Computers and Electrical Engineering 64 (2017) 96–109

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Network security assessment using a semantic reasoning and graph based approach☆

Songyang Wu, Yong Zhang*, Wei Cao

The Third Research Institute of Ministry of Public Security, Shanghai 201204, China

ARTICLE INFO

Article history:
Received 11 July 2016
Revised 1 February 2017
Accepted 1 February 2017
Available online 21 February 2017

Keywords:
Network security
Security ontology
Attack graph
Semantic reasoning

ABSTRACT

Owing to the high value of business data, sophisticated cyber-attacks targeting enterprise networks have become more prominent, with attackers trying to penetrate deeper into and reach wider from the compromised machines. An important security requirement is that domain experts and network administrators have a common vocabulary to share security knowledge and quickly help each other respond to new threats. We propose an innovative ontology and graph-based approach for security assessment. An ontology is designed to represent security knowledge such as that of assets, vulnerabilities, and attacks in a common form. Using the inference abilities of the ontological model, an efficient system framework is proposed to generate attack graphs and assess network security. The performance of the proposed system is evaluated on test networks of differing sizes and topologies.

1. Introduction

Enterprise networks that employ various IT (Information Technology) services are becoming more and more important for achieving business objectives. Owing to the high value of business related data, sophisticated attacks targeting enterprise networks have become more prominent in recent years. The attackers will no longer stop after a single successful attack, but try to penetrate deeper into and reach wider from the compromised machines. A security administrator for an enterprise network has to combat these multistage and multi-host attack scenarios [1,2]. As the combination of machines and services deployed on enterprise networks become increasingly complex, assessing the overall security of an enterprise network becomes a daunting task for human administrators. To maintain the security and availability of a productive IT infrastructure, a common vocabulary and automated solutions are important for exchanging security knowledge and analysing potential attacks on an enterprise network.

Attack graphs [3] are useful tools that help facilitate scalable security analysis of enterprise networks by illustrating complicated-potentially multistage-attack paths to network administrators. Many security management schemes based on attack graphs have been proposed [2,4–7], these enable analysts to understand the cause and consequences of security risks, and help to determine appropriate countermeasures. In general, security assessment of a network through attack graphs requires the several steps. Firstly, collecting the attributes of the network including topology, services, vulnerabilities, and configurations. Secondly, mapping each attack action to different vulnerabilities using certain formal language (such as Datalog used in MulVAL [4]). Thirdly, generating attack graphs to illustrate the threat of various attack scenarios to critical

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Area Editor Dr. G. Martinez.

* Corresponding author.
E-mail address: zhangyonglab@yeah.net (Y. Zhang).

http://dx.doi.org/10.1016/j.compeleceng.2017.02.001
0045-7906/

assets. Finally, to recommend damage mitigation measures. However, one major drawback of attack graphs is the lack of a common vocabulary to share security knowledge between security domain experts and network administrators. Because of the large volume of information collected from various devices on enterprise networks, administrators face obstacles in effectively modelling security problems, and making correct decisions based on limited security assessment experience.

In this paper, we propose an innovative security assessment approach that uses ontology [8] to share attack knowledge between multiple parties and infer logic-based attack graphs. The ontology serves as a common vocabulary, which is designed to represent security knowledge such as that of vulnerabilities, attacks, and the relationships between them. This ontological model also adopts SWRL (Semantic Web Rule Language) rules to help express the cause-consequence relationships of all known attack scenarios. Using a reasoning engine such as JESS (Java Expert System Shell) [9], potential threats to critical IT assets can be inferred using the predefined SWRL rules. A full attack graph may be constructed depending on the inference abilities of the ontological model. The ontology and SWRL use the Open World Assumption [10], whereby the semantic ontology is extended to incorporate knowledge of new vulnerabilities and threats. Such an approach facilitates information exchange and enables administrators to accomplish security assessment tasks with the help of domain experts. Our major contributions include:

- An enhanced security ontology based upon previous works [11,12] is designed for modelling central concepts and relationships, as well as inferring rules about vulnerabilities, threats, and attacks. This serves as the basis for sharing security knowledge among various agents.
- Based on the inference ability of the security ontology, we propose an efficient and scalable system framework and algorithm to compute a full attack graph.
- Our attack graph model can be easily transformed into a MulVAL attack graph, which supports application of various sophisticated approaches for quantifying attacks risk and developing optimal mitigation plans.
- To clarify discussion, we illustrate the generation of an attack graph with a case study, and the performance of our system is evaluated on test networks of varying sizes and topologies.

The remainder of the paper proceeds in the following manner. Section 2 is a discussion of existing related works. Section 3 presents the design of the architecture. In Section 4, we present the design of an ontological security model. Section 5 describes a scheme for security assessment using semantic reasoning and attack graphs. Section 6 describes performance analysis for attack graph generation on varying networks sizes and topologies. Finally, we outline our conclusions in Section 7.

2. Related works

The issue of information security assessment for enterprise networks has long attracted attention in literature. In the last decade, attack graphs have emerged as a mainstream technique for security analysis on enterprise networks [6].

An attack graph models the interdependencies between network topology, vulnerabilities, and attacks. Various risk metrics can be extracted from attack graphs. Literatures [2,4] constructed the reasoning engine MulVAL, and built an attack graph generation tool based on it. The key idea of MulVAL is the usage of Datalog rules to specify attack techniques and system security semantics. The attack simulation traces computed by the MulVAL reasoning engine are utilized to construct a logical attack graph. Homer et al. [6] presented a model that can be used to produce quantitative security metrics that measure the likelihood of breaches occurring within a given enterprise network. They utilize existing tools [2] for generating attack graphs, and then apply probabilistic reasoning to produce a vulnerability metrics aggregation that is computationally sound and has clear semantics. Poolsappasit et al. [5] proposed a Bayesian network-based risk management framework to quantify the likelihood of attacks on a network using metrics defined in the Common Vulnerability Scoring System (CVSS) [13]. Furthermore, they utilize a genetic algorithm to recommend optimal mitigation plans according to a cost model specified by the user. The structure of Poolsappasit et al.'s Bayesian attack graph is similar to the MulVAL attack graph. NetSecuritas [7] is a security management tool that integrates an open-source scanner, as well as exploit databases and methodologies available in the public domain to facilitate security assessment. It aggregates network and vulnerability information to generate attack graphs for security assessment and mitigation actions.

Although existing graph-based schemes have made significant contributions to enterprise network security analysis , we argue that one major drawback is the lack of a common vocabulary to precisely define and share security knowledge between domain experts and network administrators. The aforementioned proposals generally store attack knowledge and network configurations in Datalog tuples or databases. The relationships of network attack knowledge are concealed within complex data structure definitions. Because of the large volume of information collected from enterprise networks, administrators may find it difficult to effectively model security problems considering their often limited experience with security risk assessment.

Development of a security ontology is one possible solution for this problem, because it allows for a clear and precise definition of all entities and their relationships to each other. The security ontology creates a common, unambiguous semantic language for representing knowledge in the security domain, and serves as a basis for information sharing between people or various other agents [14].

Ontology has many applications in the security domain. Vorobiev et al. [11] argued that collaboration between distributive components is a better way to withstand security attacks. They developed several ontologies including a security attack

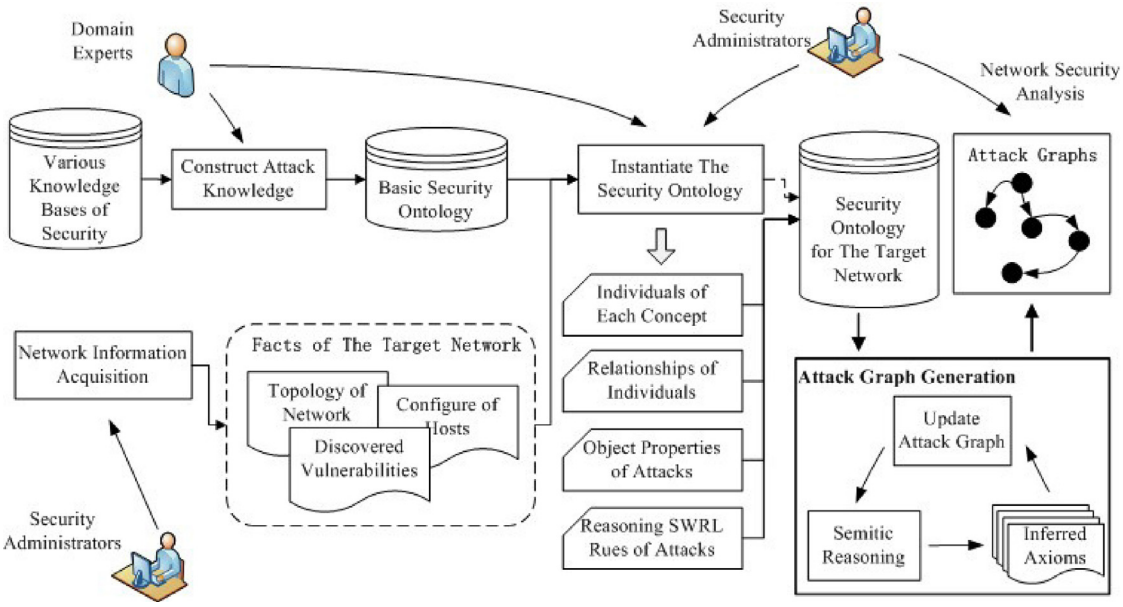


Fig. 1. Overview of the security assessment system.

ontology, a defence ontology, etc. as common vocabularies to share knowledge about attacks and countermeasures across distributed environments. Souag et al. [15] proposed a method that employs both security and domain ontologies to guide analysis of security requirements. Singhal et al. [16] presented an ontology for modelling enterprise level security metrics; this ontology contains many concepts including threats, vulnerabilities, security mechanisms, assets, and risks. The main application of this ontology is to collect data about the security of an enterprise system, and then query that data to generate reports. Literature [17] designed an ontological model that caters to generalized work for web services and the related protocols, vulnerabilities, and attacks. This ontology, equipped with reasoning features through SWRL, captures the context of web attacks and recommends mitigation measures. Li et al. [18] proposed an intrusion alert correlation system based on the XSWRL ((Extended Semantic Web Rule Language) ontology. Their system collects and translates network states and attack alerts into the ontology, then an XSWRL interpreter infers the attack process (a series of attack steps) to identify a possible attack. Foley and Fitzgerald [12] proposed a management system using security policy configurations, whereby knowledge about mitigation countermeasures can be used to guide the autonomic configuration of security mechanisms.

The aforementioned ontology-based schemes, however, do not focus on quantitative risk assessment of network security. To fill this gap, we propose an enhanced ontological model based on previous models [11,12]. The unique aspect of our model is the definition of a more concise ontology for modelling attack knowledge and inferring logical attack graphs. We also outline an approach for integrating our scheme with well-known security risk measurement systems [5,6], which achieves the goal of quantitative security risk assessment on enterprise networks.

3. System overview

Our proposal employs an ontology to define basic security concepts including “Device”, “Vulnerability”, “Attack”, etc. and the relationships between them. The security ontology, largely built on Foley et al.’s work [12], uses the SWRL rules for inferring potential attack behaviours, and various properties that could be acquired by adversaries via these attacks.

As illustrated in Fig. 1, the proposed framework for network security assessment implements different phases including construction of attack knowledge, network information acquisition, instantiation of the security ontology, and generation of the attack graph. We accomplish risk assessment and develop mitigation plans by integrating our attack graph with existing security risk management schemes. These major modules are described as follows:

- Constructing attack knowledge: The design of the security ontology is based upon expertise in the information security domain, as well as knowledge collected from various other sources. The basic ontology is always designed and constructed by domain experts, and constitutes a common vocabulary used to share knowledge across distributed parties.
- Network information acquisition: This module collects information related to the network, which is subjected to security assessment. The acquired data is stored for the next stage (Instantiate the security ontology). The work of information acquisition is performed by network administrators.
- Instantiating the security ontology: Security administrators (with the help of experts) construct an instance of the security ontology for performing assessment using the acquired network information. The tasks for this stage include the

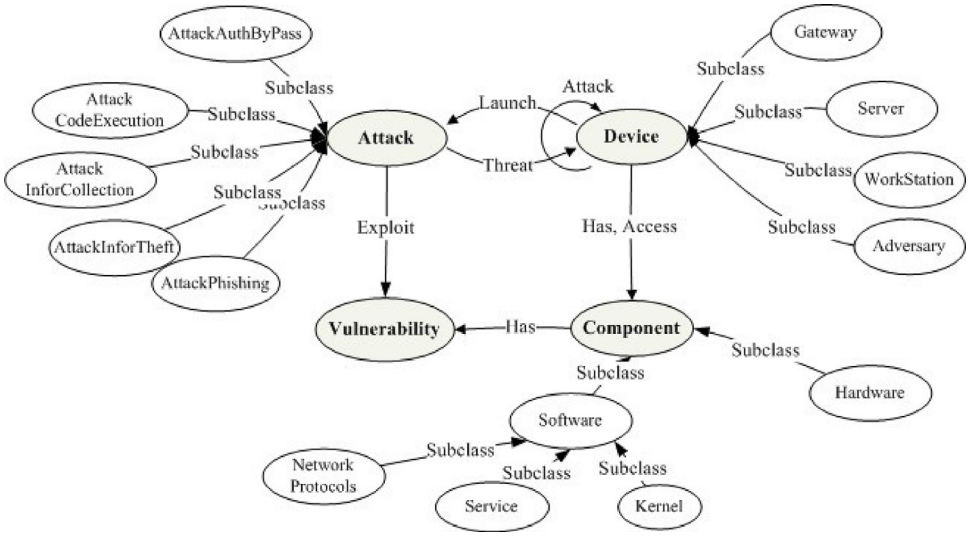


Fig. 2. Overview of the ontology for attack analysis graphs.

definition of individuals for each concept and representation of relationships through object properties. Once instantiation is completed, the new security ontology is ready for risk assessment.

- Attack graph generation: It is an iterative process to discover multistage and multi-host attacks. During each iteration, the reasoning engine takes individuals, object properties, and SWRL rules as inputs and infers new facts to represent potential attack paths and privileges obtained by the attacker. All inferred information is combined into the attack graph.
- Assessing risk and recommending mitigation measures: Computing a security risk metric and recommending mitigation measures are important for security management. Considering the many existing comprehensive schemes [5–7], we outline a general method to transform our attack graph into a MulVAL attack graph [2,6] in such a way that the proposed scheme can be integrated with the aforementioned models to produce risk assessment and mitigation plans.

We discuss the step "Constructing Attack Knowledge" in Section 4, and other modules in Section 5.

4. An ontology for security assessment

Our ontology has been developed keeping in view the following objectives:

- Detailed descriptions of all important security concepts, individuals, relationships, and reasoning rules related to security assessment.
- Explicitly specify information including topology, configurations, and discovered vulnerabilities of the target network using a formal language.
- Provide the ability to infer potential attacks and benefits that an attacker can obtain.
- Enable information sharing and reusability.

In reality, the construction of an ontology is done in two stages. In first stage, security domain experts design the fundamental conceptualizations for the attack ontological model. They are defined in the form of classes, properties, and individuals. Another important task is the definition of inference rules for predicting possible attack behaviours. The output from this phase is a shareable and reusable basic security ontology (see Fig. 1). In second stage (the "Instantiate the Security Ontology" step in Fig. 1), IT security administrators from enterprise networks define individuals within each concept according to the information collected from the targeted network. The instantiation step produces a special security ontology. After instantiation, with the inference ability of the security ontology, Algorithm 1 defined in Section 5.3.2 can discover possible multistage attacks. Sections 4.1 and 4.2 outline the construction of the basic security ontology in the first phase, while work for the second phase is described in Section 5.2.

4.1. Definition of concepts

Foley and Fitzgerald [12] defined a clear and concise security ontological model to represent the most basic concepts in security management. The design of our ontological model, illustrated in Fig. 2, is based on Foley et al.'s model. We modified this ontology definition according to our needs for modelling attack behaviours and used it to outline major security concepts and the relationships between them. The term "Device" represents any entity of the computer system, such as

Algorithm 1 Generation of Attack Graph.

Input: Instance of ontology: OT;Host of attacker: n_a;
Output: Attack Graph AG.

S, s_a and N_a are blank Sets;
$s_a \leftarrow getProperites(n_a)$; $S \leftarrow S \cup \{s_a\}$;
$N_a \leftarrow getHostsByProperites(n_a, s_a)$;
while true **do**
  $Jess.Reason(s_a, n_a, N_a)$;
  $S' \leftarrow Jess.inferredFacts()$;
  $LA \leftarrow getAllAttackProperties(S')$;
  for each $la \in LA$ **do**
    $a \leftarrow getRuleOfAttack(la)$;
    $s_{pre} \leftarrow pre(a)$; $s_{post} \leftarrow post(a)$;
    $S_{tmp} \leftarrow matchAllS_{pre}(S, s_{pre})$;
    for all $s_{tmp} \in S_{tmp}$ **do**
      $attack \leftarrow CreateAttackFrom(a)$;
      $pre(attack) \leftarrow s_{tmp}$;
      $post(attack) \leftarrow s_{tmp} \cup s_{post}$;
      $S \leftarrow S \cup \{post(attack)\}$; $A \leftarrow A \cup \{attack\}$;
    end for
    $S_{tmp} \leftarrow matchMinSatisfiedS_{pre}Set(S \backslash S_{tmp}, s_{pre})$;
    if $|S_{tmp}| > 0$ **then**
      $c \leftarrow createCombiningEdge()$;
      $pre(c) \leftarrow S_{tmp}$; $post(c) \leftarrow \{\cup s_i | s_i \in S_{tmp}\}$;
      $pre(a) \leftarrow post(c)$; $post(a) \leftarrow pre(a) \cup s_{post}$;
      $S \leftarrow S \cup \{post(a)\}$; $A \leftarrow A \cup \{a\}$; $C \leftarrow C \cup \{c\}$;
    end if
    $s_a \leftarrow s_a \cup post(a)$;
  end for
  $N_a \leftarrow getHostsByProperites(n_a, s_a)$;
  $CheckLoop(OT, N_a, n_a, s_a)$;
  if $N_a == \varnothing$ **then**
    break;
  end if
end while
Output $AG \leftarrow \{S, C, A\}$;

Web servers, gateways, workstations, etc. within the entire model, which may be the source or target of an attack. Individuals of concept Device have (hardware and software) components, and may be vulnerable owing to vulnerabilities in these components. As a result, those devices may be exposed to various individuals under the "Attack" concept. A device can also be the source of an attack, as it has access to vulnerable components and adversaries always launch attacks from compromised machines. The device class is further defined to include various kinds of device subclasses, such as Firewalls, Servers, Workstations, and Adversary Hosts.

$$Device \sqsubseteq \forall\, has.Component \sqcup \forall\, access.Component \sqcup \forall\, launch.Attack \sqcup \forall\, attack.Device$$

The "Attack" concept is a violation of security. An adversary could compromise hosts or privileges in the system by attempting to launch an attack. The attack class can be defined to include more kinds of attack subclasses based different attack modes, such as authentication bypass, code execution, information theft, phishing, etc. Any individual in the attack class is able to exploit one vulnerability.

$$Attack \sqsubseteq \forall\, exploit.Vulnerability \sqcap \forall\, threat.Device$$

The "Component" concept represents any software or hardware present in a device. A component may be defined as having vulnerabilities by using the "has" property. As a result, the device that uses the vulnerable components is also exposed to attacks.

$$Component \sqsubseteq \forall\, has.Vulnerability \sqcup \forall\, accessedBy.Device$$

A vulnerability is a flaw or security weakness in a component, that could be exploited by an attack.

$$Vulnerability \sqsubseteq \forall\, exploitedBy.Attack \sqcap \forall\, existsOn.Component$$

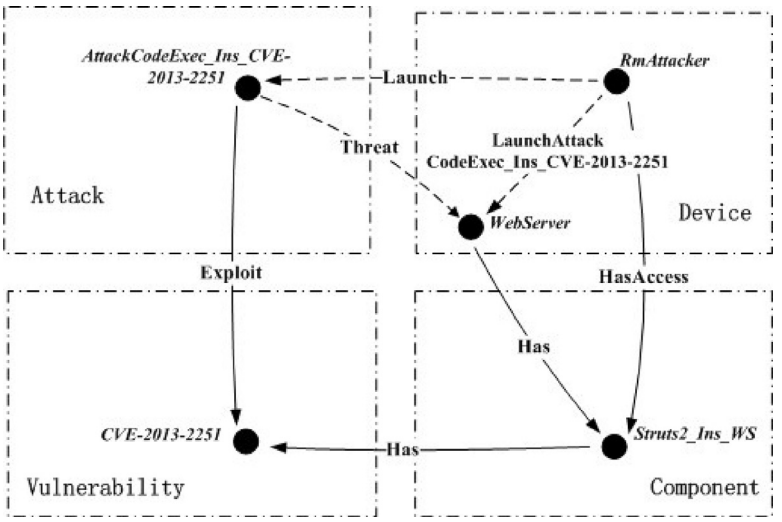


Fig. 3. An instance of a concrete attack behaviour.

OWL (Web Ontology Language) properties represent the relationships between individuals. There are two main types of properties, object properties and datatype properties [19]. Object properties are binary relationships between two individuals that are defined in the form of *Property*(*Domain, Range*). For example, the property *exploit*(*Attack, Vulnerability*) links the individuals of Attack and Vulnerability (seeing Fig. 3). We focus exclusively on object properties. For simplicity, we specify an individual of a concept using the notation $indv_{concept}$, and use the abbreviations "Dev", "Cmpt" and "Vul" to represent Device, Component , and Vulnerability respectively. The most common object properties in our system are as follows:

- *exploit*(*Attack, Vul*) states that $indv_{Attack}$ exploits $indv_{Vul}$,
- *has*(*Dev, Cmpt*) states that $indv_{Dev}$ has $indv_{Cmpt}$,
- *has*(*Cmpt, Vul*) states that $indv_{Cmpt}$ has $indv_{Vul}$,
- *hasAccess*(*Dev, Cmpt*) states that $indv_{Dev}$ is able to access $indv_{Cmpt}$,
- *hasAccount*(*Dev, Cmpt*) states that $indv_{Dev}$ has account of $indv_{Cmpt}$,
- *hasPrivilegeUser*(*Dev, Dev*) states that $indv_{Dev}$ has user privilege on $indv_{Dev}$,
- *hasPrivilegeRoot*(*Dev, Dev*) states that $indv_{Dev}$ has root privilege on $indv_{Dev}$,
- *hasShellRemote*(*Dev, Dev*) states that $indv_{Dev}$ has a remote shell from $indv_{Dev}$,
- *hasCompromiseFiles*(*Dev, Cmpt*) states that $indv_{Dev}$ compromises the integrity of $indv_{Cmpt}$,
- *hasFilesDependence*(*Dev, Cmpt*) states that $indv_{Dev}$ trusts and frequently accesses the data/files of $indv_{Cmpt}$,
- The "launchAttack" properties named in the form of "$launchAttack\{type\}\{cve\_code\}(Dev, Dev)$" express that a host could launch the *type* of attack that exploits *cve_code* to target the specified machine.

4.2. Modelling attack behaviours

Previous works have used SWRL rules for identifying attack payloads [17] and aiding detection of proper security behaviours [12]. These works inspired us to apply the inference ability of a security ontology to identification of possible attacks caused by existing vulnerabilities.

After defining the fundamental concepts, we represent attacks in the basic security ontology using several steps: constructing individuals from the "Vulnerability" and "Attack" concepts, creating a vocabulary of object properties to express the prerequisites for and consequences of an attack, and defining the SWRL rules for inferring potential attacks and adversarial gains. The design of such a model is based the expertise of information security experts and information collected from various other sources, such as CVE (Common Vulnerabilities and Exposures) [20] and the Exploit Database [21]. Fig. 3 illustrates an instance of an attack that exploits the vulnerability "CVE-2013-2251". We use the notation $Indvs_{concept}$ to express the individuals contained in a particular concept. Using the attack fragment that exploits vulnerability "CVE-2013-2251", the entire process of modelling an attack is as follows:

Define individuals of each concept:

$$Server(WebServer), Adversary(RmAttacker) \in Indvs_{Dev}$$
$$Service(Struts2\_ins\_WS) \in Indvs_{Cmpt}$$
$$Vulnerability(\text{CVE-2013-2251}) \in Indvs_{Vul}$$
$$AttackCodeExec(attak\_ins\_\text{CVE-2013-2251}) \in Indvs_{Attack}$$

Create object properties to depict the potential attack:

$$launchAttackCodeExec\_ins\_\text{CVE-2013-2251}(Dev, Dev)$$

The "*launchAttack*..." property states that the attacker is able to launch an attack targeting vulnerability CVE-2013-2251, which exists on the targeted machine.

Define relationships between individuals.

$$has(webServer, Struts2\_ins\_WS);\ has(Struts2\_ins\_WS, \text{CVE-2013-2251});$$
$$hasAccess(Rm\_Attacker, Struts2\_ins\_WS);\ exploit(attak\_ins\_\text{CVE-2013-2251}, \text{CVE-2013-2251})$$

Define the SWRL rule for expressing the attack: The rule named "Rule-launchAttackCodeExec_Ins_CVE-2013-2251" is described as:

$$Device(?a) \wedge Component(?b) \wedge Vulnerability(?c) \wedge sameAs(?c, \text{CVE-2013-2251})$$
$$\wedge has(?a, ?b) \wedge has(?b, ?c) \wedge AttackCodeExec(?d) \wedge exploit(?d, ?c)$$
$$\wedge Adversary(?e) \wedge hasAccess(?e, ?b) \longrightarrow$$
$$launchAttackCodeExec\_Ins\_\text{CVE-2013-2251}(?e, ?a) \wedge hasCompromiseFiles(?e, ?b)$$

This SWRL rule states the prerequisites for and consequences of the "CodeExec" attack. Using the aforementioned definition of relationships, a reasoning engine such as Jess can infer new facts:

$$launchAttackCodeExec\_Ins\_\text{CVE-2013-2251}(RmAttacker, WebServer)$$
$$\wedge hasCompromiseFiles(RmAttacker, Struts2\_Ins\_WS)$$

The inferred *launchAttack* relation is represented by a dotted line in Fig. 3.

Along with the rules of "launchAttack", we defined several "transitivity" rules to express the fact that an adversary could obtain privileges on the compromised machine. The following rule named "Rule-TransitivityAccessUser" means that an adversary (host a) is able to "inherit" all "hasAccess" relationships of the victim (host b) after compromising user privilege and obtaining remote shell access via a successful attack. Owing to space limitations, we have omitted the representations of the other "transitivity" rules.

$$Device(?a) \wedge Device(?b) \wedge Device(?d) \wedge Component(?c) \wedge has(?d, ?c)$$
$$\wedge hasPrivilegeUser(?a, ?b) \wedge hasShellRemote(?a, ?b) \wedge hasAccess(?b, ?c)$$
$$\wedge different From(?a, ?d) \longrightarrow hasAccess(?a, ?c)$$

5. Security assessment using semantic reasoning

In this section, we present detailed discussion on a method for generating attack graphs for network security assessment. This approach uses the reasoning ability of the security ontology proposed in Section 4 to discover potential multistage attack behaviours. The major processes (shown in Fig. 1) are outlined in the following subsections.

5.1. Network information acquisition

A security administrator collects information including the topology and configurations of every host and vulnerability on the network for security assessment. The methodologies of traditional penetration testing can be used to identify vulnerable services running on network machines. OpenVAS [22] is a tool that was designed for probing a server or host to detect active services or applications and their potential vulnerabilities. One can extract network data including lists of hosts, services, and discovered vulnerabilities from the scan reports generated by OpenVAS.

5.2. Instantiating a security ontology for the targeted network

Using the results from the "Network Information Acquisition" stage, security administrators instantiate a concrete security ontology for the network subjected to security assessment. Generally, the following steps are required:

- Create individuals of the *Device, Component* and *Vulnerability* concepts.
- Setup relationships between individuals using the object properties *has, hasAccess, hasAccount, hasFilesDependence*, etc.
- If certain discovered vulnerabilities are not predefined in the basic security ontology, administrators must model the new attack behaviours that exploit these vulnerabilities following the steps illustrated in Section 4.2. This process may require guidance from security domain experts.

The concrete instantiation of a security ontology can be performed by using the dynamic Protégé's API [23] or the Graphical User Interface of this tool.

5.3. Attack graph generation

To generate a full attack graph, we need to simulate attack behaviours of penetrating the network. To accomplish this, the attacker continuously exploits resources to launch attacks that compromise more resources with every iteration. This process continues until there are no new targets remaining. We formally define the attack graph and provide a case study to illustrate this process in following subsections.

5.3.1. Definitions of an attack graph

The resources compromised by an adversary are useful attack data. An attribute-set helps to represent these compromised resources. We define the *AttributeSet* as follows:

Definition 1 (Attribute Set)**.** A set of object properties defined in the security ontology.

An attack represents that an adversary has exploited the attribute set S_{pre} to compromise more resources. The attribute set after a successful attack is denoted as S_{post}. Informally, an attack relates different attribute sets by defining cause-consequence relationships between them.

Definition 2 (Attack)**.** Let S be a set of *AttributeSet*. Given that S_{pre}, $S_{post} \in S$ and $S_{pre} \subset S_{post}$, we define a: $S_{pre} \rightarrow S_{post}$ as an *Attack*. S_{pre}, S_{post} are called a precondition and postcondition of the attack a respectively, denoted by $pre(a)$ and $post(a)$.

A *Combination* represents the fact that an attacker could obtain attributes from various attack paths.

Definition 3 (Combination)**.** Let S be a set of *AttributeSet*. We define $c \leftarrow \{S_1 \cup \ldots \cup S_n\}$, where $S_1, \ldots, S_n \in S$, as a combination of multiple *AttributeSets*.

An attack graph (*AG*) expresses multi-stage, multi-host attack scenarios using sets of *AttributeSet*, *Attack* and *Combination*.

Definition 4 (Attack Graph (AG))**.** Let S be a set of *AttributeSet*, A be a set of *Attack* and C be a set of *Combination*. An Attack Graph is a tuple $AG = (S, A, C)$. Attack Graph AG is an acyclic directed graph where $S \cup C$ is the vertex set and A is the edge set.

5.3.2. Algorithm for generating the attack graph

We use an algorithm (refer to Algorithm 1) based on the reasoning engine JESS [9] to generate attack graphs. The algorithm takes an instance of the security ontology for the target network and a machine controlled by the adversary as inputs, and generates an attack graph in the form of a tuple {*S, C, A*}. The algorithm uses a forward chaining approach in which it begins at the initial conditions and attempts to gain additional properties by continuously attacking vulnerable hosts in the network.

In line 2 of Algorithm 1, the method *getProperites* returns the initial object properties of the adversary. The method *getHostsByProperites* returns all target hosts and binary relationships in the set s_a. The major loop continues running (starts on line 4) until no new machines can be discovered from the current properties of the adversary (lines 28–31).

The method *Jess.Reason* (line 5) takes the adversary n_a, the properties of the adversary s_a, and all machines that are accessible by n_a (through s_a) as inputs, and invokes JESS to perform inference. The inferred information (object properties) is returned by *Jess.inferredFacts. getAllAttackProperties* extracts all *launchAttack* relationships that constitute possible attacks from the inferred properties. The method *getRuleOfAttack* returns the SWRL rule related to the attack *la*. The $matchAllS_{pre}$ method retrieves all property sets S_{tmp} from the set S such that $s_{tmp} \in S_{tmp}$ satisfies $s_{pre} \subseteq s_{tmp}$. At line 16 of Algorithm 1, we generate the attack edges that start from a single property set that exists in S. In the event that the preconditions of attack *la* can be satisfied by multiple property sets where $s_{pre} \not\subseteq s_1$, $s_{pre} \not\subseteq s_2$, ..., $s_{pre} \not\subseteq s_m$ and $s_{pre} \subseteq \{s_1 \cup s_2 \cup \ldots \cup s_m\}$, lines 19 through 24 will create a combination edge c. The expression $s_a \leftarrow s_a \cup post(a)$ in line 25 denotes that the properties of the adversary are expanded by a successful attack.

The *checkLoop* method removes hosts whose vulnerabilities have all been exploited from the set of target machines N_a, this effectively prevents our algorithm from falling into an infinite loop.

5.3.3. Integrating our AG with existing schemes for risk management

One major concern with achieving the desired level of security is the cost incurred in network organization. To achieve an acceptable trade-off between the targeted security level and the projected mitigation costs, administrators need efficient approaches for computing network security risk metrics and recommending mitigation measures. Considering the existence of many comprehensive assessment schemes [5–7], we propose a general method for transforming our attack graph into a MulVAL attack graph that has been adopted in Homer et al.'s work [6], and is compatible with various risk assessment models [5,7]. Through transformation of the generated attack graph, the proposed scheme can be integrated with the aforementioned security risk management models. In the "Case Study" section, we provide an example of transforming the generated attack graph (Fig. 5b) into a MulVAL attack graph (Fig. 6). The attack graph conversion steps are as follows:

- Convert attack edges to attack-step nodes, which are represented in the MulVAL AG as circular AND-nodes. Merge the combination node and its subsequent AND-node. Transform AttributeSet nodes ($S_0, S_1, \ldots, S_n$) into privilege nodes, which

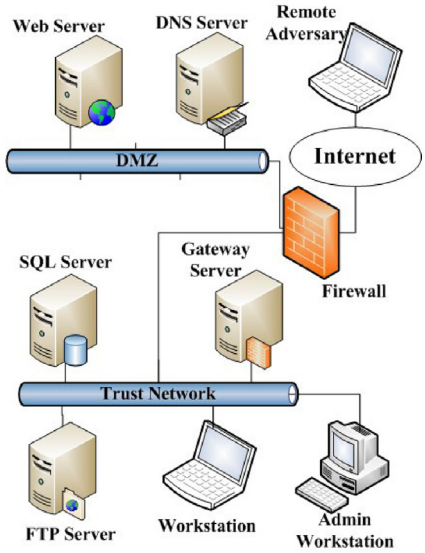


Fig. 4. A test-bed network.

Table 1
Access properties of the test network.

From host	To host	Access component
Workstation	FTP server	FTP service
	WEB server	WEB service
	DNS server	DNS service
	SQL server	SQL service
Admin workstation	FTP server	FTP service
	WEB server	WEB service
	DNS server	DNS service
	SQL server	SQL service
Gateway server	Workstation	RemoteDesktop
Web server	SQL server	SQL service
Remote attack	Gateway server	OpenSSL

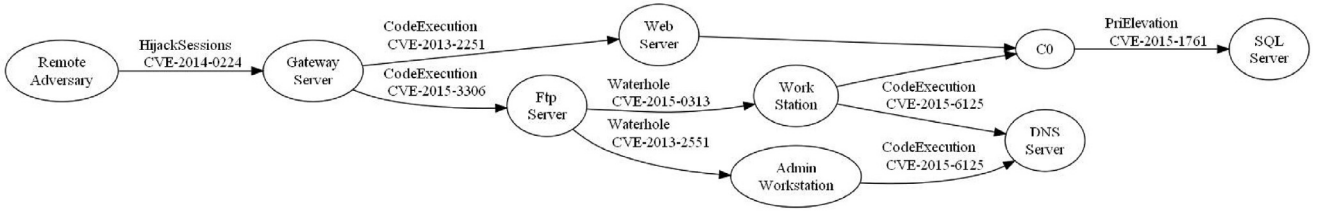
are represented within the MulVAL AG as diamond-shaped OR-nodes. Each privilege node only retains attributes that can be derived from its predecessor.

- Split privilege nodes with out-degree greater than 1. We assume that this type of privilege node P_i has privilege set $S_i = \{s_0, \ldots, s_n\}$ and its subsequent nodes (*out* edges) are the attack-step node set $\{A_0, \ldots, A_m\}$, where $pre(A_0) = \{s_0, \ldots, s_k\}$, $\ldots$, $pre(A_m) = \{s_m, \ldots, s_n\}$. We then split node P_i to $m+1$ nodes $\{P_{i0}, \ldots, P_{im}\}$, where $P_{ij} = pre(A_j)$ and the subsequent node of P_{ij} is node A_j.
- Merge similar nodes. Merge privilege nodes if they have the same privilege set, then merge attack-step nodes if they have the same parent and children nodes (connecting with *in* and *out* edges).

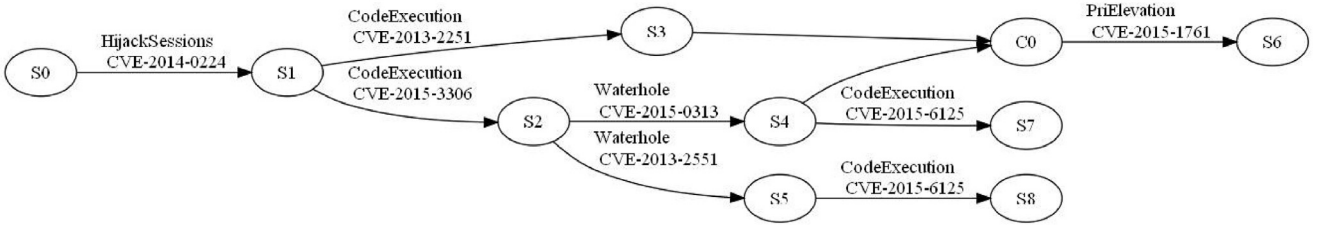
5.3.4. Case study

Fig. 4 depicts the test network used for this study. The network consists of seven hosts split into two subnets and an adversary host that accesses the test network through the internet. A tri-homed firewall is installed with preset policies to protect the two local subnets. The firewall policies are designed to prevent remote access to the internal hosts. All servers in the DMZ (Demilitarized Zone) receive service requests passively and only respond to the sender as needed. However, to complete the necessary functions of the web service, the web server is allowed to access a SQL server located in the trusted zone through a designated channel. Local machines in the trusted zone are located behind a NAT (Network Address Translation) server and all communications to external hosts are delivered through the Gateway server. All local workstations, including the administrator machine, have remote desktop enabled to facilitate remote operations for company employees working at remote sites. These remote connections are monitored by SSHD (Secure Shell Daemon) installed on the Gateway server.

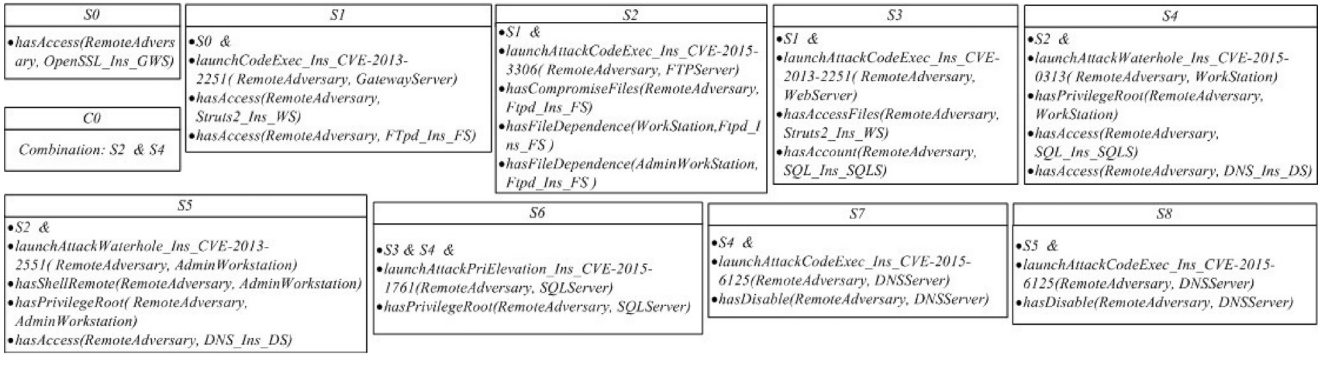
Initial access properties of the test network are listed in Table 1. We assume that the vulnerabilities listed in Table 2 exist in this test case. The knowledge of these vulnerabilities (column Type of attack in Table 2) is obtained from CVE [20] and SCAP [24].



(a) Attack steps



(b) Attack graph



(c) Sets of object property

Fig. 5. Evaluation results of classifiers.

Table 2
Configuration list for the test network.

Host	Component	Vulnerability	Type of attack
Workstation	Application	CVE-2015-0313	Execute arbitrary code, remote-2-root
Admin workstation	Application	CVE-2013-2551	Execute arbitrary code, remote-2-root
Gateway server	OpenSSL	CVE-2014-0224	Hijack sessions, authentication bypass
Web server	Apache Struts2	CVE-2013-2251	Execute arbitrary code; Information leakage
FTP server	ProFTPD	CVE-2015-3306	Read and write to arbitrary files
SQL server	SQL service	CVE-2015-1761	Elevation of privilege, remote-2-root
DNS server	DNS service	CVE-2015-6125	Execute arbitrary code, disruption of service

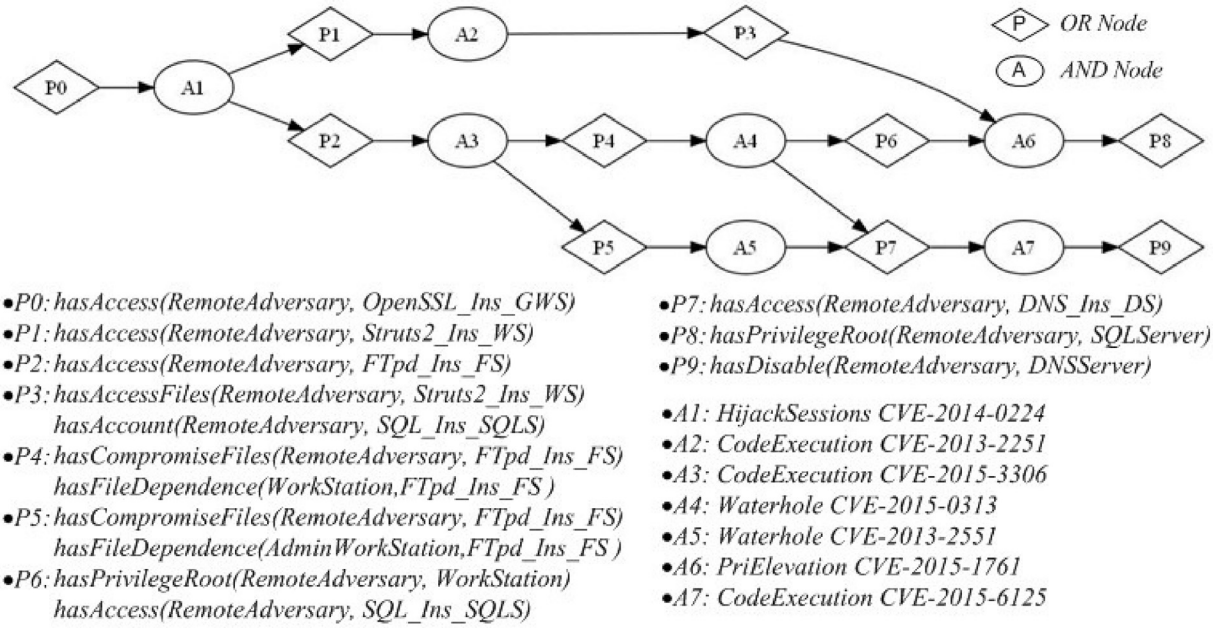


Fig. 6. Transform AG for risk assessment and mitigation.

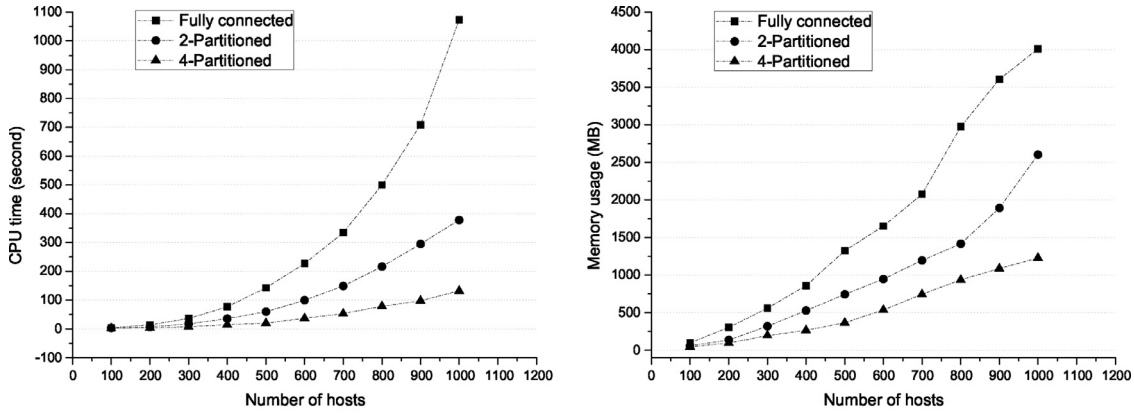
The settings for the test network produce the attack graph shown in Fig. 5. The final output of Algorithm 1 includes the attack graph in a visualized representation (Fig. 5(b)) as well as descriptions of properties set (Fig. 5(c)). To clearly show the progression of the attack, we provide a flow diagram of the attack process (Fig. 5(a)), which illustrates how an adversary can attack and compromise several objectives in the test network through multiple hosts and stages. Fig. 6 shows how the generated attack graph (in Fig. 5(b)) is transformed into a "MulVal" attack graph.

6. Evaluation of performance

The process of computing an attack graph consists of exploring a network and invoking a reasoning method. In this section, we first analyse the complexity of the algorithm for generating the attack graph, and then evaluate the performance of the reasoning method through experimentation.

6.1. Complexity of computing the attack graph

For this example, we use a network with N hosts and m vulnerabilities. In the main loop of Algorithm 1 (line 4 - 32), we complete the generation of an attack graph for an entire network through multiple iterations of discovery of sub-networks that can be directly accessed by an adversary. As an adversary is able to spread their influence through the network using compromised hosts, the size of the sub-network, denoted as N_s, changes with each iteration. The maximum steps for a multistage attack, denoted H_{max}, equals the maximum number of iterations. The complexity of *Jess.Reason*() is assumed to be a function of $(N_s^k \times m)$, denoted $R(N_s^k \times m)$, where $k \geq 1$ and m is the number of vulnerabilities that exist on each host. The cost of execution for the *while*-loop in Algorithm 1 is approximated by: $C_{alg1} = O(H_{max} \times (R(N_s^k \times m) + N_s))$, where $N_s \in [1, \ldots, N]$. Note that the maximum number of attack stages, H_{max}, and the maximum number of vulnerabilities on each host are the limiting values. The complexity of Algorithm 1 can be further simplified to $C_{alg1} = O(R(N_s^k))$, which means that C_{alg1} is largely dependent on the complexity of the reasoning method, *Jess.Reason*().



(a) CPU Time of Reasoning Method (b) Memory Usage of Reasoning Method

Fig. 7. Evaluation results of reasoning method.

According to the description by literature [9], it is hard to accurately determine the computational complexity of the method *Jess.Reason*(). In the next subsection, we attempt to estimate the performance of the reasoning method in terms of CPU time and memory usage with different network sizes.

6.2. Experimental results

The reasoning method of Jess was tested on a laptop PC with the Intel Core i5-5200U CPU (2.2 GHz) and 4GB of RAM. The operating system used was Microsoft Windows 8.1. The tested version of Jess was 7.1P2. Network configuration information was simulated for a variety of network sizes and topologies. The number of vulnerabilities for every host was set to four, and all vulnerabilities on active services were exploitable remotely. The test case for each network configuration was presented as an OWL model that was exported into a script for batch execution in the Jess reasoning engine.

The topologies for the test networks included "fully-connected", "2-partitioned" and "4-partitioned" schemes. The "fully-connected" network topology simulation allowed network accessibility for all vulnerable service components between every pair of hosts. The "n-partitioned" ($n = 2, 4$) topologies were simulated with n approximately equal sized fully-connected networks, which were connected to each other by a single pair of machines (neither being the adversary machine). The number of hosts for our experiments were set in increments of 100 up to 1000.

Fig. 7(a) shows the computational overhead in CPU time for each simulated network. The worst case was the fully connected network, with asymptotic CPU time nearing $O(N^2)$, where N is the number of hosts. Fig. 7(b) shows the memory usage for the same set of test networks. The values for memory consumption were obtained from Windows task manager. All test cases have asymptotic memory usage slightly below $O(N^2)$, with the worst-case scenario again being the fully connected network.

7. Conclusion

We proposed an innovative ontology and graph-based approach for network security assessment. An ontology is designed to represent security knowledge such as that of assets, vulnerabilities, attacks, relationships, and the inference rules for identifying possible attacks. Afterwards, we designed an efficient system framework and attack graph generation algorithm for discovering logical attack paths in enterprise networks. Owing to the existence of various comprehensive schemes for network security risk assessment, we provided a method for transforming our attack graph into a MulVAL attack graph. This facilitates the ability to compute risk metrics and recommend optimal mitigation plans by integrating other available security schemes. As discussed in Section 6.1, the computational complexity for the proposed attack graph generation algorithm is largely dependent on that of the reasoning engine. According to our experimental results, assuming that four vulnerabilities exist on each host, the computational complexity in terms of both CPU time and memory usage nears $O(N^2)$ in the worst-case scenario for networks with less than 1000 machines. This indicates that our proposal facilitates scalable security analysis for enterprise networks.

The security ontology provides a common and unambiguous semantic language for representing information security knowledge and serves as a basis for information exchange between domain experts and enterprise network administrators. Using the Open World Assumption features of the ontology and SWRL, newly discovered vulnerabilities and attacks can be modelled quickly. This enables enterprise administrators to accomplish security risk assessment tasks and react to new threats.

Going forward, we intend to construct a security countermeasure ontological model using public security specifications such as NIST (National Institute of Standards and Technology) Special Publications, and to integrate this countermeasure ontology with risk assessment to support security policy management and decision making.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61402117) and Research Project from Ministry of Public Security of China (No. 2016GABJC25).

References

[1] Almohri H, Yao D, Watson L, Ou X. Security optimization of dynamic networks with probabilistic graph modeling and linear programming. IEEE Trans Dependable Secure Comput 2016;13(4):474–87.
[2] Ou X, Boyer WF, McQueen MA. A scalable approach to attack graph generation. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. CCS '06. New York, NY, USA: ACM; 2006. p. 336–45.
[3] Jajodia S, Noel S, O'Berry B. Topological analysis of network attack vulnerability. Boston, MA: Springer US; 2005. p. 247–66.
[4] Ou X, Govindavajhala S. Mulval: a logic-based network security analyzer. In: In 14th USENIX Security Symposium; 2005. p. 113–28.
[5] Poolsappasit N, Dewri R, Ray I. Dynamic security risk management using bayesian attack graphs. IEEE Trans Dependable Secure Comput 2012;9(1):61–74.
[6] Homer J, Zhang S, Ou X, Schmidt D, Du Y, Rajagopalan SR, et al. Aggregating vulnerability metrics in enterprise networks using attack graphs. J Comput Secur 2013;21(4):561–97.
[7] Ghosh N, Chokshi I, Sarkar M, Ghosh SK, Kaushik AK, Das SK. Netsecuritas: An integrated attack graph-based security assessment tool for enterprise networks. In: Proceedings of the 2015 International Conference on Distributed Computing and Networking. ICDCN '15. New York, NY, USA: ACM; 2015. p. 30:1–30:10.
[8] Allemang D, Hendler J. Semantic web for the working ontologist: effective modeling in RDFS and OWL. Elsevier; 2011.
[9] Hill EF. Jess in action: java rule-based systems. Greenwich, CT, USA: Manning Publications Co.; 2003.
[10] Baader F. The description logic handbook: theory, implementation and applications. Cambridge university press; 2003.
[11] Vorobiev A, Bekmamedova N. An ontology-driven approach applied to information security. J Res Pract Inf Technol 2010;42(1):61–76.
[12] Foley SN, Fitzgerald WM. Management of security policy configuration using a semantic threat graph approach. J Comput Secur 2011;19(3):567–605.
[13] Common vulnerability scoring system. 2016. URL: http://www.first.org/cvss.
[14] Arbanas K, Čubrilo M. Ontology in information security. J Inf Organ Sci 2015;39(2):107–36.
[15] Souag A, Salinesi C, Wattiau I, Mouratidis H. Using security and domain ontologies for security requirements analysis. In: Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual; 2013. p. 101–7.
[16] Singhal A, Wijesekera D. Ontologies for modeling enterprise level security metrics. In: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. CSIIRW '10. New York, NY, USA: ACM; 2010. p. 58:1–58:3.
[17] Razzaq A, Anwar Z, Ahmad HF, Latif K, Munir F. Ontology for attack detection: an intelligent approach to web application security. Comput Secur 2014;45:124–46.
[18] Li W, Tian S. An ontology-based intrusion alerts correlation system. Expert Syst Appl 2010;37(10):7138–46.
[19] Horridge M. A practical guide to building owl ontologies using protégé 4 and co-ode tools edition1.3. The University Of Manchester; 2011.
[20] Common vulnerabilities and exposures (cve). 2016. URL: https://cve.mitre.org/.
[21] Exploit database. 2016. URL: https://www.exploit-db.com/.
[22] Openvas. 2016, 7. URL: http://www.openvas.org/.
[23] Knublauch H, Fergerson RW, Noy NF, Musen MA. The Protégé OWL plugin: an open development environment for semantic web applications. Berlin, Heidelberg: Springer Berlin Heidelberg; 2004. p. 229–43.
[24] Security content automation protocol chinese community. URL: http://www.scap.org.cn/index.html.

Songyang Wu is an associate professor at The Third Research Institute of Ministry of Public Security, China .Vice director. He received his Ph.D. Degree in computer science from TongJi University, China in 2011. His current research interests are in information security, cloud computing and digital forensics.

Yong Zhang, received his Ph.D. Degree in computer science from East China Normal University, China in 2013. He is a research member in The Third Research Institute of Ministry of Public Security, Shanghai, China. His research interests include information security, digital forensics and cryptography.

Wei Cao received master degree in computer science from Shandong University of Science and Technology. He is a research member in The Third Research Institute of Ministry of Public Security, Shanghai, China. His research interests include network security and big data.