. The research can be further developed to integrate attacker profile ontology with the attack pattern ontology, which enables more systematic analysis of cyber attacks.

Keywords: *Cyber security; Information sharing; Cyber attack; Attack pattern; Ontology*

1. Introduction

Cyber security is the act of protecting information and communication technology systems and their contents (Fischer, 2014). It is intertwined with everyone's daily life; citizens, businesses and government bodies are using the Internet for interactions, collaboration and communication (The Minister of Security and Justice, 2013). However, as a consequence of the convenience brought by connectivity, Internet and Internet users are vulnerable to cyber attack (Geers, 2011). Cyber attack nowadays is growing not only in frequency but also in scale scope and complexity (Johnson *et al.,* 2014). The complexity and size of system increase while the number and the skill level of attackers continues to grow (Barnum & Sethi, 2007).

Sharing understanding about attacker's perspective may be effective for cyber security. Organisations must protect every vulnerability to secure a system; yet, to attack a system, attackers only need to find a single vulnerability (Barnum & Sethi, 2007). Thus, in order to accurately anticipate threats and effectively mitigate attacks, people must have a good understanding of attacker's perspective and their approaches (Barnum & Sethi, 2007; Hoglund & McGraw, 2004). Attack pattern is such a means that it captures attacker's perspective and facilitates early mitigation of potential attacks (Moore et al., 2001; Barnum & Sethi, 2007; Fernandez et al., 2007; Uzunov & Fernandez, 2014). Attack pattern represents commonly occurred attacks and reuses attack information (Schaeffer-Filho & Hutchison, 2014), which can help enhancing security system design and evaluation to prevent a variety of attacks (Fernandez, VanHilst, Petrie, & Huang, 2006; Schumacher *et al.,* 2013). In order to gain objective and generic understanding of attacker's perspective, attack pattern could be shared. The information of attack pattern can be modelled in various ways; in this paper, we propose ontology to model the shared information. Compared with ontological data model, non-ontological approach systems face a number of challenges:

1.1 Ontology-based vs. non-ontology based approach

Ontology is a semantic web model to provide a common language of a domain of knowledge that is exchanged and shared (Uschold & Gruninger, 1996; Noy & McGuinness, 2001). It gives a description of entities and their properties, relationships, constraints (Gruninger & Fox, 1995). Some mainstream non ontological approaches to modelling data are taxonomy and database schema; these approaches are not suitable for communication between complex systems. Taxonomy describes and classifies resources based on hierarchical relationships among entities; but it cannot provide contextual information or rich meaning of these concepts to further define restrictions and interdependencies among concepts (Kim et al., 2005). Moreover, because hierarchical relation is the only kind of relation that connect elements in taxonomy, taxonomy is not able to provide class-based reasoning such as automatically classification. When adopting database schema, new primary keys are necessary that primary keys in two different databases cannot be synchronised. For example, two countermeasure datasets can be linked through the 'Measurement_id' primary key, but these IDs refer to different countermeasures. So only the chosen data can be shared with the new primary keys; if a third database is added, the primary keys need to be defined again ('Tutorial 3', n.d.).

An ontological model can express and interpret the meaning behind the data through detailed definition and description. Thus in our attack pattern ontology, all information about attack pattern is related; people can find information via the linked standard terminology without even knowing the existence of the information ('Tutorial 3', n.d.). Moreover, this happens without the need for transformation or mapping between the two sites; it is all settled through semantics ('Tutorial 3', n.d.). An ontology helps the integration of information from different sources with the least deviation from the origin semantics. To summarise, compared with non-ontological approaches, an ontology approach has the following features:

- Different types of relations can be added between any two elements (compared with the sole hierarchical relation in taxonomy and the table-to-table connection in database)
- Reasoning and automatically classification.
- Semantics can be added to data for further specification.

1.2 Problem and proposed answer

The aim of this research is to develop a common language that uses attack pattern as the carrier of data in information sharing, rather than treat it only as an approach to record public knowledge or present particular types of attack. Although the ultimate goal of sharing attack pattern may be same to the goal of presenting public knowledge in attack pattern, the process of sharing involves interactions between organisations and it causes integration of knowledge, consensus on the shared information and unexpected inspiration. We propose an ontological model as the base language to share attack pattern.

The proposed ontology of attack pattern provides a powerful construct to enhance cyber security. It captures the context of important cyber attacks, the techniques used by attackers, impact on the victim system, suggested or applied countermeasures, etc. We review articles about attack pattern as the start of developing this ontology. The main contents of the ontology are based on these articles. However the existing researches talked more about reusing attack pattern but not how to share it; taking into consideration of the goal of sharing information, we adapt the existing concept of attack pattern for our different purpose.

This paper has additional four sections. In the next section, related works are presented. Literatures about attack pattern are the focus and characteristics of pattern are also mentioned. The third section introduces some popular ontology development methodologies and points out the methodologies used in this paper. In the fourth section a study is done towards sharing attack pattern and the ontological model is developed as the base language of sharing. The findings of this study are brought together and analysed in the fifth section. Section five also discusses the limitations of the research and concludes the research.

2. Related work

Before analysing attack pattern, the details of pattern should be explained. The Merriam-Webster English Dictionary defines pattern as, 'a repeated form or design especially that is used to decorate something; the regular and repeated way in which something happens or is done; something that happens in a regular and repeated way'. From these definitions, we can extract the core concept of pattern that patterns are 'regular' and 'repeated'; what has been captured in the pattern today can happen again tomorrow. Therefore people build patterns to encapsulate and reuse knowledge (Schaeffer-Filho & Hutchison, 2014; Uzunov & Fernandez, 2014).

In computer science, patterns are general, reusable solutions to commonly occurring problems (Bayley, 2014). Pattern has been found useful in diverse areas including software engineering, where this concept has received much attention both in academia and industry (Uzunov & Fernandez, 2014). The idea of pattern was originated in architecture from Christopher Alexander's architectural patterns for architecture design (Alexander, Ishikawa, & Silverstein, 1977). Then, it was transferred to software design as design patterns in the book Design Patterns: Elements of Reusable Object-Oriented Software (Gamma, Helm, Johnson, & Vlissides, 1995). Following design pattern, attack pattern and security pattern were also introduced to the cyber security domain (Bayley, 2014).

2.1 Attack pattern in theories

Our work benefits from literature on attack pattern, an emerging research topic that focuses on attack modelling for cyber security. In 2001, the term attack pattern was introduced in the paper

Attack Modeling for Information Security and Survivability (Moore et al., 2001). Three years later, it was extended and enriched in greater detail and with a solid set of specific examples in the book Exploiting Software: How to Break Code (Hoglund & McGraw, 2004). Since then, several individuals and groups have tried to push the concept forward (Barnum & Sethi, 2007).

We reviewed twelve articles that introduced the concept of attack pattern (Moore *et al.,* 2001; Hoglund & McGraw, 2004; Barnum & Sethi, 2007; Fernandez *et al.,* 2007; Gegick & Williams, 2005; Gegick & Williams, 2007; Thonnard & Dacier, 2008; Zhu, 2011; Blackwell, 2012; Huang et al., 2013; Uzunov & Fernandez, 2014; Bayley, 2014). Most of the articles defined attack pattern as a generic representation of attacks from the point of view of attacker that it presents the critical features of the attack. However, the works of Thonnard & Dacier (2008), Zhu (2011) and Huang et al. (2013) are describing something different. The attack pattern defined by Thonnard & Dacier (2008) is a time signature, which is a time series (figure based) that show the 'aggregated source count for a given type of attack'. This definition describes a pattern of number of attacks on the scale of time, but not a pattern of attacks. Zhu (2011) defines attack pattern as a sequence of attacks; it presents only the steps and execution flow of an attack, which is the same from the attacker's perspective and the defender's perspective. Huang et al. (2013) also relate attack with time; they define attack pattern as the regularity of time intervals between attacks.

Four of the review articles have given the specific information captured by attack pattern (Moore *et al.,* 2001; Barnum & Sethi, 2007; Fernandez *et al.,* 2007; Blackwell, 2012). Moore *et al.* (2001) proposed the simplest template that one attack pattern captures only five attributes: name, goal, precondition, attack and postcondition. Blackwell (2012) provided the most complex template with more than ten attributes: name, perpetrator (who), motivation (why), intent (what), target (to what), security or immediate impact, security or ultimate impact, execution (how), process diagram, methods (with what), context or prerequisites (when), resources (with what), attacker skill (internal with what), attacker knowledge (know what) and reference. We can find the corresponding relation between most of the elements from different articles. For instance, the three elements from Blackwell (2012):execution (how), process diagram and methods (with what) can corresponding to the one element from Moore *et al.* (2001): attack.

Only Barnum & Sethi (2007) described the process of generating attack pattern; other articles just simply provided attack pattern examples to show how the templates work. According to Barnum & Sethi (2007), when one particular attack is being reported many times and not matches the existing attack pattern (public knowledge), people can discover the cause of the attack and build a new attack pattern to describe such type of attacks.

Some articles also provided the application of attack pattern. Attack pattern can educate people about common attacks that make future threat modelling tasks easier (Uzunov & Fernandez, 2014). It can be used for identifying the potential vulnerabilities and attacks that are applicable to one system (Moore *et al.,* 2001; Gegick & Williams, 2007), which then guide the design process of a system and support the judgements about possible design solutions (Barnum & Sethi, 2007; Faily *et al.,* 2012). It can also be seen as data source to find evidence of attacks (Fernandez *et al.,* 2007).

Nevertheless, the articles mentioned above treated attack pattern only as an approach to present knowledge, but did not think about the possible of using it as a means to share information. Hence the existing researches did not take the following aspects into consideration that are necessary for the task of sharing attack pattern (Harrison & White, 2012):

- How to preserve the privacy of information provider.
- How to optimise the attack pattern contents and format to make it useful for information consumer.
- How to model the data and maintain the compatibility when the amount of participants as well as the database size keeps growing.

2.2 Attack pattern in practice: CAPEC

CAPEC (Common Attack Pattern Enumeration and Classification) is an open data resource that provides a comprehensive dictionary of known attacks (Mitre Corporation, n.d.). It aims at identifying and understanding attacks, which is more focused for academia (WASC, 2010; Mitre Corporation, n.d.). CAPEC adopts several perspectives to present attacks: by hierarchical representation, by relationship to external factors, by relationship to specific attributes. The hierarchical representation, which includes two logics (16 mechanisms of attack and 6 domains of attack), is the main navigation method to the CAPEC dictionary; it covers most of the attack pattern records (category 286 and its subclasses are excluded) and is the only perspective showed on the home page of CAPEC. However, these two logics are parallel and not connected; they can be presented as two separate attack trees with the same root of cyber attack. In CAPEC, attack patterns have 3 different completeness levels: hook, stub, complete; and 3 different abstraction levels: meta, standard, detailed (Mitre Corporation, 2014). The abstraction level shows a hierarchical structure of attack pattern that the hierarchy often starts with a category, followed by a standard or meta attack pattern and ends with a detailed attack pattern (Mitre Corporation, n.d.a).

CAPEC has the following disadvantages in presenting shared knowledge:

- The classification of attack pattern is disorganised and not mutually exclusive. For example, CAPEC-13 is the child of both the 'Exploitation of Authorization' and the 'Manipulate Resources' mechanisms of attack, but it is not the child of any of the six domains of attack.
- The abstraction level is not implemented as designed following the sequence of meta attack pattern, standard attack pattern, detailed attack pattern. For instance, attack pattern in the social engineering category and the physical security category are all in the meta level; only one detailed attack pattern exist in the supply chain category, others are all in the standard level; attack pattern in the software category are in an unorganised status that a standard pattern (CAPEC-20) can has a meta pattern child (CAPEC-97).
- A top-down approach is used to produce attack pattern rather than focusing on gathering evidence from multiple datasets to identify pattern (Schaeffer-Filho & Hutchison, 2014). Thus several classes are too broach such as 'abuse of functionality' while the next level

classes would be too specific, for example ‘WSDL scanning’ that applies only to system based on web-services (Uzunov & Fernandez, 2014).

3. Ontology development methodology

In the process of developing our attack pattern ontology, we will avoid the shortcomings and fill in the gaps of the existing works.

There are various ontology development methodologies within different disciplines. Caracciolo (2006) designs a methodology to build an ontology for logic and linguistics. This work integrates a set of hierarchical relations with two non-hierarchical relations to enable an explicit navigation (Caracciolo, 2006). Ontology Development 101 is a guide to create ontologies for beginners where a process of 7 steps is introduced and an ontology of wine and food is developed (Noy & McGuinness, 2001). METHONTOLOGY is a chemical ontology building methodology that focuses on the reuse of ontologies (López, Gómez-Pérez, Sierra, & Sierra, 1999). According to METHONTOLOGY, most of the evaluation work of the ontology should be carried out in the conceptualisation stage to prevent errors in implementation. Uschold & Gruninger (1996) introduced the principles and methods of developing ontologies for knowledge engineering. The article intends to introduce the design and use of ontology as a shared understanding to improve communication among people, organisations and software systems (Uschold & Gruninger, 1996).

In this paper, we apply the combination of the ontology development methodology introduced by Noy & McGuinness (2001) and Uschold & Gruninger (1996). Noy & McGuinness (2001) has detailed explanation of what to do in each step whereas it omitted some major steps mentioned by Uschold & Gruninger (1996) including the choice of ontology language and the evaluation. Therefore we mainly follow the process in Ontology Development 101 but add the steps of choosing ontology language and tool: 1) define the ontology scope and purpose, 2) enumerate key concepts, 3) define classes and properties, 4) define facets of properties, 5) create instances, 6) choose a representation language, 7) evaluate the ontology.

The shortcomings of existing works about attack pattern will be overcame through the seven ontology development steps. In step 2 and 3, we try to keep the necessary concepts and leave out redundant concepts, which deals with the needs of preserving privacy of information provider and making attack pattern contents useful for information consumer. In step 3, we integrate top-down and bottom-up methods to build a structure of the main concepts where the disadvantages of CAPEC can be avoided; we only generate one structure to classify attack pattern and this structure has pre-defined hierarchy that all users can follow the same classification system. In step 5, we choose an expressive ontology language that is capable of presenting the ontology and data.

We will implement the ontology in Protégé, which is a free, open-source tool with intuitive user interface. Protégé is written in Java, thus supports running in a wide range of operating systems (Noy et al., 2003). It allows users to create and edit ontologies in an application area (Noy et al., 2003) and it has the building blocks that we expect in developing an ontology: classes, relations and instances. Protégé can record ontologies in various formats including RDF/XML, OWL/XML, N-Triples, N3 and Turtle RDF.

4. Development of attack pattern ontology

Following the 7 step ontology development process and taking into consideration the shortcomings of the current researches and CAPEC, we develop an attack pattern ontology as described below:

4.1 Define the ontology scope and purpose

The ontology will be used as the base language and to be shared between organisations within an attack information sharing community for the purpose of gathering collective intelligence.

4.2 Enumerate key concepts

After extracting key concepts from the definition of attack pattern that shared by most of the articles we reviewed, we got a list of key concepts: attack, vulnerability, method, knowledge required by attacker, target system, countermeasures. All these concepts can be found in the attack pattern templates of Moore *et al.* (2001), Barnum & Sethi (2007), Fernandez *et al.* (2007) and Blackwell (2012), which proved the reasonability of these templates. We took an union of the concepts from these four templates and ended up with a new version of attack pattern template.

In order to overcome the three shortcomings of the existing articles about attack pattern, we made the following changes to the template:

- For the purpose of preserving the privacy of information provider, we deleted 'attack' or 'known uses' from the template
- For the purpose of making attack pattern contents useful, we added 'number of occurred attacks' and 'typical severity' of attack to give more information consumers more senses about the type of attack

Table 1 Key concept

Concept	Enumeration
Name	Brief descriptive name of the pattern
ID	Unique integer identifier of the pattern
Attack prerequisites	The condition or characteristics of the target system much has or behaves in order for such type of attacks to happen
Resources required	Resources required by an attacker to execute this type of attack, such as CPU cycles, IP addresses, tools, etc.
Attacker skill or Knowledge required	Level of skill or knowledge required
	Skill or knowledge required by an attacker to execute this type of attack
Related vulnerabilities or weaknesses	The mistake or improperness that can be used to perform an attack
Method of attack	The mechanism used by this type of attack
Attack target	The targeted information asset

Solutions and Mitigations	Approaches that can prevent or mitigate the attack
Description	The steps for an attacker to execute the typical flow of the attack
Attack consequence	The technical result achieved by the attack
Related patterns	Other attack pattern instances relate to, dependent on, chained together, etc. with this pattern
Status	The progress, version of the pattern
Number of occurred attacks	The happened times of this type of attack within a fixed time period (per month, per year, etc.)
Typical severity	The severity of the impact to the target system if the attack occurs

4.3 Define classes and properties

Protégé allows three types of properties: object property, data property, annotation property (Horridge et al., 2011). Both name and ID could be data properties. Description could be annotation property because it is not comparable and contains large amount of text. Other entities in Table 1 are comparable thus we will set them as classes. One object property and one inverse object property will be added to link class pairs. We can name them in the form hasProperty and isPropertyOf (Horridge et al., 2011). For example, one attack pattern A hasConsequence C, thus consequence C isConsequenceOf A.

In order to overcome the structural shortcomings of CAPEC, we use a combination method of bottom-up and top-down method to build the class hierarchy. It will end up with only one hierarchy thus the classification of attack pattern will be mutually exclusive. Furthermore, the bottom-up method focuses on tailoring the structure for attack pattern instead of targeting nowhere. The bottom level concepts are distilled from 15 CAPEC (Common Attack Pattern Enumeration and Classification) attack pattern instances, which are from different attack domain and different mechanisms of attacks. CAPEC attack pattern instances are built by experts and 'far more comprehensive than anything online' (WASC, 2010), therefore we believe in the quality and the comprehensiveness of these attack pattern instances. The hierarchy is presented in Appendix A.

4.4 Define facets of properties

Facets of properties are used to further restrict properties. Here are some common restrictions of properties (Noy & McGuinness, 2001; Horridge et al., 2011):

- The number of the values - cardinality restrictions
  - Min: the cardinality restrictions describe the class of individuals that have at least a specified number of relationships with other individuals or data values
  - Max
  - Exactly
- The relationships an individual participate in- quantifier restrictions

  - Only: the universal restriction, also known as ‘allValuesFrom’, means the set of individuals only (∀) has relationships with an individual of a specified class; it is important to point out that the universal restrictions do not guarantee the existence of a relationship for a given property
  - Some: the existential restriction, also known as ‘someValuesFrom’, means the set of individuals has at least one (∃) relationship with an individual of a specified class
- The relationship an individual link to another specific individual - hasValue restrictions (∋)
- Data property value type
  - *String* is the simplest value type, which stores a sequence of elements. It is used for slots such as name.
  - *Number* (Float, Integer, etc.) describes properties with numeric values.
  - *Boolean* properties are simple yes or no (‘true’ or ‘false’) flags.
  - *Enumerated* properties specify a list of values allowed.

According to the relation from the instances in the domain class to the instances in the range class (or data), we define the facets of the properties in the following three tables:

Table 2 Data property characteristic

Domain	Data properties	Range	Characteristic
Attack pattern	hasName	String	Functional
Attack pattern	hasID	Integer	Functional
Attack pattern	hasNumberOfOccurredAttacks	Integer	Functional

Table 3 Annotation property restriction and characteristic

Domain	Annotation properties	Range
Attack pattern	Execution flow	Literal
Attack pattern	Status	String
Attack pattern	Version	String

Table 4 Object property restriction and characteristic

Domain	Property restriction	Property name	Range
Attack pattern	Only	hasRelatedPattern	Attack pattern
Attack pattern	Only Some	hasTarget	Target
Target	Some	isTargetOf	Attack pattern
Target	Only	hasVulnerabilities	Vulnerability
Vulnerability	Only	isVulnerabilitiesOf	Target
Attack pattern	Some	hasPrerequisites	Prerequisite
Prerequisite	Some	isPrerequisitesOf	Attack pattern
Attack pattern	Only Some	hasResourceRequired	Resources required

Resources required	Some	isResourceRequiredOf	Attack pattern
Attack pattern	Only Some	hasSkillOrKnowledgeRequired	Skill or knowledge required
Skill or knowledge required	Some	isSkillOrKnowledgeRequiredOf	Attack pattern
Attack pattern	Only Some	hasSkillOrKnowledgeLevel	Skill or knowledge level
Skill or knowledge level	Some	isSkillOrKnowledgeLevelOf	Attack pattern
Attack pattern	Only Some	exploit	Vulnerability
Vulnerability	Some	isExploitedBy	Attack pattern
Attack pattern	Only Some	employ	Method
Method	Some	isEmployedBy	Attack pattern
Attack pattern	Only Some	hasConsequences	Consequence
Consequence	Some	isConsequenceOf	Attack pattern
Attack pattern	Only Some	isWorkedAgainst	Countermeasure
Countermeasure	Some	workAgainst	Attack pattern
Vulnerability	Some	isWorkedAgainst	Countermeasure
Countermeasure	Some	workAgainst	Vulnerability
Attack pattern	Only Some	hasTypicalSeverity	TypicalSeverity
Typical Severity	Some	isTypicalSeverityOf	Attack Pattern
Attack pattern	Only Some	hasTypicalLikelihoodOfExploit	Typical Likelihood of Exploit
Typical Likelihood of Exploit	Some	isTypicalLikelihoodOfExploitOf	Attack Pattern

4.5 Create instance

We produce an attack pattern instance for a payment card data breach incident of the retail industry. The ontology will be used in similar ways for other industries. We chose this incident because it caught lots of attention and we can get access to the details through case studies. But still, some technical details are not available even from these case studies. Note that the original information of cyber attacks is not accessible to external individuals or organisations; the cases we referred to are exception of this condition.

Table 5 Attack pattern instance

Attack pattern: POS Intrusion (domain)
Execution/description: - Attacker(s) search how the victim company interact with its vendors - An email containing malware was sent to a vendor to get the credentials to an online vendor portal. - Get access to the victim's system via the vendor portal and further infiltrate the network - Install malware on point of sale system. The malware gather payment card information as cards were swiped. - Data was sent to a shared central repository; default user account name and password for an IT

management software suite were used to log in to the shared drive - Data was moved to drop locations on hacked servers via FTP and sold on black market			
Property	Individual or data (range)	Class path (range)	Annotation
hasName	POS Intrusion	n/a	
hasID	100	n/a	
hasNumberOfOccurredAttacks	1	n/a	
hasTarget	Point of sale machine	-Target -Network -Application -Client	
hasPrerequisites	Access to a vendor portal	-Prerequisites -Access to the target -Remote access	
hasResourceRequired	No specific resource required	-Resource required - No specific resource required	
hasSkillOrKnowledgeRequired	Find vulnerability of the vendor portal	-Skill or knowledge required -Skill of investigating system feature	
	Create virus scanner undetectable malware	-Skill or knowledge required - Knowledge and skill of specific software	
hasSkillOrKnowledgeLevel	High	-Skill or knowledge level -High	
exploit	Email vendors	-Vulnerability -Social engineering	Attackers may did a Google search that results in a great deal of information of how the victim interacts with vendors.
	Vulnerability found by common network tools	- Vulnerability - Misconfiguration	Technical details unclear
employ	Software installed on POS system	- Method - Installed malware - Spyware	
isWorkedAgainst	Remove the malware from the network	- Countermeasure - Reduce the negative effect or probability of the attack	The victim was initially informed by external organisation about the suspicious activities
		-Countermeasure -Implementation	
hasConsequences	Disclose guest	-Consequence	

	payment card data	-Information disclosure	
	Disclose personal data	-Consequence -Information disclosure	Includes name, mailing address, phone number or email address
hasTypicalSeverity	High	High	
hasTypicalLikelihoodOfExploit	Low	Low	

Source: adapted from Protecting Consumer Information (2014); Radichel (2014); Tipton & Choi (2014)

4.6 Choose a representation language

We choose OWL (web ontology language) as the formal language to encode the ontology for the following reasons:

- OWL language is designed for the need of information processing that it facilitates greater machine interpretability than other languages such as XML and RDF (Heflin, 2009).
- OWL language has been widespread and used for the vast majority of ontologies (Vrandečić, 2009).
- OWL is a formal syntax for defining ontologies ('Tutorial 3', n.d.), which is recommended and fully supported by Protégé. Besides, the Protégé using guide is written for leading users to build OWL ontologies.

4.7 Evaluation

We invite five master students of Delft University of Technology to use the attack pattern ontology in Protégé to build attack pattern and do queries. They gave the following feedbacks on using this attack pattern ontology to present cyber attack:

- After an initial introduction, the participants can understand easily what is the attack pattern for, what is the relationship between an attack and an attack pattern, what does the key concepts of the attack pattern mean
- It is easy to recognise the required elements from the attack descriptions, the attack pattern can be built very fast.
- For some glossaries of the cyber security domain, participants have difficulties to understand them. However based on the context descriptions, they can make decisions on what is attack method, what is attack consequence, etc.
- If any suggestions would be made on the redundancy or shortage of the attack pattern attributes, a long-term experience (few months) on using the attack pattern structure is necessary
- The names of the entities (all or some) are hard to understand
- The ontology might need to be adjusted based on the condition of each industries

Some suggestions were given by the students to improve the ontology:

- Some entities under the class of vulnerability and under the class of network are unclear that the users do not know what are those entities

- Classes with levels (high, medium, low) do not give benchmark to users that which levels should be chosen
- Attack consequence could be expended and be more detailed

For the problems of having difficulties on understanding entity names, we propose these two improvements:

- Adding annotations to define the classes and explain their usages
- Adding more subclasses, i.e. increase the depth of the hierarchy, to provide sense of what kind of events or objects belongs to one class

5. Conclusion

This research provides a solution to use attack pattern for information sharing. An attack pattern ontology is developed as the base language, which has several advantages over other kinds of data models including automatically classification and reasoning. The delivered ontology has the following differences to the current studies:

- In order to preserve the privacy of information provider, our attack pattern ontology will not include the attribute of known uses or attacks
- In order to make the attack pattern more useful for information consumer, we added number of attacks and severity of attacks to the ontology; the number of attacks can make up the loss caused by the absence of known uses
- In order to make the ontology easily to be used, we tried to avoid the shortcomings of the CAPEC structure and applied the combination method of top-down and bottom-up method to build one single structure
- In order to maintain the compatibility of the ontology, we chosen an popular and expressive ontology language OWL

This research has some limitations on the methodology as well as the deliverable. Firstly, professionals were not invited in the qualitative evaluation. The identities of the questionnaire participants were all students, which has both advantages and disadvantages in itself. Advantage is that participants can easily point out the parts that they do not understand. However, it is hard to judge the reason they do not understand is lack of certain background or not. If IT professionals and cyber security experts were also invited for the questionnaire, they might have provided some keen points on the organizational fitness criteria. The evaluation results and the experiences on use cases may not deviate much from the current results, because the difference between students and professionals is their experience in the cyber security area.

Secondly, the ontology user group is limited to well-educated people. The evaluation questionnaire participants could be master students and cyber security professionals. These participants are also the potentials users of the ontology, they can be clustered as well-educated people. Nevertheless, the ontology contains many glossaries of the cyber security domain, which may cause difficulties in using; compared with other people, well-educated people have better knowledge background in order to use it.

Finally, the ontology may lead to confusion if users do not make agreement on the usage of it beforehand. Because of the generosity of the ontological model, all the classes remain at conceptual and high level classification of information, which enable different versions of interpretations. Users can have their own ways of using this ontology that different to our expectation, however they need to agree on it before they can start to use it. Furthermore, users are expected to add new classes and instances into the ontology according to the circumstance of each sharing community; they have to build rules in advance such as how to make decision on what to add. Therefore, this ontology usage is limited that users must make their own rules before using it.

In future researches, the attack pattern ontology can be combined with attacker information ontology. This will lead to a more powerful attack model; attack pattern captures the attacker's perspective and attacker profile records the corresponding abilities of each attackers. For example, one attack pattern presents the attack skill needed and resource required, the attacker profile match each attacker with their skills and resources, people can use the combination of the two to check which attackers are capable of employing this type of attack.

Reference

Barnum S. (2008). *Common attack pattern enumeration and classification (CAPEC) schema description.* Retrieved from https://capec.mitre.org/documents/documentation/CAPEC_Schema_Description_v1.3.pdf

Barnum, S., & Sethi, A. (2007). Attack patterns as a knowledge resource for building secure software. In *OMG Software Assurance Workshop: Cigital.*

Blackwell, C. (2012). A Strategy for Formalizing Attack Patterns. *Cyberpatterns 2012*, 35.

Caracciolo, C. (2006). Designing and implementing an ontology for logic and linguistics. *Literary and linguistic computing*, *21*(suppl 1), 29-39.

Fernandez, E. B., VanHilst, M., Petrie, M. M. L., & Huang, S. (2006). Defining security requirements through misuse actions. In *Advanced Software Engineering: Expanding the Frontiers of Software Technology* (pp. 123-137). Springer US.

Fernandez, E., Pelaez, J., & Larrondo-Petrie, M. (2007). Attack patterns: A new forensic and design tool. In *Advances in Digital Forensics III* (pp. 345-357). Springer New York.

Fischer, E., A. (2014). *Cybersecurity Issues and Challenges: In Brief.* Retrieved from https://www.fas.org/sgp/crs/misc/R43831.pdf

Gal-Or, E. & Ghose, A. (2004). The economic consequences of sharing security information. In Economics of information security (pp. 95-104). Springer US.

Geers, K. (2011). *Strategic cyber security*. Kenneth Geers.

Gegick, M., & Williams, L. (2007). On the design of more secure software-intensive systems by use of attack patterns. *Information and Software Technology*, *49*(4), 381-397.

GFI. (2009). *Targeted cyber attacks.* Retrieved from GFI: http://www.gfi.com/whitepapers/cyber-attacks.pdf

Grüninger, M., & Fox, M. S. (1995). Methodology for the Design and Evaluation of Ontologies.

Harrison, K., & White, G. (2012, November). Information sharing requirements and framework needed for community cyber incident detection and response. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for* (pp. 463-469). IEEE.

Hoglund, G. & McGraw, G. (2004, August). Exploiting Software: How to Break Code. In *Invited Talk, Usenix Security Symposium, San Diego*.

Huang, J. Y., Liao, I. E., Chung, Y. F., & Chen, K. T. (2013). Shielding wireless sensor network using Markovian intrusion detection system with attack pattern mining. *Information Sciences*, *231*, 32-44.

Johnson, C., Badger, L., Waltermire, D. (2014). Guide to Cyber Threat Information Sharing (Draft). *NIST special publication, 800-150*.

Kim, A., Luo, J., & Kang, M. (2005). *Security ontology for annotating resources* (pp. 1483-1499). Springer Berlin Heidelberg.

López, M. F., Gómez-Pérez, A., Sierra, J. P., & Sierra, A. P. (1999). Building a chemical ontology using methontology and the ontology design environment. *IEEE intelligent Systems*, *14*(1), 37-46.

Microsoft. (2012, June). Determined Adversaries and Targeted Attacks. Retrieved from Microsoft: http://www.microsoft.com/en-us/download/details.aspx?id=34793

Mitre Corporation. (n.d.). *About CAPEC.* Retrieved from https://capec.mitre.org/about/index.html

Moore, A. P., Ellison, R. J., & Linger, R. C. (2001). *Attack modeling for information security and survivability* (No. CMU-SEI-2001-TN-001). Carnegie Mellon university Pittsburgh PA software engineering institute.

National Cyber Security Centre. (2012). *Cyber Security Assessment Netherlands CSAN-2.*

National Cyber Security Centre. (2013). *Policy for arriving at a practice for Responsible Disclosure*

National Cyber Security Centre. (2014). *Cyber Security Assessment Netherlands CSAN-4.* Retrieved from https://english.nctv.nl/Images/cybersecurityassessmentnetherlands2014_tcm92-580598.pdf?cp=92&cs=65035

National Cyber Security Centre. (n.d.). Retrieved from https://www.forensicinstitute.nl/knowledge_lab/partners_in_developing_knowledge/national_cyber_security_centre/

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology.

Pauli, J. J., & Engebretson, P. H. (2008, April). Towards a specification prototype for hierarchy-driven attack patterns. In *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on* (pp. 1168-1169). IEEE.

Schaeffer-Filho, A., & Hutchison, D. (2014). Attack Pattern Recognition Through Correlating Cyber Situational Awareness in Computer Networks. In *Cyberpatterns* (pp. 125-134). Springer International Publishing.

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2013). *Security Patterns: Integrating security and systems engineering*. John Wiley & Sons.

Terms of Reference Working Group 2 – Information sharing (2013). Retrieved from https://resilience.enisa.europa.eu/nis-platform/shared-documents/wg2-documents/wg2-nis-platform-terms-of-reference/at_download/file

The Minister of Security and Justice. (2013). *National Cyber Security Strategy 2.*

Thonnard, O., & Dacier, M. (2008). A framework for attack patterns' discovery in honeynet data. *digital investigation*, *5*, S128-S139.

Tutorial 3: Semantic Modelling. (n.d.). Retrieved from http://www.linkeddatatools.com/semantic-modeling

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The knowledge engineering review*, *11*(02), 93-136.

Uzunov, A. V., & Fernandez, E. B. (2014). An extensible pattern-based library and taxonomy of security threats for distributed systems. *Computer Standards & Interfaces*, *36*(4), 734-747.

Verizon. (2014). 2014 Data Breach Investigation Report. Retrieved from http://www.verizonenterprise.com/DBIR/2014/reports/rp_dbir-2014-executive-summary_en_xg.pdf

Web Application Security Consortium. (2010). *WASC,'Threat Classification,'*. Technical report, Version 2.00.

Zhu, Y. (2011). Attack pattern discovery in forensic investigation of network attacks. *Selected Areas in Communications, IEEE Journal on*, *29*(7), 1349-1357.

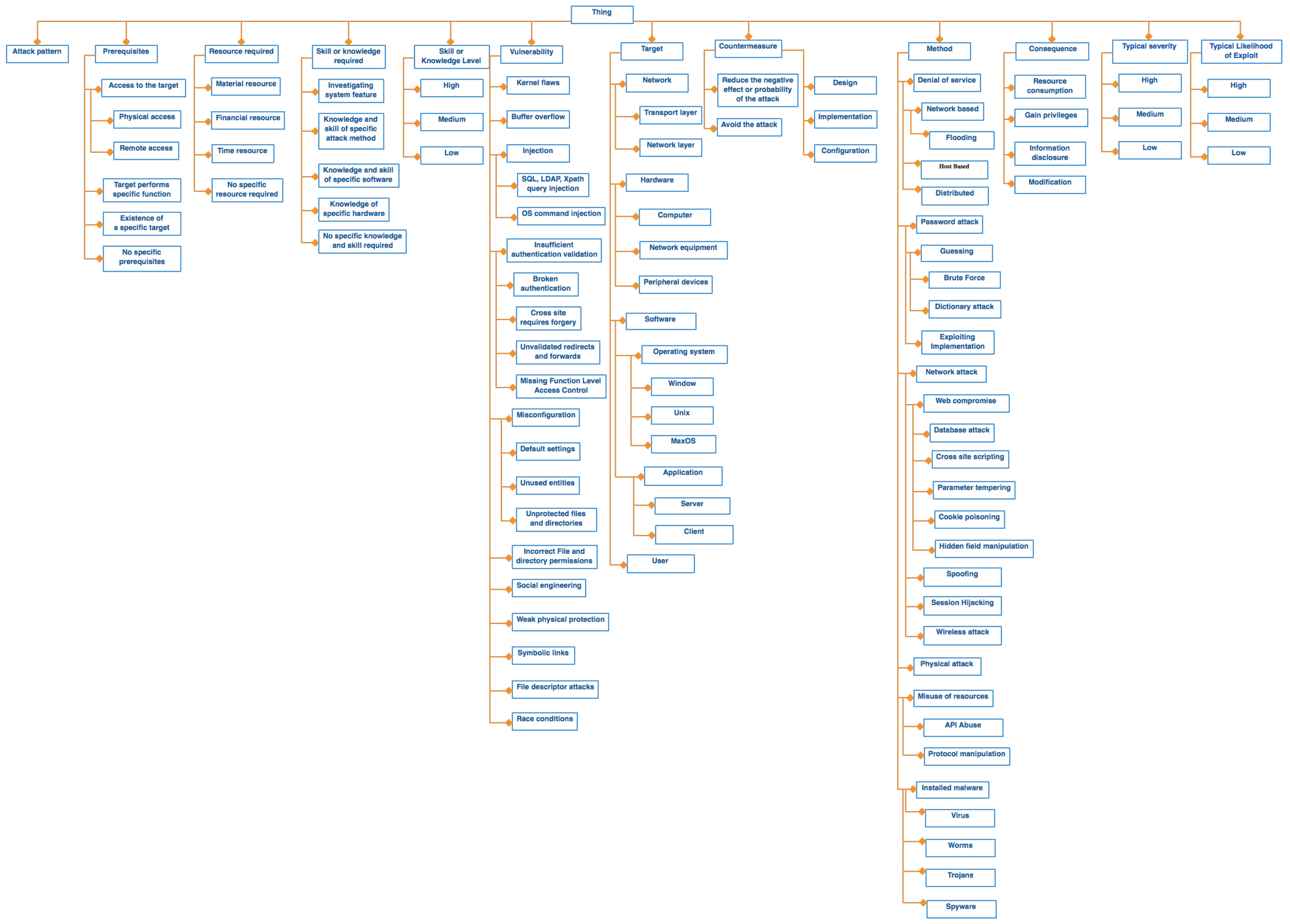
Appendix A Class hierarchy

Table 6 Class hierarchy

Top-level concepts	Middle level concepts					
Prerequisites	Target performs specific function					
	Existence of a specific target					
	Access to the target	Physical access				
		Remote access				
	No specific prerequisites					
	…					
Resource required	Material resource required					
	Financial resource required					
	Human resource required					
	Time resource required					
	No specific resource required					
	…					
Skill or Knowledge required	Skill of investigating system feature					
	Knowledge and skill of specific attack method	Knowledge of SQL				
		Send HTTP requires, run the scan tool				
		Social engineering technique				
		…				
	Knowledge and skill of specific software					
	Knowledge of specific hardware					
	No specific knowledge and skill required					
	…					
Target	Network	Application				
		Presentation layer				
		Session layer				
		Transport layer				
		Network layer				
		…				
	Software	Operating system	Windows		Name	Version
			Unix			
			MaxOS			
			…			
		Application	Server	Database		
				Email		
				Web		
			Client			
	Hardware	Computer				
		Network equipment				

		Peripheral devices	
	User		
Vulnerabilities	Kernel flaws		Unrestricted Consumption
			…
	Buffer overflow		
	Insufficient input validation	Injection	SQL, LDAP, Xpath query injection
			Cross-site Scripting (XSS)
			OS command injection
			…
		…	
	Insufficient authentication validation	Broken authentication	
		Cross site requires forgery	
		Unvalidated redirects and forwards	
		Missing Function Level Access Control	
		…	
	Misconfiguration	Default settings	
		Unused entities	
		Unprotected files and directories	
		…	
	Incorrect File and directory permissions		
	Social engineering		
	Weak physical protection		
	Symbolic links		
	File descriptor attacks		
	Race conditions		
Method	Denial of service	Network based	Flooding
		Host based	
		Distributed	
	Password attack	Guessing	Brute Force
			Dictionary attack
		Exploiting Implementation	
	Network attack	Web compromise	Database attack
			Cross site scripting
			Parameter tempering
			Cookie poisoning
			Hidden field manipulation
		Spoofing	
		Session Hijacking	
		Wireless attack	
	Physical attack		
	Misuse of resources	API Abuse	
		Protocol manipulation	

	Installed malware	Virus
		Worms
		Trojans
		Spyware
Countermeasure	Reduce the negative effect or probability of the attack	
	Avoid the attack	
Countermeasure	Design	
	Implementation	
	Configuration	
Consequence	Resource consumption	
	Gain privileges	
	Information disclosure	
	Modification	
Skill or Knowledge Level	High Skill or Knowledge Level	
	Medium Skill or Knowledge Level	
	Low Skill or Knowledge Level	
Typical Severity	High Typical Severity	
	Medium Typical Severity	
	Low Typical Severity	
Typical Likelihood of Exploit	High Typical Likelihood of Exploit	
	Medium Typical Likelihood of Exploit	
	Low Typical Likelihood of Exploit	


Thing
Attack pattern
Prerequisites
Access to the target
Physical access
Remote access
Target performs specific function
Existence of a specific target
No specific prerequisites
Resource required
Material resource
Financial resource
Time resource
No specific resource required
Skill or knowledge required
Investigating system feature
Knowledge and skill of specific attack method
Knowledge and skill of specific software
Knowledge of specific hardware
No specific knowledge and skill required
Skill or Knowledge Level
High
Medium
Low
Vulnerability
Kernel flaws
Buffer overflow
Injection
SQL, LDAP, Xpath query injection
OS command injection
Insufficient authentication validation
Broken authentication
Cross site requires forgery
Unvalidated redirects and forwards
Missing Function Level Access Control
Misconfiguration
Default settings
Unused entities
Unprotected files and directories
Incorrect File and directory permissions
Social engineering
Weak physical protection
Symbolic links
File descriptor attacks
Race conditions
Target
Network
Transport layer
Network layer
Hardware
Computer
Network equipment
Peripheral devices
Software
Operating system
Window
Unix
MaxOS
Application
Server
Client
User
Countermeasure
Reduce the negative effect or probability of the attack
Avoid the attack
Design
Implementation
Configuration
Method
Denial of service
Network based
Flooding
Host Based
Distributed
Password attack
Guessing
Brute Force
Dictionary attack
Exploiting Implementation
Network attack
Web compromise
Database attack
Cross site scripting
Parameter tempering
Cookie poisoning
Hidden field manipulation
Spoofing
Session Hijacking
Wireless attack
Physical attack
Misuse of resources
API Abuse
Protocol manipulation
Installed malware
Virus
Worms
Trojans
Spyware
Consequence
Resource consumption
Gain privileges
Information disclosure
Modification
Typical severity
High
Medium
Low
Typical Likelihood of Exploit
High
Medium
Low