ONTIDS: A Highly Flexible Context-Aware and Ontology-Based Alert Correlation Framework

Alireza Sadighian[1(✉)], José M. Fernandez[1], Antoine Lemay[1], and Saman T. Zargar[2]

[1] Department of Computer and Software Engineering, École Polytechnique de Montréal, Montréal, Canada
{alireza.sadighian,antoine.lemay,jose.fernandez}@polymtl.ca
[2] School of Information Sciences, University of Pittsburgh, Pittsburgh, PA, USA
stzargar@sis.pitt.edu

Abstract. Several alert correlation approaches have been proposed to date to reduce the number of non-relevant alerts and false positives typically generated by Intrusion Detection Systems (IDS). Inspired by the mental process of the contextualisation used by security analysts to weed out less relevant alerts, some of these approaches have tried to incorporate contextual information such as: type of systems, applications, users, and networks into the correlation process. However, these approaches are not flexible as they only perform correlation based on the narrowly defined contexts. information resources available to the security analysts while preserving the maximum flexibility and the power of abstraction in both the definition and the usage of such concepts, we propose ONTIDS, a context-aware and ontology-based alert correlation framework that uses ontologies to represent and store the alerts information, alerts context, vulnerability information, and the attack scenarios. ONTIDS employs simple ontology logic rules written in Semantic Query-enhance Web Rule Language (SQWRL) to correlate and filter out non-relevant alerts. We illustrate the potential usefulness and the flexibility of ONTIDS by employing its reference implementation on two separate case studies, inspired from the DARPA 2000 and UNB ISCX IDS evaluation datasets.

Keywords: Intrusion detection · Alert correlation · Ontology · Context-aware

1 Introduction

IDS collect data from the IT infrastructure and analyse it to identify ongoing attacks. Various IDS types have been proposed in the past two decades and commercial off-the-shelf (COTS) IDS products have found their way into Security Operations Centres (SOC) of many large organisations. Nonetheless, the usefulness of the single-source IDS has remained relatively limited due to two main factors: their inability to detect new types of attacks (for which new detection rules or training data are unavailable) and their often very high false positive rates.

J.-L. Danger et al. (Eds.): FPS 2013, LNCS 8352, pp. 161–177, 2014.
DOI: 10.1007/978-3-319-05302-8_10,

One of the approaches that has been suggested to address these challenges is the *alert correlation*, where the alert streams from several different IDS or more generally various alert sensors are jointly considered and analysed to provide a more accurate picture of the threats. When individual IDS examine the same type of data, their alert correlations is called the *homogeneous IDS correlation*. In fact, the majority of the research projects and real-world deployments of the correlation approaches involve the analysis of the alerts generated by different network IDS (NIDS), such as: SNORT or Bro, examining network traffic streams at different network locations.

Nonetheless, most of the attacks, whether automated malware infections or manual network intrusions, do not leave traces only on the captured network traffic but also on the host-based IDS (HIDS), on other security products, and sometimes even on non security-related logs of commodity or corporate applications. This fact has been successfully exploited by security analysts worldwide to detect sophisticated attacks by automatically or manually correlating these versatile information and alert sources. Since various sensors examine different types of events and raw data sources, their alert correlations is called the *heterogeneous alert correlation*.

One of the main challenges for heterogeneous correlation is the integration of data from various alert sources with potentially different formats and semantics. At the same time, sensor-specific attributes must also be retained in order to preserve the ability for the security analyst to *drill down* and refine his analysis. For instance, for finding root causes, determining attack type, attack objectives, and *etc.*

Moreover, one of the fundamental alert management principles is that the security analysts must be able to understand the alerts and the *context* they are generated in. This is what allows one to consider the relevance and relative importance of the alerts. Unfortunately, security analysts often need to manually gather such information from multiple systems to feed the correlation process in order to integrate and validate the alerts and identify the consequences of any intrusion. This is why certain researchers have proposed approaches to automatically include such contextual information into the alert correlation process, approaches that are referred to as the *context-aware* alert correlation approaches. The simple and intuitive idea here to reduce the false positive is that the alerts that are related to a certain type of attack are only relevant if the context in which they happen is indeed vulnerable to that type of attack. Hence, the context aware alert correlation must consider the vulnerability information, and potentially the attack models in order to be useful.

The difficulty in implementing the aforementioned approaches resides in integrating the information into a data model that is generic enough to allow the global view of the data while retaining maximum data granularity for drill-down analysis. The flexibility and extensibility of the data model is thus a key requirement of any such approaches. Finally, the method by which security analysts extract information and intelligence from such data stores must itself also be flexible and extensible. It must support generic simple queries and detailed analysis.

To this purpose, in this paper, we present ONTIDS that relies heavily on the ontologies and the ontology description logic to accomplish these goals. We mainly describe how ONTIDS addresses the high positive rate problem of the existing IDS. ONTIDS has the following characteristics:

1. It performs heterogeneous alert correlation to detect complex attacks that might leave traces in different types of sensors.
2. It includes a set of comprehensive but extensible ontologies, allowing correlation and reasoning with information collected from various resources, including context information, vulnerability databases and assessments, and attack models.
3. It can be used to seamlessly and automatically implement various alert correlation approaches on the same data model.
4. It can be applied in different deployment and analysis contexts, from simple to complex IT infrastructures, generic threat detection to complex attack forensics analysis.

The rest of this paper is organized as follows. In Sect. 2, we discuss the related work. In Sect. 3, we present our framework in detail. In Sect. 4, we demonstrate the flexibility of our framework by describing a reference implementation and applying it to the analysis of two different case studies. We conclude, in Sect. 5, by describing the limitations of these case studies to conclusively demonstrate the reduction of the non-relevant alerts and the false positives.

2 Related Work

In a keynote publication, *Valeur et al.* propose a correlation approach consisting of ten steps, which we will use later to exemplify our generic framework in Sect. 3.5. This is perhaps the most comprehensive approach, with other work concentrating only on one particular aspect of the correlation process, such as the alert fusion [1,2] or the attack thread reconstruction [3]. From a classification point of view, *Cuppens et al.* classify the attack reconstruction approaches into two categories [4]:

1. *Explicit alarm correlation*, which relies on the capabilities of security administrators to express logical and temporal relationships between alerts in order to detect complex multi-step attacks. For instance, *Morin et al.* propose an explicit correlation scheme based on the formalism of the *chronicles* [5].
2. *Implicit alarm correlation*, which is based on employing the machine learning and the data mining techniques to fuse, aggregate, and cluster the alerts for the alert correlation and the intrusion detection purposes. For instance, *Chen et al.* employ the Support Vector Machine (SVM) and the co-occurrence matrix in order to propose a masquerade detection method [6]. In [7], Raftopoulos performs the log correlation using C4.5 decision tree classifiers after analysing the diagnosis of 200 infections that were detected within a large operational network.

One of the shortcomings of the approaches in both categories is that they do not take into account all the available and important information resources such as contextual and vulnerability information. The contextual information has proved to be useful in better identifying specific alerts or in improving the IDS efficiency. *Gagnon et al.*, in [8], have studied the use of target configuration as the context information in order to identify the non-critical alerts. The Workload-aware Intrusion Detection (WIND) proposal by *Sinha et al.* combines the network workload information with the Snort rules to improve Snort's efficiency [9]. Unfortunately, these studies only consider partial contextual information such as target configuration or network traffic and do not allow for the inclusion of other types of contextual concepts.

Ontologies are knowledge representation models that allow the description of the concepts, their attributes, and the inheritance or the association relationships between them. In addition, various types of ontologies have formal description languages that allow for the definition of the complete reasoning logic that are machine-interpretable and solvable. Hence, some researchers have proposed the ontology-based alert correlation approaches for the alert correlation process. capabilities for detecting new types of attacks such as multi-step distributed attacks and various distributed denial of service (DDoS) attacks. The proposed ontologies, however, only include general security concepts and no discussion on how they can be adapted to different contexts. The Intrusion Detection and Diagnosis System (ID2S) proposed by *Coppolino et al.* uses ontologies as well to correlate the detection information at several architectural levels for further intrusion symptom analysis [11].

In summary, while *Valeur et al.* [12] provides a good generic framework for alert correlation into which various other attack reconstruction approaches can be incorporated [3,6,7,13], none of these attacks contrast the alert information with the context. On the other hand, those alert correlation approaches have limited notions of the context that cannot be readily extended and they do not perform attack reconstruction. Finally, the correlation approaches that have employed ontologies have not fully taken the advantage of their expressive power in terms of data modelling and logic reasoning.

Motivated by these shortcomings and in order to provide a common solution encompassing the advantages of all of these approaches, we design and propose henceforth the ONTIDS alert correlation framework that address the data integration problems while attaining the flexibility and extensibility objectives mentioned in Sect. 1.

3 The ONTIDS Alert Correlation Framework

The ONTIDS framework was made context-aware in order to take full advantage of the context information typically available to security analysts have typically access to prioritise alerts, and ontology-based in order to provide a technological solution to the problem of heterogeneous data integration and retrieval.

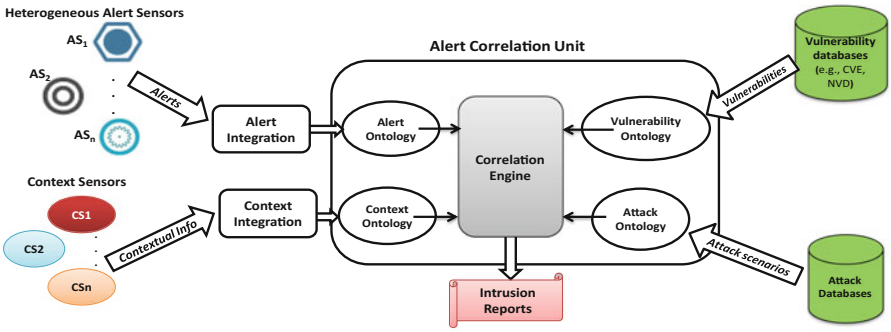

Fig. 1. The ONTIDS ontology-based context-aware alert correlation framework

The ONTIDS framework is depicted in Fig. 1. In its first step, the alerts generated via distributed homogeneous or heterogeneous IDS are collected and transfered into the alert integration component. Also in this step, all the information required for reasoning on these alerts is gathered from three different information resources namely: Context Sensors (CS), common vulnerability databases, and attack databases.

The second step consists of the following two tasks: (*i*) integrating and converting all the alerts generated by the various IDS into a unified format analysable by the alert correlation unit, and (*ii*) integrating all the contextual information received implicitly or explicitly from the various tools implemented in the system.

In the third step, the alert and context ontologies are populated based on the integrated and converted alert and context information. In order to fully automate the alert correlation process, we have designed a group of comprehensive and extensible ontologies, namely alert, context, attack and vulnerability ontologies. The explicit relationships between these ontologies reasoning on the information gathered from various resources, including the (mostly) static attack and vulnerability databases.

The last step consists in correlating the existing information within the ontologies, which is done via the correlation engine using ontology description logic.

3.1 Information Resources

Alert sensors generate alerts based on the suspected malicious behaviours that they observe on the systems they monitor. The most typical and commonly deployed type of sensor are NIDS that generate alerts by examining individual network traffic packets. They can also include host-based IDS that generate alerts based on system or application activity observed on a particular machine. Finally, it can also include other type of non security-related sensors such as application and system logs that are not generating alerts *per se*, but rather system events that the security analyst consider important enough to be correlated

with other sources of alert. The difficulty here is that while many NIDS and HIDS will generate IDMEF-compliant alerts by filling generic attributes (e.g. time, severity, etc.), there might some sensor- or log-specific attributes that we might want to correlate on, and that must therefore be integrated also. This is what ontologies are particularly suited for.

Context sensors is a generic term for any information source that can provide contextual information about the systems that are being monitored. The concept of context is purposefully vague to allow analysts to define and use the particular aspects that they think is suitable for monitoring of their systems. This can include different types of information such as configuration (network, host or application), vulnerabilities, user role and profile, location, and even criticality of the corresponding IT asset. Contextual information can be implicitly collected by methods such as vulnerability scanning, network fingerprinting, passive network monitoring tools, or they can be explicitly provided by system administrators through tools such as Configuration Management Systems (CMS), for example.

Known vulnerabilities. At first, we gather information about vulnerabilities from the well-known public databases such as the Common Vulnerabilities and Exposures (CVE) [14] or the NVD. Then, vulnerabilities can be associated to context instances (e.g. hosts, networks, applications) through vulnerability scanning or asset management. Severity information from these databases, in combination with information on asset criticality, can then be used to help prioritise alerts occurring in these contexts.

Attack scenarios and models. Attack information and models can obtained from standardised databases such as the Common Attack Pattern Enumeration and Classification (CAPEC) [15] or expert knowledge. In order to model attacks, any of the existing attack modelling languages such as LAMBDA [13] or STATL [16] could be used. However, it is outside of the scope of this work to implement these formalisms within the ontology description logics that we use.

3.2 Alert and Context Integration

Different types of IDS sensors produce alerts in various formats that might not be natively interpretable by the correlation unit. Hence, it is necessary to pre-process these alert streams and export them in a format that is understandable by the correlation unit. In following good ontological engineering practises, all alert sensor-specific fields should be translated into class attributed at the highest possible class in the taxonomy of alerts, i.e. that where all subclasses contain that type of information (or an equivalent one). The use of standard representations such as IDMEF [17] or the Common Event Expression (CEE) [18] should be encouraged, but not at the detriment of not integrating sensor-specific information that is not standard-compliant; that is what sensor-specific alert subclasses are for. For simplicity of presentation and for illustrative purposes, we use an IDMEF-specific ontology in the rest of this paper.

The context integration component of our framework also integrates all the contextual information in various formats received implicitly or explicitly from

various tools implemented in the system. In this component, the contextual information gathered using our designed drivers is converted into a unified format analysable by the other components, i.e. into instances in the context ontology. Once the integration process is complete, the correlation process can start.

3.3 Description of the Ontologies

We chose to use ontologies because they provide a powerful knowledge representation information structure in a unified format that is understandable by both machines and humans. The use of ontologies and ontology description logic enables us to fully automate the correlation process that is typically done manually by security analysts, and this uniformly considering all relevant information, no matter what its original format or source.

In order to integrate the data inputs to the correlation process and allow generic correlation reasoning, independent of specificities of information resources, we have constructed basic ontologies capturing the essence of the concepts of alert, context, vulnerability, and attack. Essentially, they correspond to the following intuitive security facts:

1. Attack scenarios will generate system events that might in turn trigger sensors to *cause* related alerts. Depending on the attack model, an attack scenario might be described as linear sequence of events, or a partial ordering of events with pre- and post-conditions, an attack graph, etc.
2. Alerts *happen in* a context, whether this is an IT asset, network location, application, user, etc. In our case this relationship will be made explicit through information provided by the sensor with the alert (e.g. IP address).
3. Vulnerabilities are always associated to a context, whether to high-level context concepts (e.g. an asset or service type) or to lower-level context subclasses (e.g. particular versions of OS or applications). Conversely, explicit context instances can be linked to specific or generic vulnerabilities, through vulnerability assessment or CMS information.
4. (Most) attack scenarios will require certain vulnerabilities to be present on the systems (context) so that they can *exploited* by that attack.

Figure 2 illustrates these class relationships and some of the subclasses of the basic ontology. These "starter" ontologies are not meant to be the end state of knowledge representation that security analysts would need in using our framework, but rather a starting point or template from which to build on, depending on the kind of sensors, context information or granularity of vulnerabilities and attack modelling desired. We now describe each of these ontologies in more detail.

Alert ontology. All the integrated alerts are transferred into this ontology as its instances. Alert ontology has dependency relationship with the context ontology and an association relationship with the attack ontology, since usually each alert a is typically by a (suspected) attack *at* in a particular context c. The generic base class *Alert* in Fig. 2 includes generic alert attributes such as source, target, time, and analyser (i.e. sensor). It is important to note that because the concept

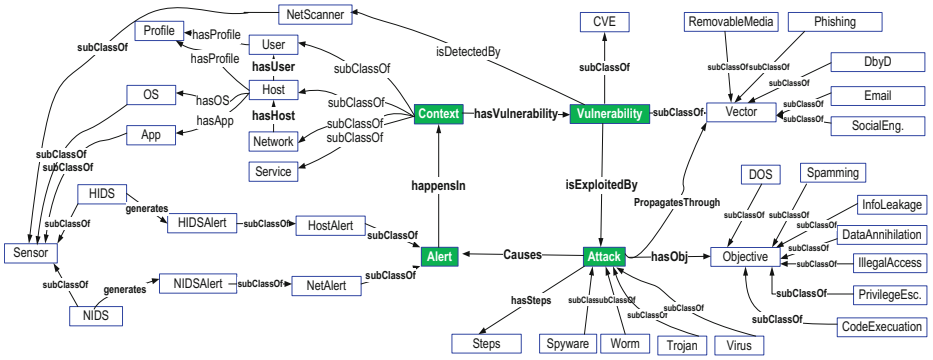

Fig. 2. Class diagram relationship of the designed ontologies

of context is potentially very rich and multifaceted, it is likely that a single alert might have to be linked to multiple context instances from various subclasses (e.g. a user, a network segment, an application), and thus the association between alert and context will be many-to-many at the Context base class level.

Context ontology. The integrated contextual information is transferred into the context ontology. We split contextual information into two categories: (*i*) static context information that rarely changes over time (e.g. network architecture, host/user profiles, and OS type), and (*ii*) dynamic context information that changes continuously over time (e.g. traffic type, system usage, time of day/week). As depicted in Fig. 2, the context ontology includes a Context base class and User, Host, Network and Service subclasses with their corresponding attributes.

Vulnerability ontology. This ontology represents the list of vulnerabilities related to the existing assets in the underlying context. This ontology has a part-whole relationship (composition) with the context ontology, since every vulnerability v is specific to a particular type of system, which is represented as a subclass of Context (typically Host). Thus, v can be associated with all the asset (context) instances $[c_1, \ldots, c_n]$ that are vulnerable to it, by querying the ontology for Host instances whose applications (App) or OS are those associated to that vulnerability. This ontology also has an association relationship with attack ontology, since usually every vulnerability v is exploitable by one or more attacks.

Attack ontology. The attack ontology includes information related to the known attack scenarios, and it includes generic attack attributes such as vectors, objectives, and so on. The Vector class represents the method that is used by an attack to infect computer systems, with subclasses in Fig. 2 including social engineering, phishing and removable media are examples of such methods. The Objective class includes subclasses such as information leakage, remote core execution, spamming and privilege escalation. The attack ontology has dependency

relationships with the context ontology, and association relationship with the alert and vulnerability ontologies, since basically every attack *at* needs a particular context c to proceed, it might need to exploit particular vulnerabilities $[v_1, \ldots, v_n]$, and it results in triggers some alerts $[a_1, \ldots, a_n]$.

3.4 Correlation Engine

In order to implement the correlation logic, we employ Ontology Web Language-Description Logic (OWL-DL) to design and populate an ontology for each of the above four inputs. The use of generic language like OWL-DL provides significant flexibility to the framework by allowing the reuse or adaptation of data queries expressed in that logic to various deployment and security monitoring scenarios, e.g. on-line detection or after-the-fact network forensics analysis.

The correlation process is two-fold and can be viewed as two independent traversals on the core ontology classes:

1. *Context- and vulnerability-based filtering.* Given an alert (or alerts) determine which contexts instances are involved, what are their associated known vulnerabilities, and finally determine which attack scenarios could be exploiting them.
2. *Attack reconstruction.* For each possible attack scenario related to this (or these) alert(s), try to match the sequence of previous alerts with the steps of the attack.

The outcome of this process should hopefully provide the security analyst with a reduced list of high level descriptions of potential ongoing (or completed) attacks that includes few redundancies, non relevant scenarios and false positives.

In order to implement both components of this alert correlation approach we use a set of logic rules expressed in Semantic Web Rule Language (SWRL) and Semantic Query-Enhanced Web Rule Language (SQWRL). While various specific correlation approaches could be implemented within the above generic model, we use certain aspects of the approach described in [12] to illustrate the use of our framework.

3.5 Example Implementation of Valeur *et al.*'s Approach

The alert correlation approach proposed in [12], includes a comprehensive set of steps that covers various aspects of the alert correlation process. In order to show how ONTIDS automatically implement these steps, in the following, we explain the implementation details of some of these steps employing ONTIDS.

Alert fusion. Alert fusion is the process of merging alerts that represent the independent detection of the same malicious event by different IDS. An important condition in order to fuse two or more alerts is that they should be in the same time window. We have defined Rule 1 within the correlation engine in order to perform alert fusion:

Rule 1

$$ALERT(?a1) \wedge ALERT(?a2) \wedge ANALYSER(?an1) \wedge ANALYSER(?an2) \wedge DetectTime(?dt1) \wedge DetectTime(?dt2) \wedge SOURCE(?s1) \wedge SOURCE(?s2) \wedge TARGET(?tar1) \wedge TARGET(?tar2) \wedge CLASSIFIC(?cl1) \wedge CLASSIFIC(?cl2) \wedge ASSESSMENT(?as1) \wedge ASSESSMENT(?as2) \wedge stringEqual(?s1,?s2) \wedge stringEqual(?tar1,?tar2) \wedge stringEqual(?cl1,?cl2) \wedge stringEqual(?as1,?as2) \wedge subtractTimes(?td,?dt1,?dt2) \wedge lessThan(?td,"5s") \longrightarrow sqwrl:select(?a1)$$

Alert Verification. Alert verification is the process of recognising and reducing non-relevant alerts which refer to the failed attacks. The major reason of attack failure is the unavailability of the contextual requirements of the attack, i.e. the absence of required vulnerabilities in the attack context. Identifying failed attacks allows the correlation engine to reduce the effects of non-relevant alerts in its decision process. Rules 2 and 3 within the correlation engine of our framework perform alert verification based on the targeted system vulnerabilities:

Rule 2

$$ALERT(?a) \wedge HOST(?h) \wedge OS(?o) \wedge VULNERABILITY(?v) \wedge CLASSIFICATION(?cl) \wedge REFERENCE(?ref) \wedge hasTarget(?a,?h) \wedge hasClassific(?a,?cl) \wedge hasOS(?h,?o) \wedge hasReference(?c,?ref) \wedge hasVulnerability(?o,?v) \wedge hasName(?ref,?n1) \wedge hasName(?v,?n2) \wedge stringEqual(?n1,?n2) \longrightarrow sqwrl:select(?a)$$

Rule 3

$$ALERT(?a) \wedge HOST(?h) \wedge APP(?ap) \wedge VULNERABILITY(?v) \wedge CLASSIFICATION(?cl) \wedge REFERENCES(?ref) \wedge hasTarget(?a,?h) \wedge hasClassific(?a,?cl) \wedge hasApp(?h,?ap) \wedge hasReference(?c,?ref) \wedge hasVulnerability(?ap,?v) \wedge hasName(?ref,?n1) \wedge hasName(?v,?n2) \wedge stringEqual(?n1,?n2) \longrightarrow sqwrl:select(?a)$$

Attack thread reconstruction. Thread reconstruction is the process of merging a series of alerts that refer to an attack launched by one attacker against a single target, and is another step in the alert correlation process of [12]. Similarly to the alert fusion process, the alerts should happen in the same time window to be correlated. Rule 4 performs the thread reconstruction process:

Rule 4

$$ALERT(?a) \wedge HOST(?h1) \wedge HOST(?h2) \wedge TIME(?t1) \wedge TIME(?t2) \wedge hasSource(?a,?h1) \wedge hasTarget(?a,?h2) \wedge hasDetectTime(?a,?dt) \wedge greaterThanOrEqual(?dt,?t1) \wedge lessThanOrEqual(?dt,?t2) \longrightarrow sqwrl:select(?a,?h1,?h2)$$

In summary, we can see how the first component of our canonical description is implemented by the correlation engine by applying first Rules 1, 2 and 3 to reduce non-relevant alerts. For this purpose, it retrieves required information from the alert, context, and vulnerability ontologies. Next, and for those alerts and scenarios that are relevant attack thread reconstruction is performed by applying Rule 4, where the engine attempts to make a mapping between the filtered alerts and the steps of attacks in the attack ontology. Once it finds any mapping between the two ontologies, it will output the whole attack scenario.

4 Implementation and Evaluation

In order to illustrate and validate our approach, we constructed an example reference implementation using various ontology representation and reasoning tools, and used it to conduct some simple security analysis on the well-known UNB ISCX and DARPA 2000 datasets. the capabilities and flexibility of framework, by describing how it is used in two distinct case studies.

4.1 Reference Implementation

To illustrate the integration of distinct IDS, we have selected the Snort [19] and IBM RealSecure [20] NIDS as our alert sensors. As an alert integration tool, we use Prelude [21], which is an agent-less, universal, Security Information Management System (SIM, a.k.a SIEM).

We use the Protégé ontology editor and knowledge acquisition system to design and implement the ontologies using the Ontology Web Language Description Logic (OWL-DL). We instantiate the above-mentioned ontologies from information coming from the alert integration component, the contextual information gathering sensors such as Nessus [22] and Nmap [23], the CVE vulnerability database, and the designed attack scenarios. and relational databases. We use the DataMaster plug-in [24] in order to transfer data from relational databases and the XML Tab plug-in to transfer data from a XML files.

Finally, we utilise the Pellet plug-in [25] as a reasoner for OWL-DL, the Jess rule engine [26] as SWRL rule compiler, and SQWRL [27] in order to query the ontologies for various purposes.

4.2 Case Study 1: Island-Hopping Attacks

As our first case study, we describe an instance of island-hopping attack scenario described in [12] which is part of the UNB ISCX Intrusion Detection Evaluation Dataset [28]. As shown Fig. 3, in this scenario the attacker employs the Adobe Reader `util.printf()` buffer overflow vulnerability (CVE-2008-2992) to execute arbitrary code with the same privileges as the user running it.

To launch the attack, the attacker creates a malicious PDF file using Metasploit (for example), and embeds a Meterpreter reverse TCP shell on port 5555 inside it. Then, the attacker sends a system upgrade email including the PDF file on behalf of admin@[...] to all the users of the testbed. Through user5, who initiates the first session (alert 1), the attacker starts to scan potential hosts on two subnets 192.168.1.0/24 and 192.168.2.0/24 (alert 2). User12 is identified as running Windows XP SP1 with a vulnerable SMB authentication protocol on port 445 (CVE-2008-4037) (alerts 3 and 4). The attacker exploits this vulnerability to capture user12 (alert 5), and a scan is performed from this user to the server subnet (192.168.5.0/24) (alert 6). This scan identifies a Windows Server 2003 running an internal Web application using MS SQL Server as its backend database with only port 80 opened. This leads to the use of Web application hacking techniques such as SQL injection. Finally, the attacker compromises the

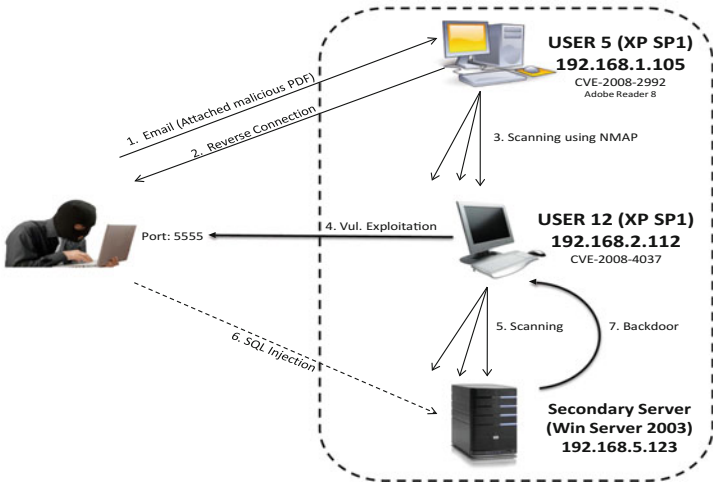

Fig. 3. An instance of island-hopping attack

Table 1. Alerts generated by alert sensors in the island-hopping attack scenario

Alert ID	Name	Sensor	Date	Source	Target	Tag
1	Local exploit	HIDS	6/13/10 16:02:20	192.168.1.105	192.168.1.105	Step 1
2	Scanning	NIDS	6/13/10 16:42:24	192.168.1.105	192.168.1.0/24 192.168.2.0/24	Step 2
3	Windows file sharing	NIDS	6/13/10 17:20:32	192.168.1.105	192.168.2.112	Step 3
4	Windows file sharing	NIDS	6/13/10 17:34:32	192.168.1.105	192.168.2.112	Step 3
5	Local exploit	HIDS	6/13/10 17:50:24	192.168.2.112	192.168.2.112	Step 3
6	Scanning	NIDS	6/13/10 18:02:37	192.168.2.112	192.168.5.0/24	Step 4
7	HTTPWeb	NIDS	6/13/10 18:19:41	192.168.2.112	192.168.5.123	Step 4
8	SQLInjection	AIDS	6/13/10 18:20:19	192.168.5.123	192.168.5.123	Step 5

target system (alerts 7 and 8). Table 1 presents a summary of the alerts, and indicates their corresponding steps.

In order to correlate the alerts generated by alert sensors during the above scenario, first, the alert integration component integrates all received alerts. Then, the integrated alerts are transferred into the alert ontology. Additionally, we manually populate vulnerability and context ontologies based on the published documents related to the UNB ISCX dataset. Therefore, the Adobe Reader `util.printf()` vulnerability and others that might be present in the IT infrastructure are input into the vulnerability ontology. Contextual information about the existing hosts (IP addresses, open ports, available services, etc.), services and users are also manually input into the context ontology. In this case,

Fig. 4. The island-hopping attack graph detected by the proposed framework

this includes the information about the three compromised hosts (IP addresses 192.168.1.105, 192.168.2.112, and 192.168.5.123), their open ports (i.e. 5555 and 445).

Next, the correlation engine correlates the existing information within the ontologies. For this purpose, we first use the Rules 1, 2 and 3 to eliminate non-relevant alerts. Then, using the following rule we reconstruct the attack scenario.

Rule 5

$$ATTACK(?at) \wedge hasName(?at,"InsiderAttack1") \wedge ALERT(?a1) \wedge CLASSIFICATION(?cl1) \wedge HOST(?h1) \wedge$$
$$REFERENCE(?ref1) \wedge hasTarget(?a1,?h1) \wedge hasClassific(?a1,?cl1) \wedge hasReference(?cl1,"CVE-2008-2992")$$
$$\wedge ALERT(?a2) \wedge hasSource(?a2,?h1) \wedge hasName(?a2,"Scanning") \wedge ALERT(?a3) \wedge HOST(?h2) \wedge$$
$$CLASSIFICATION(?cl2) \wedge REFERENCE(?ref2) \wedge hasTarget(?a2,?h2) \wedge hasClassific(?a2,?cl2) \wedge$$
$$hasReference(?cl2,"CVE-2008-4037") \wedge ALERT(?a4) \wedge hasSource(?a4,?h2) \wedge hasName(?a4,"Scanning") \wedge$$
$$ALERT(?a5) \wedge hasSource(?a5,?h2) \wedge hasName(?a5,"SQLInjection")$$
$$\longrightarrow sqwrl:select(?a1,?a2,?a3,?a4,?a5,?at)$$

Rule 5 correlates alert and attack ontologies, and attempts to discover corresponding alerts for each step of the attack. If it finds at least one match regarding each step, the rule will be successful in detecting the whole attack scenario. Figure 4 represents the result of applying Rule 5 to the ontologies, showing that ONTIDS should be able to reconstruct the attack.

4.3 Case Study 2: Recon-Breakin-Escalate Attacks

As the second case study, we evaluate the proposed alert correlation framework using the DARPA 2000 dataset LLDDOS 1.0 scenario [29]. LLDDOS 1.0 is a multi-step scenario corresponding to a Distributed Denial of Service (DDoS) flooding attack. The attack has 5 phases and it takes about three hours to be completed. We again use both the RealSecure and Snort NIDS as base our alerts sensors to detect all the steps of the attack. Snort outputs around 1,211 raw alerts for the LLDDOS 1.0 dataset, but it does not detect the installation phase of the DDoS attack (i.e. phase 4). On the other hand, and as is described in [30], RealSecure outputs 924 raw alerts for the same dataset, corresponding to the 22 alert types shown in Table 2. However, it does not output any alerts related to ICMP pings (i.e. phase 1). Consequently, the combination of Snort and RealSecure can detect all phases of the attack. Nonetheless, just using a combination of both IDS alerts with a simple `OR` rule will result in a significant number of redundant alerts and false positives, as we will see. With ONTIDS, we expect to have lower redundancy, and fewer non-relevant alerts and false positives.

Table 2. Alert types generated by ISS RealSecure based on the DARPA 2000 dataset

ID	AlertType	ID	AlertType	ID	AlertType
1	*RIPExpire*	9	*Admind*	17	*SSH_Detected*
2	*RIPAdd*	10	*Sadmind_Ping*	18	*Email_Debug*
3	*Email_Ehlo*	11	*Email_Almail_Overflow*	19	*TelnetXdisplay*
4	*TelnetTerminaltype*	12	*HTTP_Java*	20	*TelnetEnvAll*
5	*FTP_User*	13	*Sadmind_Amslverify_Overflow*	21	*Port_Scan*
6	*FTP_Pass*	14	*Mstream_Zombie*	22	*Stream_DoS*
7	*FTP_Syst*	15	*Rsh*		
8	*HTTP_Shells*	16	*HTTP_Cisco_Catalyst_Exec*		

In the second step, Prelude converts all the received alerts into the IDMEF format, and transfers the integrated alerts into the alert ontology as its instances. We manually populate the context and vulnerability ontologies based on the information existing in the published documents related to the DARPA 2000 dataset. Thus, the Solaris sadmind vulnerability (CVE-1999-0977) and others existing vulnerabilities in the underlying network are transferred into the vulnerability ontology. The same is done with contextual information about the existing hosts and users, in this case including the three compromised hosts (IP addresses 172.16.115.20, 172.16.112.50, 172.16.112.10), and their open ports (i.e. telnet port 23) and users (e.g. hacker2).

As before, the correlation engine uses Rules 1–3, to eliminate redundant and non-relevant alerts. Based on our analysis, 32.7 % of all alerts were generated by both Snort and IIS RealSecure. In our case, we report these alerts as a single alert (in order to reduce redundancy) since both IDS agree. Alerts reported by only one IDS are then further analysed by attempting attack reconstruction on the 5 phases of the LLDDOS 1.0 attack scenario, by using the following rule:

Rule 6

$$ATTACK(?at) \wedge hasName(?at,"LLDDOS1") \wedge ALERT(?a1) \wedge ALERT(?a2) \wedge ALERT(?a3) \wedge ALERT(?a4) \wedge ALERT(?a5) \wedge ALERT(?a6) \wedge ALERT(?a7) \wedge HOST(?h1) \wedge hasName(?a1,"Scanning") \wedge hasTarget(?a1,?h1) \wedge hasService(?h1,"ICMP") \wedge hasName(?a2,"Sadmind_Ping") \wedge hasName(?a3,"Sadmind_Amslverify_Overflow") \wedge hasName(?a4,"Admind") \wedge hasName(?a4,"Rsh") \wedge hasName(?a4,"MStream_Zombie") \wedge hasName(?a4,"Stream_DOS") \longrightarrow sqwrl:select(?a1,?a2,?a3,?a4,?a5,?a6,?a7,?at)$$

Rule 6 correlates alert and attack ontologies, and discovers corresponding alerts for the each step of the attack. If at least one match is found for each step, the rule will be successful in detecting the whole attack scenario. According to this rule, our results indicate that 91.08 % of the alerts were false positives, and only 8.92 % of the alerts were true positives.

Table 3 summarises our results. Since both Snort and ISS RealSecure only detect a few of the 33,787 attack events in Phase 5 (launching DDoS), their total false negative rates are quite high. The recall column consequently reports low values for both sensors and ONTIDS. On the other hand, ONTIDS does considerably well at reducing false positives, in fact reducing it to 0.

Table 3. Experimental results based on the DARPA 2000 dataset

IDS	Redundant alerts (%)	FP	TP	FN	Precision	Recall	F-measure
Snort	0.00	1118	93	33814	0.07	2×10^{-3}	2×10^{-3}
RealSecure	0.00	870	54	33853	0.05	10^{-3}	10^{-3}
ONTIDS	32.7	0	123	33784	1.00	3×10^{-3}	5×10^{-3}

5 Conclusions

In this paper, we introduced ONTIDS an ontology-based automated alert correlation framework to try to benefit from the combined advantages of previous alert correlation approaches (including context awareness), while providing a level of flexibility that would allow it to be used in the many different deployment scenarios that security analysts are likely to face.

The main idea behind ONTIDS is to use and leverage a template ontology containing base classes and some subclasses for the concepts of IT asset context, alert, vulnerability and attack. The correlation engine is then implemented using logic rules written in Semantic Web Rule Language (SWRL) and Semantic Query-Enhanced Web Rule Language (SQWRL) based on the OWL description logic (OWL-DL). The ontologies and correlation rules described here are generic enough to (*i*) implement as special cases other existing correlation approaches such as that of *Valeur et al.*, and, (*ii*) be applied with minimal changes to different analysis scenarios, such as in the two case studies demonstrated.

We have demonstrated the use of the ONTIDS alert correlation framework in two quite different case studies involving considerably distinct attack scenarios. More important than the reduction in false positives (in this somewhat contrived evaluation scenario), the point of this exercise was to show the level of flexibility of such an approach. The fact that the same correlation Rules 1–3 are used for the context-based alert filtering in both scenarios deceptively hides the fact that the vulnerability and context instances in both cases are quite different as they come from different sources, and hence have different attributes and properties. As security analysts start to use ONTIDS, we expect that these ontologies will naturally expand to include new subclasses capturing the idiosyncrasies of the systems being monitored, the various types of sensors monitoring them, and richer and more complex attack models and vulnerabilities.

In conclusion, we hope to continue to evaluate its viability and usefulness by conducting field studies with data collected from real-world systems and analysed by real security analysts. On the one hand, this will force us to test the flexibility of the framework by incorporating richer context and sensor ontologies (possibly stretchning the limits of abstraction), while also having to express richer correlation algorithms, possibly based on more sophisticated attack models and description languates such as LAMBDA or STATL.

Acknowledgements. This research was sponsored in part by the Inter-networked Systems Security Strategic Research Network (ISSNet), funded by Canada's Natural Sciences and Engineering Research Council (NSERC).

References

1. Li-Zhong, G., Hui-bo, J.: A novel intrusion detection scheme for network-attached storage based on multi-source information fusion. In: 2012 Eighth International Conference on Computational Intelligence and Security, pp. 469–473 (2009)
2. Thomas, C., Balakrishnan, N.: Improvement in intrusion detection with advances in sensor fusion. Trans. Inf. For. Sec. **4**(3), 542–551 (2009)
3. Dreger, H., Kreibich, C., Paxson, V., Sommer, R.: Enhancing the accuracy of network-based intrusion detection with host-based context. In: Julisch, K., Kruegel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 206–221. Springer, Heidelberg (2005)
4. Cuppens, F., Miege, A.: Alert correlation in a cooperative intrusion detection-framework. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 202–215 (2002)
5. Morin, B., Debar, H.: Correlation of intrusion symptoms: an application of chronicles. In: Vigna, G., Kruegel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 84–112. Springer, Heidelberg (2003)
6. Chen, L., Aritsugi, M.: An SVM-based masquerade detection method with online update using co-occurrence matrix. In: Büschkes, R., Laskov, P. (eds.) DIMVA 2006. LNCS, vol. 4064, pp. 37–53. Springer, Heidelberg (2006)
7. Raftopoulos, E., Egli, M., Dimitropoulos, X.: Shedding light on log correlation in network forensics analysis. In: Flegel, U., Markatos, E., Robertson, W. (eds.) DIMVA 2013. LNCS, vol. 7591, pp. 232–241. Springer, Heidelberg (2013)
8. Gagnon, F., Massicotte, F., Esfandiari, B.: Using contextual information for ids alarm classification (extended abstract). In: Flegel, U., Bruschi, D. (eds.) DIMVA 2009. LNCS, vol. 5587, pp. 147–156. Springer, Heidelberg (2009)
9. Sinha, S., Jahanian, F., Patel, J.M.: WIND: workload-aware intrusion detection. In: Kruegel, C., Zamboni, D. (eds.) RAID 2006. LNCS, vol. 4219, pp. 290–310. Springer, Heidelberg (2006)
10. Vorobiev, A., Bekmamedova, N.: An ontology-driven approach applied to information security. J. Res. Prac. Inf. Technol. **42**(1), 61 (2010)
11. Coppolino, L., D'Antonio, S., Elia, I., Romano, L.: From intrusion detection to intrusion detection and diagnosis: An ontology-based approach. In: Lee, S., Narasimhan, P. (eds.) SEUS 2009. LNCS, vol. 5860, pp. 192–202. Springer, Heidelberg (2009)
12. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.: Comprehensive approach to intrusion detection alert correlation. IEEE Trans. Depend. Secur. Comput. **1**(3), 146–169 (2004)
13. Cuppens, F., Ortalo, R.: LAMBDA: A language to model a database for detection of attacks. In: Debar, H., Mé, L., Wu, S.F. (eds.) RAID 2000. LNCS, vol. 1907, pp. 197–216. Springer, Heidelberg (2000)
14. CVE: Common vulnerabilities exposures (CVE), the key to information sharing. http://cve.mitre.org/
15. CAPEC: Common attack pattern enumeration and classification (capec). http://capec.mitre.org/
16. Eckmann, S.T., Vigna, G., Kemmerer, R.A.: STATL: An attack language for state-based intrusion detection. J. Comput. Secur. **10**(1), 71–103 (2002)

17. Debar, H., Curry, D., Feinstein, B.: The intrusion detection message exchange format (idmef) (2007)
18. Mitre Corporation: A standardized common event expression (CEE) for event interoperability (2013)
19. Roesch, M.: Snort - lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX Conference on System Administration (LISA '99), pp. 229–238. USENIX Association, Berkeley (1999)
20. Corporation, I.: IBM RealSecure. http://www-935.ibm.com/services/us/en/it-services/express-managed-protection-services-for-server.html
21. Zaraska, K.: Prelude ids: current state and development perspectives (2003). http://www.prelude-ids.org/download/misc/pingwinaria/2003/paper.pdf
22. Deraison, R.: The nessus project (2002). http://www.nessus.org
23. Lyon, G.F.: Nmap network scanning: The official Nmap project guide to network discovery and security scanning. Insecure, USA (2009)
24. Nyulas, C., O'Connor, M., Tu, S.: Datamaster–a plug-in for importing schemas and data from relational databases into protege. In: Proceedings of the 10th International Protege Conference (2007)
25. Parsia, B., Sirin, E.: Pellet: An OWL-DL reasoner. In: Third International Semantic Web Conference-Poster, p. 18 (2004)
26. Friedman-Hill, E. et al.: Jess, the rule engine for the java platform (2003)
27. O'Connor, M., Das, A.: SQWRL: a query language for OWL. In: Proceedings of the 6th Workshop on OWL: Experiences and Directions (OWLED2009) (2009)
28. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput. Secur. **31**(3), 357–374 (2012)
29. MIT Lincoln Laboratory: 2000 DARPA intrusion detection scenario specific data sets (2000)
30. Hu, Y.: TIAA: A toolkit for intrusion alert analysis (2004)