2007

An Ontological Approach Applied to Information Security and Trust

Artem Vorobiev
Swinburne University of Technology, Melbourne, avorobiev@ict.swin.edu.au

Nargiza Bekmamedova
Swinburne University of Technology, Melbourne, nbekmamedova@ict.swin.edu.au

An Ontological Approach Applied to Information Security and Trust

Artem Vorobiev, Nargiza Bekmamedova
Faculty of Information and Communication Technologies
Swinburne University of Technology
Melbourne, Australia
Email: {avorobiev, nbekmamedova}@ict.swin.edu.au

Abstract

Software applications become highly distributed and complex, involving independent collaborating components working towards achieving system goals. At the same time, security attacks against these applications have also grown being more sophisticated and are quite difficult to detect and withstand, especially distributed attacks. In this paper, we argue that one way to identify and mitigate such attacks is through the trust-based collaboration of application components. However, to achieve collaborative defense in distributed environments, a common vocabulary is needed for the components to collaborate with each other in identifying security incidents. Thus, we employ an ontological approach to define security ontologies as a common vocabulary that is understandable for both humans and software agents. Further, we introduce basic security concepts and trust implications, explain our security ontologies (specified in OWL) that include the security asset-vulnerability ontology (SAVO), the security algorithm-standard ontology (SASO), the security function ontology (SFO), and the security attack and defence ontologies (SAO and SDO respectively). Trust is also examined while its dimensions are employed to create trust-based communications used to distribute security ontologies. We use a case study involving Mitnick attacks to demonstrate our approach.

Keywords

Information security, Security ontology, Security attacks, Security defenses, Trust.

Introduction

Software applications become highly distributed and increasingly complicated, implicating various components that collaborate with each other in order to achieve system objectives. Simultaneously, attackers become smarter in creating new types of attacks, especially distributed attacks, which are quite difficult to identify and mitigate. Thus, we argue that it is possible to detect and resist against such distributed attacks through the collaboration of a system's constituent components. However, to achieve collaborative defense in distributed environments such components should have a common vocabulary to allow them to communicate with each other regarding security attacks and countermeasures in a trusted way. For such purpose, we employ an ontological approach that allows sharing a common understanding of information about information security (security attacks and defences, in particular) among both humans and software agents.

In recent years the development of ontologies, explicit formal specifications of the terms in the domain and relations among them, has been moving from the realm of Artificial-Intelligence laboratories to the desktops of domain experts (Noy & McGuinness 2007). Such practice of applying ontologies has become a very wide and common fact on the World Wide Web. For example, the WWW Consortium (W3C) has developed the Resource Description Framework (RDF) (Brickley & Guha 1999), a language for encoding knowledge on Web pages to make them understandable to electronic agents searching for information. Then, the Defence Advanced Research Projects Agency (DARPA) in conjunction with the W3C have developed DARPA Agent Markup Language (DAML) and its successor Web Ontology Language (OWL) through extending RDF with more expressive constructs aimed to facilitate agent interaction on the Web (Hendler & McGuinness 2000). Thus, many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields. For instance, medicine can explicitly be taken as an example that has produced large standardized structured vocabularies such as SNOMED (Price & Spackman 2000). So, any ontology defines a common vocabulary for domain experts who need to share information in the domain that encompasses machine-interpretable definitions of basic concepts within this domain and relations among them.

In this paper, a number of security ontologies is given with one of the purposes to depict an overall picture of the field of information security that covers basic security concepts such as attacks, assets, functions, vulnerabilities, security algorithms, etc. Besides, these security ontologies might be very helpful for information security researchers and experts. Moreover, we suggest that the proposed security ontologies could be practically justified and commonly utilised as guidelines to better protect an organization's computer environment against various

security incidents. However, once we agree to rely on such guidelines, we should think to what degree trust should be spread out and how to guarantee trust-based collaboration among different parties. Therefore, we employ security ontologies for such purposes and verify how trust might affect them. Consequently, two types of trust are considered in this paper: technology trust and relationship trust. More specifically, trust in the context of information security is evidenced in two domains of the dependencies: (1) trust in the technology that serves as a transmission medium for transferring data flow or interaction among software agents, i.e. technology trust, and (2) trust among partners (agents), i.e. relationship trust (Ratnasingam 2005). Overall, the bottom line of the paper is to enable trust-based collaboration among humans or software agents through exploiting security ontologies as a common vocabulary in order to resist and mitigate security attacks. Finally, we specify our ontologies in OWL and demonstrate them through the use of an example involving the Mitnick attack. In conclusion, we outline directions for future research.

Theoretical Background and Motivation

Related Works

Currently, there are a few approaches that allow publishing semantic data including OWL (OWL 2007), OWL-S (OWL-S 2007), Semantic Web Services Language (SWSL) (SWSL 2007), etc. However, none of them is designed specifically to express security issues. Besides, there are only few works in the area of security ontologies. One of them (Kim, Luo & Kang 2005) introduces the NRL ontology that describes types of security information including security mechanisms, protocols, algorithms, objectives, and credentials. While it is based on the work done by Denker, Nguyen, & Ton (2004), it expresses security related information not only for Web Services but for all types of resources as well. Further, a number of studies have identified a lack of trust as one of the main possible constraints in the area of information security, particularly in terms of data protection, its secure delivery and other issues that focus on three main aspects of trust in data transactions: identity, privacy and security (Guerra & Zizzo 2003). Although, there are plenty of works done in the every single area of information security that somehow explores trust implications. However, none of the existed literature provides any information about how trust might influence on collaboration of distributed parties, i.e. whether security system (that utilise security ontologies) might be enhanced by the adoption of certain strategies designed to enhance trust-based collaboration. To be more consistent, take for example e-business area, the research in this area has been investigating trust implications for a number of years. And as the result, a variety of strategies to enhance trust in e-commerce have been developed, where four of them were best suited to meet demands of the information economy: identity establishment, third-party certification, loss insurance and legal frameworks (Guerra & Zizzo 2003). Finally, our security ontologies are more intuitive to understand and more complete than other security ontologies (Kim, Luo & Kang 2005; Denker, Nguyen & Ton 2004; Undercoffer et al. 2004; Martimiano & Moreira 2006; Seacord & Householder 2005).

Motivation of Research

Software systems become increasingly distributed that involve many independent and collaborating components working towards to achieve certain system objectives. Simultaneously, attackers become smarter in creating more sophisticated security attacks, especially distributed attacks. In order to detect and withstand such attacks, system components should collaborate and communicate with each other by sharing a common vocabulary. In our case, such vocabulary is based on the proposed security ontologies distributed among components in a trusted way. Further, the ontologies define the security concepts and their dependencies comprehensible to both humans and software agents. Additionally, the proposed approach comprises a factor of trust as one of the elements that impact on the collaboration among components within software systems.

Security Ontologies

Introduction

The following definition of the term “ontology” is extracted from Uschold, Moralee & Zorgios (1998): “An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms.” In other words, ontologies allow sharing common understanding of the structure of information among both people and software agents (Noy & McGuinness 2007).

Currently, many research works have applied an ontological approach to different knowledge domains due to their several advantages over other classifications such as taxonomies. The reasons of so much popularity of an ontological approach are provided as follows:

- Ontologies specify many semantic relationships between various entities while a standard taxonomy has only three relationships;
- Ontologies share a common understanding of structured information among different parties such as humans or software agents which in turn, can be reasoned and analysed automatically;
- Ontologies are reusable and able to evolve over time;
- Ontologies are shared among different parties to solve interoperability problems.

Our main ontology is called Security Asset-Vulnerability Ontology (SAVO) that illustrates how attacks against peers (hosts) may affect their assets protected by defensive mechanisms; how vulnerabilities are exploited by threat agents in order to perform attacks; and how assets are evaluated by using the quantitative and qualitative analysis. SAVO is the main security ontology that binds other security concepts, mechanisms and ontologies including the security attack ontology (SAO), the security defence ontology (SDO), the security algorithm-standard ontology (SASO), and the security function ontology (SFO). Besides, these security ontologies depict the situation with various information security-related issues and assist software developers to create the more efficient protection against attacks and system failures in terms of money and time. Further, SAVO can be utilized as a common vocabulary by various defensive components (e.g. intrusion detection components) in order to interact with each other, share a common understanding of information about security attacks, and ensure better protection. Also, SAO correlates closely with SDO, which is mainly used to specify certain defensive mechanisms, resist against security attacks, and reveal dependencies among security algorithms and standards expressed in SASO and SFO.

The Security Asset-Vulnerability Ontology (SAVO)

In this section, we briefly introduce several security terms used in our ontologies and explain their relations, as illustrated in Figure 1. However, we would like to note that due to the limit of space, descriptions of some of security concepts can be found in the work written by Stewart, Tittel & Chapple (2005).

SAVO is considered to be the main among all security ontologies that incorporates other our security ontologies including SASO, SAFO, SAO, and SDO. Similarly, SAVO (as other our security ontologies) is designed to support several features as the NLR ontology (Kim, Luo & Kang 2005) such as (1) the ability to specify security information for different types of resources and environments; (2) the reusability and extensibility; and (3) mapping between high-level and low-level security requirements and capabilities. Thus, SAVO is a high level security ontology that depicts information security in a simplified manner especially for non-security professionals, and its design is based upon our expertise and knowledge of information security. SAVO relates to some research works done in information security area (Stewart, Tittel & Chapple 2005; Kim, Luo & Kang 2005; Denker, Nguyen & Ton 2004; Undercoffer et al. 2004; Martimiano & Moreira 2006; Seacord & Householder 2005).

First, we introduce several security terms crucial for better understanding of the given information. The full description of security terms can be found in Stewart, Tittel & Chapple (2005). We start with the term ‘threat’ (the class *‘Threat’* in Figure 1) that refers to any occurrence that may cause any unwanted outcome for a company. A threat agent (the class *‘ThreatAgent’*) is an agent that can use a threat in order to exploit vulnerability, while vulnerability (the class *‘Vulnerability’*) in turn is the absence or the weakness of defence (e.g. error, flaw, etc). Risk (the class *‘Risk’*) is the possibility that a threat agent will exploit vulnerability to damage an asset whereas exposure (the class *‘Exposure’*) reveals the possibility that vulnerability will be exploited by a threat agent. Besides, vulnerability may also result in exposure, every instance of which equals risk that can be mitigated by using a safeguard (the class *‘Defence’*). On the other hand, a safeguard is utilised to resist attacks (the class *‘Attack’*) and applies security techniques and mechanisms (the classes *‘SecurityFunction’* and *‘SecurityAlgorithmStandard’*). Now, we can proceed further with the terms such as ‘asset’ that refers to anything within any environment that should be protected, i.e. data (the class *‘Data’*), software (the class *‘Software’*), accounts (the class *‘Account’*), resources (the class *‘Resource’*), etc. Then, the class *‘Asset’* also can be subdivided into subclasses including but not limited to *‘ClientData’* and *‘SystemData’* (both refer to client and system data); *‘Component’* and *‘Service’* (refer to software implementation) while such subclasses as *‘CPU’*, *‘Memory’* and *‘Storage’* encompass available resources; *‘ClientAccount’* and *‘SystemAccount’* are subclasses of the class *‘Account’*. Further, any asset may contain vulnerability which is explored with other vulnerabilities. The class *‘Vulnerability’* has three properties, vulnerability name (the class *‘VulnName’*), an attribute (the class *‘VulnAttribute’*) and associated values (the class *‘VulnValue’*). Then, a patch (the class *‘Patch’*) removes vulnerability and is developed by a supplier (the class *‘Supplier’*). Whereas a threat agent may use a threat to perform an attack (the class *‘Attack’*) and endanger the asset, it also exploits vulnerability for attacking a peer (the class *‘Peer’*) that hosts the assets (in our case, a peer is a victim of an attack).

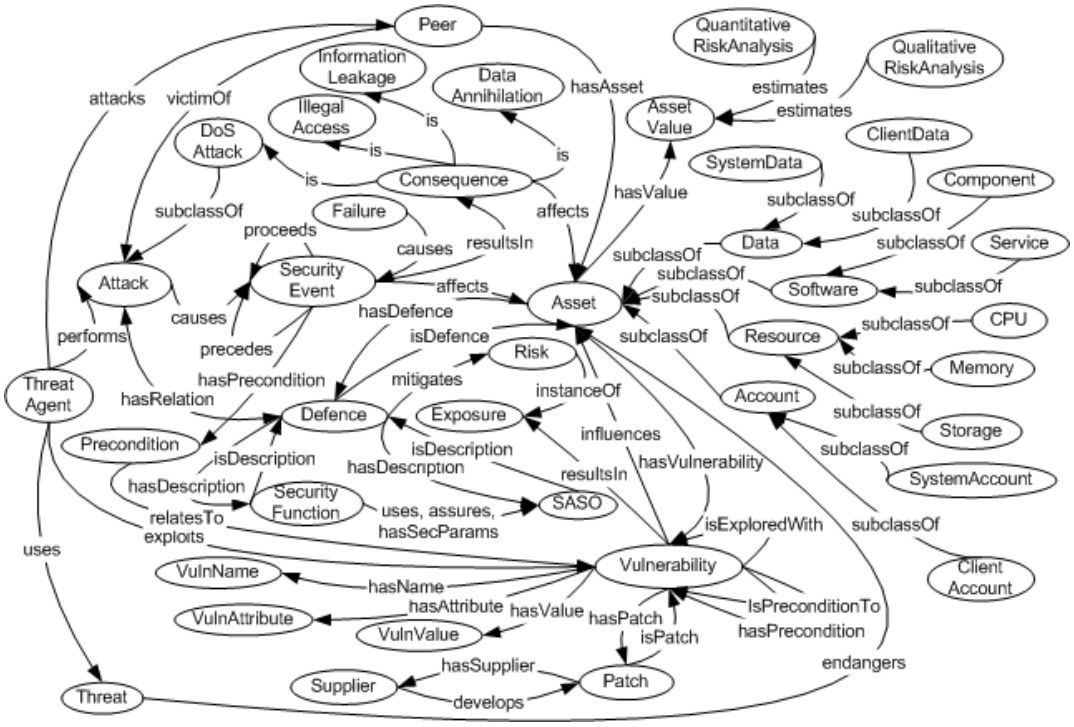


Figure 1: The security asset-vulnerability ontology

Moreover, any occurred attack leads to security event (the class *'SecurityEvent'*) which can precede another one and which, in turn, has a precondition (the class *'Precondition'*) that relates to vulnerability. Such security event is likely to be caused by a system failure (the class *'Failure'*) and it also may affect an asset and result in a consequence (the class *'Consequence'*) which in turn, affects an asset either. And this resulted consequence has a number of subclasses such as destruction of data (the class *'DataAnnihilation'*), an information leakage (the class *'InformationLeakage'*), an illegal access (the class *'IllegalAccess'*) or a DoS attack (the class *'DoSAttack'*).

In addition, any asset has a value (the class *'AssetValue'*) that might be estimated through exploiting both the quantitative risk analysis (the class *'QuantitativeRiskAnalysis'*) and the qualitative risk analysis (the class *'QualitativeRiskAnalysis'*). The quantitative risk analysis allows to estimate a concrete probability percentage. There are few formulas associated with the quantitative risk analysis:

- The exposure factor (**EF**) shows the percentage of loss that a company experiences if a certain asset is violated by a realized risk. This exposure factor is expressed in percentage (%);
- The single loss expectancy (**SLE**) is the cost of loss calculated using the formula. SLE is expressed in a money value ($) and is calculated using the following formula:

  SLE [$] =AV * EF; where **AV** – asset value;
- The annualised rate of occurrence (**ARO**) is the expected frequency of certain risk or threat occurring within one year. ARO is expressed utilizing the following formula:

  ARO= Q_o/year; where Q_o – quantity of occurrence;
- The annualised loss expectancy (**ALE**) is the possible cost per year of all instances of a certain realised threat against a certain asset. ALE is counted using the following formula:

  ALE=SLE*ARO=AV*EF*ARO

The pure quantitative analysis is not possible because some aspects of information security cannot be quantified in money figures. While the qualitative risk analysis usually goes hand in hand with the appliance of such factors as experience, judgment, intuition, and so on.

To conclude, SAVO is easily modifiable and extendable via adding supplementary subclasses.

The Security Algorithm-Standard Ontology (SASO)

This section of the paper aims to introduce the Security Algorithm-Standard Ontology (SASO) which encompasses security algorithms, standards, concepts, credentials, objectives, assurance levels, etc. Moreover, these security terms are classes of ontology and currently utilised as "building blocks" for other ontologies including the Security Function Ontology (SFO).

SASO consists of several classes including *'SecurityAlgorithm'*, *'SecurityConcept'*, *'SecurityAssurance'*, *'SecurityCredential'*, and *'SecurityObjective'*. The class *'SecurityAlgorithm'* is utilised to define various security algorithms. The structure of this class is illustrated in Figure 2, where the main class called *'SecurityAlgorithm'* has three major subclasses including *'SAlgEncryption'*, *'SAlgSignature'*, and *'SAlgKeyExchange'*. The class *'SAlgEncryption'* defines classes that deliver encryption/decryption capabilities and consists of two subclasses (the classes *'SAlgSymmetric'* and *'SAlgAsymmetric'*) that represent symmetric and asymmetric cryptographic algorithms. Then, the class *'SAlgKeyExchange'* specifies cryptographic key exchange algorithms while the class *'SAlgSignature'* includes classes responsible for digital signatures and cryptographic hash functions.

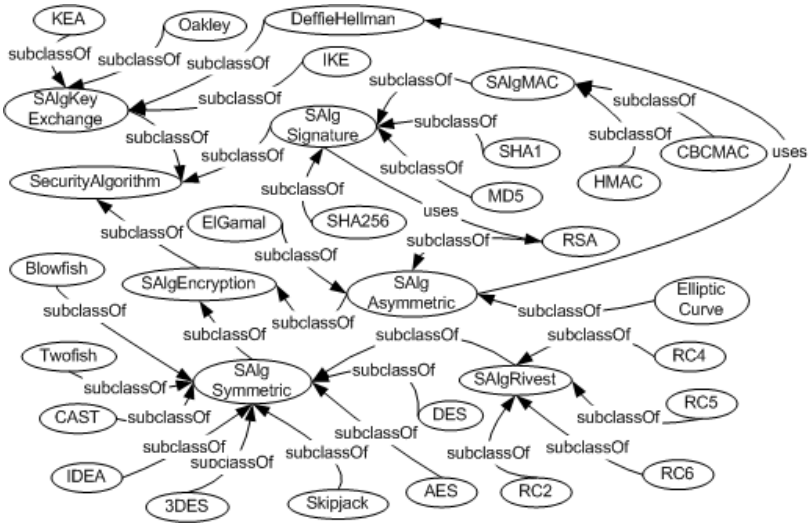


Figure 2: The class SecurityAlgorithm

The class *'SecurityConcept'* consists of three subclasses (not illustrated here due to limited space) that involve the subclasses named as *'SConMechanism'*, *'SConProtocol'* and *'SConSecurityPolicy'* consequently. We inherit the similar structure as in the class *'SecurityConcept'* of the main security ontology extracted from Kim, Luo & Kang (2005). Moreover, there is a difference between security protocols and security mechanisms, where the former (the class *'SConProtocol'*) defines how to fulfil certain tasks using certain steps while the latter (the class *'SConMechanism'*) is an implementation of security protocols. The class *'SConMechanism'* mainly defines various types of firewalls, proxies and virtual machines (VMs) while the class *'SConProtocol'* has many subclasses that specify different security protocols being used in authentication (the class *'SConAuthentication'*), encryption (the class *'SConEncryption'*), key management (the class *'SConKeyMngmt'*), digital signatures (the class *'SConSignature'*), secure e-mail systems (the class *'SConEmail'*), and secure communications (the class *'SConSecureCommunication'*). The class *'SConSecurityPolicy'* describes rules and policies used in access control models. The class *'SecurityAssurance'* (not illustrated here) specifies assurance methods for security algorithms, protocols, and mechanisms. Currently, it has only one subclass called *'CommonCriteria'* (CC) (CC IS 2007), which is an international standard for a computer security evaluation and which is used to evaluate security measures applied to software applications. Given the extensibility of our ontology, new classes such as *'TCSEC'* (Trusted Computer System Evaluation Criteria) can be easily added. The class *'SecurityCredential'* is commonly used for identification and authentication purposes. Finally, the class *'SecurityObjective'* (not illustrated here also due to limited space) is used to define security objectives including confidentiality, authorisation, integrity, availability, non-repudiation, authenticity, auditability, identification, accountability, and survivability. Some of them are related to trust issues and considered to be dimensions of the technology trust (see our example).

The Security Function Ontology (SFO)

The main objective of this section is to introduce the Security Function Ontology (SFO) that is widely applied in conjunction with SASO to provide a vocabulary for our ontologies and for SCL (Security Characterisation Language) (Khan & Han 2003; Khan 2005) and Extended SCL (ESCL) (Vorobiev & Han 2006b) in particular. The classes *'SecurityAlgorithm'* and *'SecurityConcept'* are used as SCL/ESCL function parameters and utilised for further analysis and reasoning of those security functions applied to specify the security interface of components and services. The main class of SFO is called '*SecurityFunction*' which contains six classes of security functions derived from merging eleven Common Criteria classes: *'CryptoSupportFunc'*, *'IdentificationAuthorisationFunc'*, *'PrivacyFunc'*, *'UserDataProtectionFunc'*, *'TrustedChannelFunc'*, and '*SecurityAuditResourceUtilisationFunc'*. There are also additional classes including but not limited to *'Time'*,

'Probability' and *'SecProp'*, where the latter refers to the class of security properties. The class of security functions for cryptographic support (the class *'CryptoSupportFunc'*) is designed to support encryption/decryption algorithms. Security functions or their sets assure certain security goals such as confidentiality, integrity, availability, etc. The class *'SecurityEntity'* just binds various security entities used in SFO such as cryptographic keys or certificates. To be more precise, we provide a brief example illustrated in Figure 3:

Figure 3: An example with the function Signed (X, K^{-1})

Initially, there are about forty four security functions that are not presented here due to limited space.

Relationships among Security Classes

In this section, we describe relationships among different security classes of SASO and SFO, as illustrated in Figure 4.

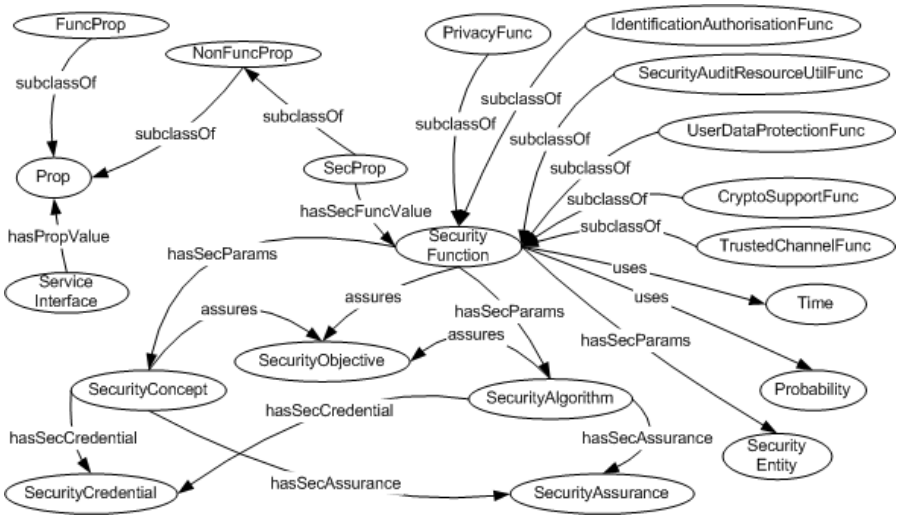


Figure 4: Relations between various security concepts

As shown above, the main class of SFO is the class 'SecurityFunction' that consists of six subclasses introduced in the previous sections. All these subclasses encompass security functions with the property named as *'hasSecParams*' that in turn, contains the classes 'SecurityConcept' (SASO), *'SecurityAlgorithm'* (SASO), and *'SecurityEntity'* as values. These three classes are used as values since they define security algorithms and standards as well as security entities such as cryptographic keys or operations. Moreover, such classes as *'Time'* and *'Probability'* may be utilised via the class *'SecurityFunction'* and its subclasses in order to specify security of software systems. Since, many security algorithms and concepts have various assurance levels and use security credentials for access control, some of the SASO classes (the classes *'SecurityConcept'* and *'SecurityAlgorithm'*) have two properties, among those are *'hasSecCredential'* and *'hasSecAssurance'* which have classes *'SecurityCredential'* and *'SecurityAssurance'* as values accordingly. The classes *'SecurityFunction'*, *'SecurityConcept'*, *'SecurityAlgorithm'* and combinations of their subclasses guarantee security objectives predefined by the class *'SecurityObjective'*.

The Security Attack and Defence Ontologies (SAO and SDO)

In this section, we define the Security Attack Ontology (SAO) and the Security Defence Ontology (SDO) used as a common vocabulary by software agents (e.g. a coalition of defensive components) or humans. Defensive components (DCs) (e.g. intrusion detection components or honey-pots) and humans use SAO and SDO to interact with each other and share a common understanding of information about security attacks and defenses. Both SAO and SDO have the ability to evolve over time.

As illustrated in Figure 5, once a DC from a coalition detects a new attack it adds the attack as a new class to SAO and shares the ontology with other members of the coalition. When any coalition member develops a

countermeasure against this attack a countermeasure is added to SDO as a new class which is then distributed among other coalition members. Additionally, many attacks especially distributed multi-phased attacks such as the Mitnick attack (Undercoffer et al. 2004) can be detected by such coalitions.

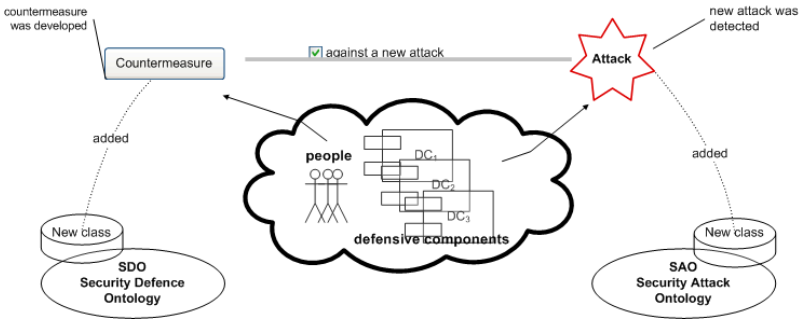


Figure 5: An illustrated example

The main SAO class called *'Attack'*, has five subclasses (not illustrated here) *'WSAttack'*, *'P2PAttack'*, *'DoSAttack'*, *'SniffingAttack'*, and the class called *'MultiPhasedDistributedAttack'*. The class *'WSAttack'* defines attacks on Web services (WS) (Vorobiev & Han 2006a; Stamos & Stender 2005) while the class *'P2Pattack'* specifies attacks on the peer-to-peer (P2P) systems. The class *'DoSAttack'* describes various denial of service attacks. Attacks on communication channels are characterized by the class *'SniffingAttack'*. So that, Mitnick attack is an explicit example of multi-phased distributed attacks (Undercoffer et al. 2004), which in turn, is the subclass of the class *'MultiPhasedDistributedAttack'* performed through utilising attacks from the classes *'DoSAttack'*, *'WSAttack'*, and *'SniffingAttack'*. The class *'Attack'* has a property called *'hasRelation'* which binds the class *'Defence'*. The class hierarchy of SDO is quite similar to SAO. The main SDO class is called *'Defence'* and it has five subclasses correlated with five SAO classes including defence against attacks on Web services (the class *'WSDefence'*) and P2P systems (the class *'P2PDefence'*), safeguards against DoS attacks (the class *'DoSDefence'*), countermeasures against sniffing attacks (the class *'SniffingDefence'*) and multi-phased distributed attacks (the class *'MultiPhasedDistributedDefence'*). The class *'Attack'* has the following two properties named as *'hasRelation'* (tied to the class *'Attack'*) and *'hasDescription'* that utilises SASO and SFO described in the previous sections. Alas, the detailed description of SAO and SDO ontologies is out of the scope of this paper.

Trust

In this section, we give an introductory word about trust, without which security ontologies cannot be distributed among various parties in a secured and trusted way. Trust has been studied at length in many disciplines. Pettit (1995) makes a distinction between the attribute of trust and the behaviour of trust. Misztal (1996) identifies different types of trust according their role: commercial, problem solving, informational, knowledge and identity; while Kini & Choobineh (1998) make a distinction from three different perspectives on trust: individual, societal and relationship. And as no consensus has emerged on what trust means, this paper begins with narrowing concepts of trust within information security. Once security ontologies have been specified, trust is expected to include high-level of assurance, reliability and confidentiality. Trust has the ability to evolve and thus, it has different levels: technical level that mainly is supported by security algorithms and cryptography; and relationship level, i.e. non-technical that includes humans to interact and communicate, however the latter is out of the scope of this paper.

Technology trust is supported by security technologies embedded in standardized interaction processes. The foundations of technology trust are technical safeguards, protective measures, and access control mechanisms that aim to provide reliable transactions, data flows, and detective mechanisms. Trust is formed among different parties through the use of various policies (e.g. security policies). Dimensions of technology trust may include security services which examine confidentiality, integrity, authentication, non-repudiation, and access controls (Ratnasingam 2005). In other words, in the frame of our security ontology there is a coalition of members that includes both humans and software agents. To securely distribute security ontologies among coalition members these ontologies should be encrypted. For example, AES (Advanced Encryption Standard) (Stewart, Tittel & Chapple 2005) can be applied here. Similar to most symmetric cryptography algorithms, AES delivers confidentiality. Then, an encrypted ontology should be digitally signed and hence, these digital signatures support integrity, non-repudiation, and authentication. Thus, such characteristics of security services include technical solutions that serve to support availability, confidentiality, integrity, authenticity, and accountability of any transactions.

Relationship trust enforces governance mechanisms to apply in the form of contracts, regular audit policies, top management commitment, quality standards, awareness training, and risks management procedures that impact and control data flows. Initially, relationship trust came from social and organizational studies and thus, it encompasses interpersonal or social aspects of relationships that involve humans. In our case, such kind of trust exists among different parties that collectively assess transactions occurred according to their expectations. Dimensions of relationship trust may include competence trust, predictability trust, and goodwill trust that primarily focuses on humans' behaviour. But due to limited space, we are unable to give a comprehensive report on every type of trust, however, the detailed picture can be found in a number of works (Ratnasingam 2005; Guerra & Zizzo 2003).

Example

We choose an example that illustrates how our security ontologies might be utilised in distributed environment and how trust might be employed. Initially, we take two hosts, Peer 1 (**P1**) and Peer 2 (**P2**), that are members of a coalition and which help each other to detect and mitigate various types of security attacks performed by an Attacker (**A**). Some of these attacks are possible to detect only by several coalition members that are distributed over the network and cooperate with each other. One of such attacks is the Mitnick attack modified via the use of Web services (WSs).

First, **A** performs the WS Mitnick attack that exploits weakness of the TCP protocol design in making a TCP connection called the three-way handshake, as illustrated in Appendix. Then, **P1** sends a SYN packet to **P2** to initiate a TCP connection. In turn, **P2** sends a SYN/ACK packet to establish a TCP connection with **P1**. Finally, to establish a TCP connection and complete the handshake **P1** receives a SYN/ACK packet and sends an acknowledgment message (a SYN/ACK packet) to **P2**.

Second, in order to detect the WS Mitnick attack, **P1** and **P2** should operate as the coalition. If **P1** detects the SYN/Flood attack, it will send a description of the attack to **P2**'s using our security ontology. To ensure that new security ontology can be trusted, **P1** encrypts it using the AES algorithm that supports confidentiality. Further, **P1** signs the ontology and puts a timestamp to provide integrity, authentication, and non-repudiation using RSA and SHA-256 cryptographic algorithms.

Third, **A** tries to intercept this ontology and modify it. If **A** is successful, then **A** will try to crack the protected ontology which is difficult since it was encrypted, signed and timestamped.

Fourth, **A** can try other ways to create a malicious ontology. **A** can apply social engineering techniques and enforce **P2** to install a Trojan horse which helps to steal a private key.

Fifth, **P2** does not know that it has been cracked; however, **P1** suggests that something wrong is occurring.

Finally, **P1** and **P2** cannot deliver trust by technical mechanisms and therefore, they have to apply non-technical means such as interpersonal communication channels (contracts, policies, conferences, meetings, awareness trainings, etc).

Conclusion and Further Research

Currently, users of software applications require more features, flexibility and better protection. Hence, such applications become highly complicated and distributed containing many various independent components. At the same time, new attacks, especially distributed multi-phased attacks, which are quite difficult to identify and mitigate, appear.

In this paper, we have demonstrated that only through the collaboration of a system's constituent components is possible to detect and withstand such attacks. However, components should have a common basis (vocabulary) to allow them to exchange information with each other about security threats in a trusted way. Therefore, we have utilised an ontological approach to specify information security issues in a way understandable to both humans and software agents. Besides, we have analysed reasons why ontologies were selected. Moreover, we have introduced various security terms, concepts and mechanisms through the specified security ontologies including SASO, SFO, SAO, SDO, and SAVO, where the latter is a high-level ontology that binds other security ontologies that all are employed as the common vocabulary. Additionally, technology trust and relationship trust have been also introduced in the paper. However, this paper focuses mainly on technology trust. Further, the proposed security ontologies have been specified in OWL and the example with the Mitnick attack has been given to illustrate how security mechanisms allow trust-based exchange of security ontologies among different parties.

In future work, we will develop more comprehensive security ontologies, specify trust ontologies and investigate how trust might impact on information security.

References

Brickley, D & Guha 1999 'Resource Description Framework (RDF) Schema Specification. Proposed Recommendation, World Wide Web Consortium, <http://www.w3.org/TR/PR-rdf-schema>.

Common Criteria International Standard (*CC IS*), viewed 6 May 2006 <http://csrc.nist.gov/cc/CC-v2.1.html>.

Denker, G, Nguyen, S & Ton, A 2004 'OWL-S Semantics of Security Web Services: a Case Study', paper presented to SRI International, Menlo Park, California, USA.

Guerra, GA & Zizzo, DJ 2003 'Economics of Trust in the Information Economy: Issues of Identity, Privacy and Security', *Oxford Internet Institute*, *Research report*, No. 1, Apr.

Hendler, J & McGuinness, D 2000 'The DARPA Agent Markup Language', *IEEE Intelligent Systems*, vol. 16, no. 6, pp. 67-73.

Khan, K & Han, J 2003 'A Security Characterisation Framework for Trustworthy Component Based Software Systems', *COMPSAC2003*, pp. 164-169.

Khan, K 2005 'Security Characterisation and Compositional Analysis for Component-based Software Systems', PhD thesis, Monash University, Apr.

Kim, A, Luo, J & Kang, M 2005 'Security Ontology for Annotating Resources', paper presented to the 4th International Conference on Ontologies, Databases, and Applications of Semantics, *ODBASE 2005*.

Kini, A & Choobineh, J 1998 'Trust in Electronic Commerce: Definition and Theoretical Considerations', paper presented to the Proceedings of the 31st Annual Hawaii Conference on System Sciences.

Martimiano, LA & Moreira, E 2006 'The Evaluation Process of a Computer Security Incident Ontology', paper presented to the 2nd Workshop on Ontologies and their Applications, Ribeirão Preto.

Misztal, B 1995 'Trust in Modern Societies', Polity Press, Cambridge MA.

Noy, NF & McGuinness, DL 2007 'Ontology Development 101: A Guide to Creating Your First Ontology', viewed 12 June 2007 <http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html>.

Pettit, P 1995 'The Cunning of Trust', *Philosophy and Public Affairs*, vol. 24, pp. 202-225.

Price, C & Spackman, K 2000 'SNOMED clinical terms', *BJHC & IM – British Journal of Healthcare Computing & Information Management*, vol. 17, no. 3, pp. 27-31.

Ratnasingam, R 2005 'E-commerce Relationships: The Impact of Trust on Relationship Continuity', *International Journal of Commerce & Management*, vol. 15, no. 1, pp. 1-16

Seacord, RC & Householder, AD 2005 'A Structured Approach to Classifying Security Vulnerabilities', *CMU/SEI-2005- TN-003*, Jan.

Semantic Markup for Web Services (OWL-S), viewed 17 May 2007 <http://www.daml.org/services/>.

Semantic Web Services Language (SWSL), viewed 17 May 2007 <http://www.daml.org/services/swsl/>.

Stamos, A & Stender, S 2005 'Attacking Web Services: The Next Generation of Vulnerable Enterprise Apps', *BlackHat2005*, USA.

Stewart, JM, Tittel, E & Chapple, M 2005 *CISSP: Certified Information Systems Security Professional. Study Guide*, 3rd edn, Sybex.

Undercoffer, J, Joshi, A, Finin, T & Pinkston, J 2004 'A target-centric Ontology for Intrusion Detection," paper presented to International Joint Conference on Artificial Intelligence, Mexico.

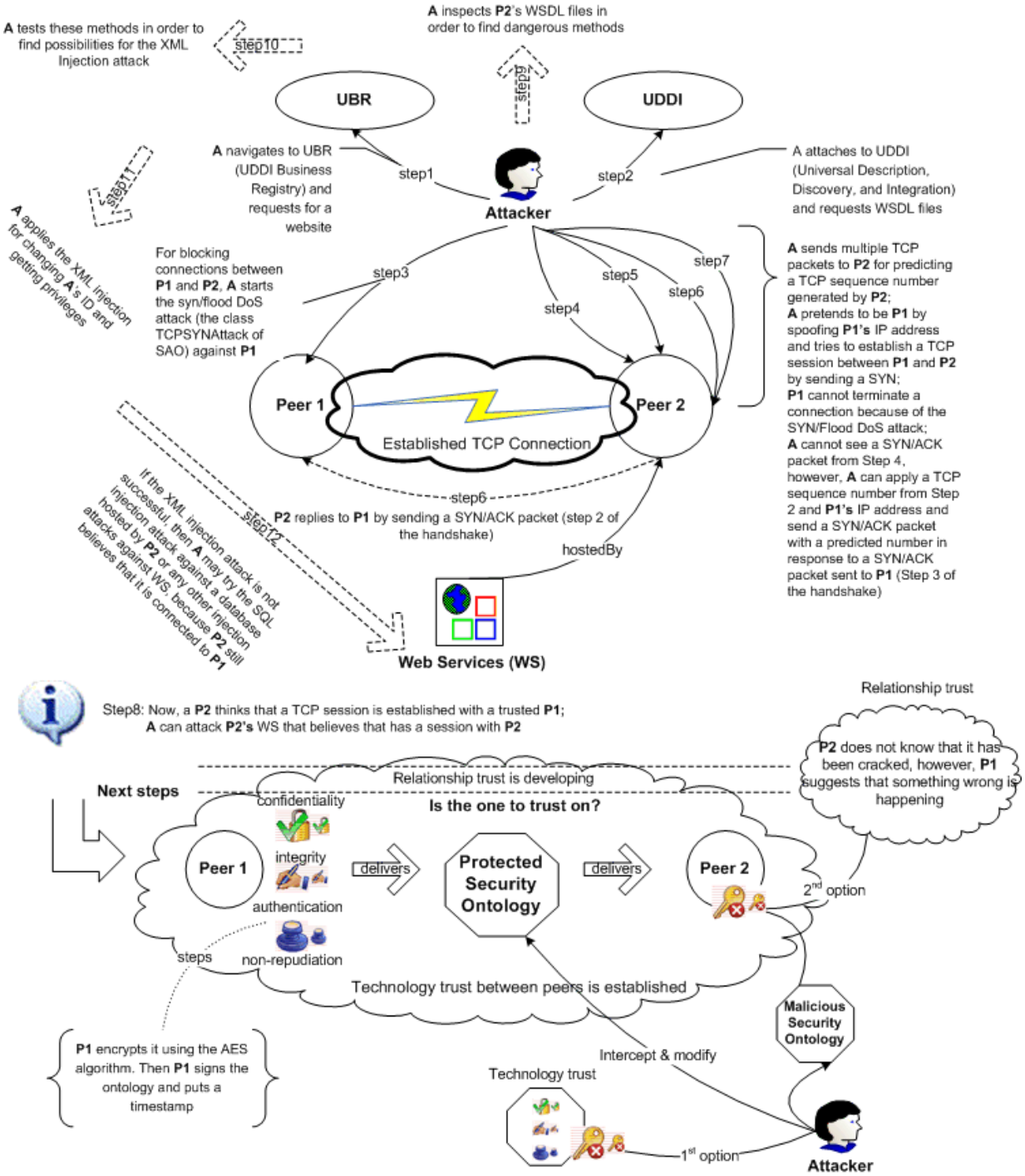
Uschold, MK, Moralee, S, & Zorgios, Y 1998 'The Enterprise Ontology', *The Knowledge Engineering Review*, vol. 13, no. 1, pp.31-89.

Vorobiev, A and Han, J 2006a 'Security Attack Ontology for Web Services", paper presented to Proceedings of the 2nd International Conference on Semantics, Knowledge and Grid (*SKG'06*), Guilin, China, Nov.

Vorobiev, A & Han, J 2006b 'Specifying Dynamic Security Properties of Web Service Based Systems', paper presented to Proceedings of the 2nd International Conference on Semantics, Knowledge and Grid.

Web Ontology Language (OWL), viewed 17 May 2007 <http:w3.org/TR/owl-features/>.

Appendix: The WS Mitnick attack



Copyright