

Intrusion Detection and Tolerance in Grid-based Applications

Jun WANG, Luigi LO IACONO

C&C Research Laboratories

NEC Europe Ltd.

Sankt Augustin, Germany

{wang, lo_iacono}@ccrl-nece.de

Abstract—With the increasing use of Grid-based applications, especially in business-driven scenarios, new types of cross-domain attacks which initiate from one site and then easily spread to other federated sites are expected to appear and become serious threats. In this paper, the need for dedicated Grid Intrusion Detection Systems (Grid-IDS) is motivated by giving such an example attack on a federated service protocol. A generic Grid-IDS architecture is presented as well as a concrete realization based on various Web services specifications. At the heart of the introduced Grid-IDS architecture is the correlation service, which receives the event information from sensors distributed across the federation and detects intrusions by analyzing and correlating the events. A protocol-aware correlation service is proposed, in which each service protocol is abstracted to a generic representation – a relationship of roles. Finally, based on the developed Grid-IDS and the gathered experiences, an approach towards intrusion tolerance is presented and discussed.

Keywords: *Grid, SOA, Intrusion Detection, Intrusion Tolerance*

I. INTRODUCTION AND MOTIVATION

With service-based software engineering technologies [1, 2, 3] arising and becoming mature in both scientific and commercial areas, many institutions are applying service-oriented technologies [4, 5, 6] to form powerful collaboration platforms. Typical usage scenarios are Enterprise Application Integration (EAI) where internal and in the majority of cases heterogeneous resources from distinct divisions are linked through the use of loosely coupled services or inter-enterprise information systems. Underlying these applications, the related standards and specifications on federated services [7, 8, 9] are evolving and adopted quickly. However, specific security issues have to be considered in federated service environments. New types of cross-domain attacks which initiate from one site and then easily spread to other federated sites are expected to appear and become serious threats. In this paper, we will use a running attack example based on the Liberty Alliance ID-FF1.2 protocol [7] to demonstrate these kinds of cross-domain attacks and to motivate as well as illustrate our work.

Let us assume an e-Business scenario where the federation includes a banking service and a shopping service, which are federated through an Identity Provider (IdP). The mechanism deployed for federation is the Liberty Alliance ID-FF1.2

protocol. A normal and legitimated workflow in a simplified description is as follows:

- Bob requests a single sign-on token (a SAML token in ID-FF1.2) by sending his username and password to the IdP.
- Bob obtains the token and requests access to the Banking Service presenting his sign-on token.
- The Banking Service will authenticate the token of Bob by verifying the existing federation relationship with IdP. After the authentication, its authorization engine will accept or refuse this request according to the local authorization policy.

This common workflow almost does not change if an attacker masquerading Bob tries to misuse the services for his profit (see Fig. 1).

- Malory contacts the IdP's Web site and tries to impersonate Bob by guessing his username and password. Malory's initial attempt fails. Her second guess succeeds and she obtains a token from the IdP (step 1, 2, 3, and 4 in Fig. 1).
- With the obtained token, Malory accesses one of the federated e-Business services, the banking service, masquerading as Bob and transfers all his money to a bank account on the Bahamas (step 5 in Fig. 1).

This is a typical cross-domain attack in which the attacker compromises one site and can then spread his attack easily to the other federated sites. Due to the occurrence of these events at distinct administrative domains without a mechanism to analyze them in an aggregated form, the attack remains undetected by each of the single sites. Only the analysis of the aggregated events provides a mean to detect a possible intrusion, since then it is possible to link the password failure attempts with the abnormal money transfer, resulting in a correlation event.

The rest of this paper is organized as follows. Section II presents a service-oriented Grid intrusion detection architecture and its components. Section III describes our idea of correlating events according to protocols first and then presents a corresponding correlation mechanism. Section IV gives a typical interaction procedure of different components in Grid-

IDS. Section V discusses the key challenges in intrusion tolerance as a further step. Section VI concludes.

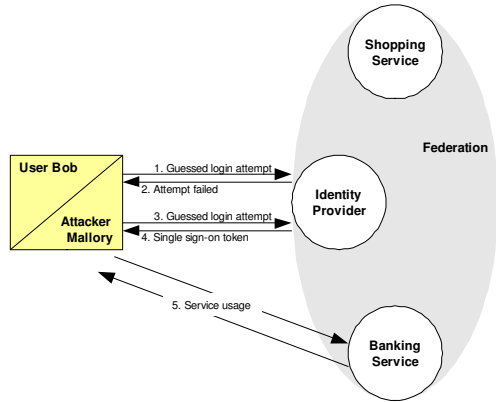


Figure 1. Cross-domain attack on federated services

II. GRID INTRUSION DETECTION SYSTEM

The Grid Intrusion Detection System (Grid-IDS) presented in this paper addresses the security requirements specific to large distributed collaborative service environments. Furthermore, it is composed of a monitoring, assessment, and management module, as found in most of the traditional IDS [10, 11], but has been considered from the viewpoint of services. None of the existing distributed IDS architectures including hierarchical IDS [12, 13, 14], peer-to-peer IDS [15, 16, 17] and mobile agent IDS [18, 19, 20] satisfy the flexible composable architectural requirements of an IDS for Grid-based applications and/or systems. An actual application could e.g. consist of a series of loosely coupled services, which is a mixture of hierarchical and peer-to-peer structures. Therefore, we present in this paper an IDS for service-oriented systems where the IDS is a service-based system itself designed set of loosely coupled components.

A. Conceptual Modules

Before introducing the design of the underlying service-oriented architecture of the developed Grid-IDS, the main conceptual modules found in any kind of IDS are discussed in the light of distributed service ecosystems.

The monitoring module keeps track of activities in the Grid and collects event data via sensors placed at different points in the Grid. The collection of alert and event data includes remote Grid sites where possible and required and is usually subject to specific policies, regulations and contracts. The monitored activities and displacement topology of sensors strongly depend on the type of attacks that the Grid-IDS should detect. An Grid-IDS should support the combination of different sensor types into a truly hybrid monitoring environment including new sensor types that, for example, collect event data from Grid-service endpoints. The monitoring module feeds the collected data to the assessment and management modules.

The assessment module analyzes, assesses and correlates event data, security alerts and incident reports in order to detect attacks. The module aims at detecting different types of attacks or specializes on a certain kind of attack. Upon detection, alerts

that contain relevant information about the detected suspicious activity are generated. Alerts can be analyzed further, by local or remote analyzers or be fed to the management module. In addition to local analysis of event data that originates at a Grid site, an Grid-IDS offers remote analysis services. These remote services make use of information from different Grid sites and sources to detect more complex attack patterns and correlate alerts.

The management module is in charge of the administration of the IDS components under its control (typically a set of sensors and analyzers). The management module manages the security of the IDS components, their availability and performance. The module is also responsible for handling alert warnings and for initiating appropriate reactions. The management may be distributed among several management points or be centralized. The most common scenario will be that of a management point at each Grid site that is in charge of the Grid-IDS components that are local to its site and the handling of warnings issued to the site by remote assessment services. Moreover, the configuration of security parameters and the management of the security status is another important capability a management component needs to provide. Finally, the functionalities found in classical security management systems should be available too, such as the heartbeat of each component, the basic metrics like the number of generated events for each component in a period, etc.

B. Service-oriented Architecture

The modules already reveal specific requirements when discussed in the target application environment of large-scale distributed service ecosystems. Besides conceptual specifics, architectural properties have to be considered, resulting in the demand for service-orientation. The designed SOA of the Grid-IDS is shown in Fig. 2.

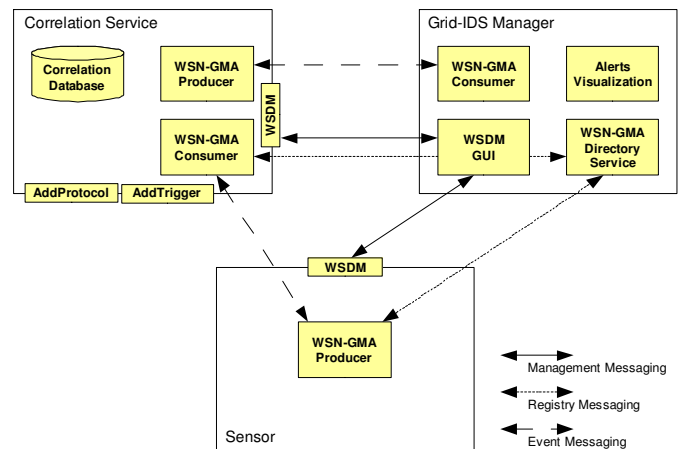


Figure 2. Service-based Grid intrusion detection architecture

The monitoring module has been designed according to the Grid Monitoring Architecture (GMA) [21] and a concrete instantiation has been developed based on WS-Notification (WSN) [22] as will be discussed in section II.B.1. This basically forms a service-oriented notification middleware from which components can be found in all the high-level Grid-IDS building blocks. The two main categories of assessment

technologies are modeled by the sensor (local assessment; see section II.B.3) and the correlation service (remote assessment; see section III). Finally, the management module is represented by the Grid-IDS manager which allows the distributed service-oriented management of the system based on the Web Service Distributed Management (WSDM) standard [23] (see section II.B.2).

1) Monitoring Components

The Grid Monitoring Architecture (GMA) [21] is a framework draft from OGF that is specifically designed to fit the requirements of monitoring in large scale distributed environments. It gives an abstract account of the core components that constitute a Grid monitoring system and their interaction protocols. There are three main roles for GMA entities: the producer, the consumer and the directory service (see Fig. 3). Producers send event data to consumers directly using either the Push (Subscribe/Notify) or the Pull (Query/Response) model. A directory service fulfils the registry and management of producers and consumers by adding, removing or updating an entry. It fulfils the discovery of matching producers or consumers by the *search* method. By a directory service, producers/consumers can look up and find the corresponding receivers/senders respectively according to, for example, alert types in Grid IDS.

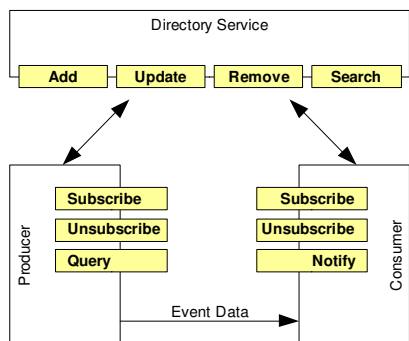


Figure 3. GMA roles, functionalities and interactions

In terms of Web services, the standards of WS- Notification (WSN) [22] define an event driven framework along with some related Web services portals that can be used to monitor events. We have designed a WSN-based GMA system called WSN-GMA as shown in Fig. 4. WSN-GMA is the middleware of our Grid-IDS to compose different sensors and analyzers flexibly according to the actual Grid-IDS deployment topologies. The Web services defined in WS-Notification including NotificationBroker(NB), NotificationConsumer (NC), NotificationProducer (NP), SubscriptionManager (SubM) and PublisherRegistrationManager (PRegM) have been mapped to implement the corresponding functionalities of GMA as follows. The GMA consumer is implemented by NC and Subscriber (Sub). The GMA Producer is implemented by NP, SubM, and Publisher (Pub). SubM and PubM handle publication-related interactions with NB that implements the GMA directory Service. SubM and PRegM are used by GMA producers and consumers for publication update and termination. They host the publication entries as resources. In order to support *Search*, NB can match the publication information either through SubM (for GMA producers' search) or through PRegM (for GMA consumers' search). The

Subscribe/Unsubscribe operations in the GMA consumer side provide a way for the GMA producer to notify the consumer that the producer expects the subscription from this consumer (for Subscribe) or expect to remove an existing subscription of this consumer (for Unsubscribe). We reuse the Notify interface of NC to support these two interfaces.

We have implemented a prototype of WSN-GMA by extending Apache Subscribe 1.1 [24]. Furthermore, in order to support secure communication in WSN-GMA, we have designed and implemented a security framework that provides end-to-end communication security based on WS-SecureConversation together with a specific trust model as well as enhancements to SAML tokens [25]. Note that although PullPoint (PP) in WS-Notification provides a Query/Response service for GMA consumers to retrieve messages from GMA producers by the Pull model, since the Pull model is not required by our Grid IDS, our prototype does not support this model yet.

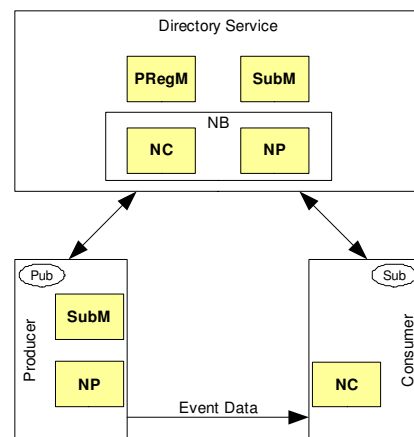


Figure 4. WSN-GMA notification middleware

2) Management Components

According to the description of the conceptual management module in section II.A, we can categorize the main management functionalities into either the management of Grid-IDS components or the management of events. The first category is required for any distributed system. The second category is specific for Grid-IDS.

Since the Grid-IDS we introduce in this paper has a service-oriented architecture, we can use the management related standards from OASIS again, namely WSDM [23] to support the first category of functionalities. As shown in Fig. 2, each sensor or analyzer will deploy a WSDM (MUWS part) service in its site. This service will send management events such as the heart beat, operational status as well as some metrics like the number of generated alerts to a Grid site manager periodically. The manager component has a WSDM GUI to receive and visualize these management events.

For the second category, most of the functionality that can be found in other IDS can be used for Grid-IDS as well, for example, the visualization of received correlation events, the event details trace, and the possible reaction options, etc.

In addition, a WSN-GMA consumer is required as well in order to receive events from analyzers.

Accordingly, a Grid-IDS management model is shown in Fig. 5. All resources in a host are managed through a MUWS conformant Web service. The three manageability capabilities HeartBeat, Metrics and OperationalStatus are exposed as resources with resource properties. For the HeartBeat and Metrics capabilities, a component called InfoBroker will collect the status or performance information from the resources and publish them as management events periodically. For the OperationalStatus capability, when the state from a resource is changed, for example, from Available to Unavailable, this resource will publish a management event by changing the resource property of the OperationStatus capability. Once the WSDM client subscribes to the specific management event topics, it will receive these events automatically by notification. The Ads component in Fig. 5 contains the notification ports to perform the run-time host and resource registration functionality to the management framework for both resources and hosts, so that the framework does not need to be stopped and restarted in response to the inclusion of new hosts and resources. Analogous, whenever a new site joins in a management domain monitored by a Grid-IDS manager, this site will advertise its existence by notifying the manager using the notification Web service port of WSN-GMA. In the same way, on the service (a type of resource) level, any introduced service will notify the site in charge of it, which will register it for information collection in the future.

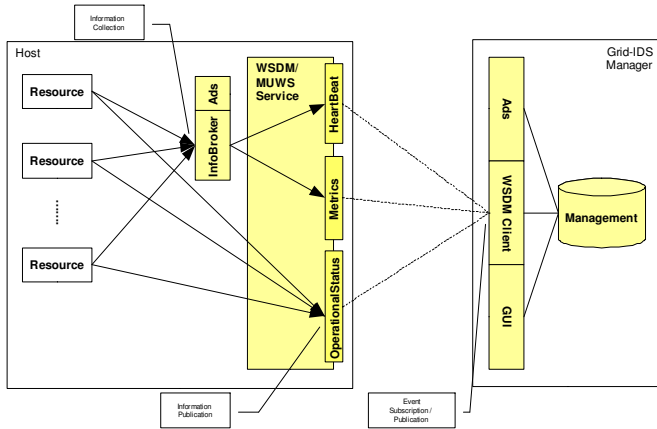


Figure 5. WSDM based resources information management model

We have implemented a prototype of the management model shown in Fig. 5 based on the Apache WSDM MUWS 1.0 open source implementation called Apache MUSE 1.0 [26].

3) Sensors

The basic alerts are produced by sensors embedded in different layers at different Grid sites. A sensor can be a traditional host IDS or a network layer sensor. It can also be a single Grid service or Web service sensor that detects the abnormal behavior of SOAP messages or a defender of Denial of Web Service (DoWS) [27].

Most external components and applications interface with Grid-IDS through sensors. In section 4, we will show a typical interfacing between Grid-IDS and applications through sensors.

III. PROTOCOL-AWARE CORRELATION SERVICE

A Grid system, distinguishing itself from an arbitrary resource sharing platform, should have built-in federation and security protocols, for example WS-Sec*, federated trust protocols, policy frameworks, delegation protocols, AAA, SLA, etc. Any resource sharing should be built above these protocols. In this section, we will investigate how to rely on the context information from these protocols for correlation and detection of cross-domain attacks.

Event correlation is currently a heavily targeted research topic for network intrusion detection. It aims to detect attacks or attack attempts by correlating different events/alerts from different sources. Correlation technologies are therefore promising in detecting cross-domain attacks with the support from the underlying protocols. In the following sections, we introduce the developed generic correlation service framework. The generalization is based on the idea of abstraction. Each participant in a communication protocol is described in an abstract fashion denoted by a role. By this, every protocol related to federated services will be integrated into the correlation service by defining the involved roles as a relationship. The correlation service is then able to recognize abnormal behavior by correlating different alerts according to the protocols they belong to.

In the following we introduce and discuss how federation protocols can be generalized in order to achieve this goal. For our correlation service, the objective is to correlate the different alerts caused by the same federation protocol, so that we can detect with a high probability that one cross-domain attack occurs through this protocol. Still, as an example, the attack scenario presented in section I is used to illustrate how the relationship among roles can be applied for alerts correlation.

In Fig. 6 two traditional IDS on the machines hosting the IdP and the banking service are deployed as Grid-IDS sensors. IDS-Sensor₁ will capture login failure attempts and IDS-Sensor₂ will record abnormal money transactions (such as abnormal transferring account or high amount transfer). Although IDS-Sensor₁ and IDS-Sensor₂ can detect and report attacks when they notice abnormal behavior in their sphere, the alert detections are limited to the local site and are not linked between the federated domains. Furthermore, the fact that many legal users will generate such kind of events from time to time as well, one of the sensors might report them as a possible intrusion (false-positive alert) since it has no other information source to cross-check. With the correlation service, we will aggregate and correlate different events from several sources in order to link the events and discover cross-domain attacks and to detect real attacks with higher accuracy reducing the rate of false-positives.

Considering the simple case shown in Fig. 6: IDS-Sensor₁ and IDS-Sensor₂ send one alert respectively to the correlation service, what information is useful to correlate these two alerts? As we mentioned before, our correlation service is to run above the underlying federation protocols and orchestrate the successful intrusion detection. From IDS-Sensor₁ and IDS-Sensor₂ side, first, they should know that they are involved in a Liberty ID-FF1.2 based federation and single-sign-on protocol; second, they should be aware about to what role they are

attached in this protocol, for example, $IDS-Sensor_1$ is attached to the identity provider and $IDS-Sensor_2$ is attached to the service provider. From the correlation service side, with the federation protocol aware event information from $IDS-Sensor_1$ and $IDS-Sensor_2$, it can correlate different events according to the protocols. Therefore, each federation protocol represents a unique relationship of roles. The dimension of a relationship is the number of maximum roles an attacker can affect automatically using a single compromised account in a cross-domain attack. For example, in Fig. 6, the dimension for the Liberty federation and single sign-on relationship is 3, namely {user, identity provider, service provider}.

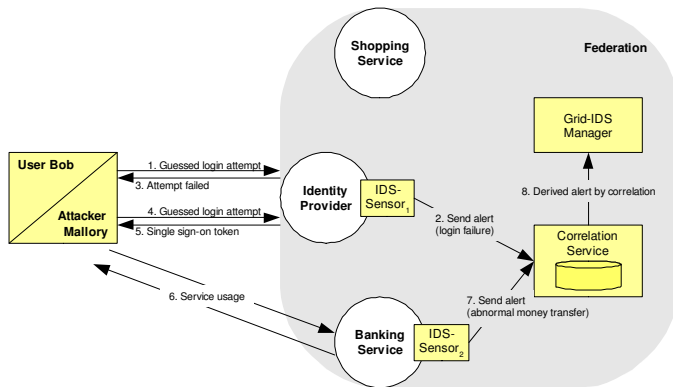


Figure 6. Usage of Grid-IDS

In summary, by enriching the event information of traditional IDS to become Grid protocol-aware, from the correlation service viewpoint, different Grid protocols can be generalized to the role-based relationship model presented here.

A. Related Work

Federated services are composed and constructed using several different protocols and technologies which are usually evolving quickly. This fact requires that the design of an attack correlation service must be easily adaptable to a diversity of protocols and their deployments. Therefore we investigated the existing work on correlation technologies [28]. A brief summary is given in the following:

- *Rule-based correlation*: this model is derived from expert systems and mainly includes an inference engine and a set of correlation rules. Although this model can express the generic relationships among facts, the system is driven by strict predication logic (if-then, precondition \rightarrow facts, etc). The strict inference systems have a performance issue, as pointed out in [28]. Furthermore the detection capability is limited resulting in the overlooking of new attacks that are usually not expressed by existing rules. Finally, from our current analysis, the rule-based correlation approach does not match well to the requirements of our targeted correlation service, due to the restricted and very specific way of expressing rules using if-then statements.
- *Attack scenario based correlation*: this approach usually first models the target scenarios and then generalizes them into a language. Most languages in

this approach are some derivation from the first order logic used in rule-based systems. For example, in [29], the authors define a pre-condition set and a post-condition set for each alert. Two alerts are correlated if one of the pre-conditions of a latter alert is satisfied by one post-condition of an earlier alert. Compared with the rule-based correlation approach, these languages add some flexibility into the inference by supporting partial matching. According to [29], the performance can be improved, but users have to define the pre-conditions and post-conditions sets that are attack scenarios oriented. As for the rule-based correlation approach, the way of expressing correlation rules is not applicable to the requirements of our targeted correlation service.

- *Context reasoning based correlation*: this approach distinguishes itself from the above two by extracting the features or context of the target protection system, and formalizes a model according to the context. Depending on the underlying features, the models here can be quite different from each other. For example, a model can be a state transition graph, or several graphs that refer to each other. As not all correlation facts should be in the form of if-then necessarily, this approach is closer to our correlation requirements than the aforementioned ones.

According to the analysis of the existing work on correlation services for network intrusion detection systems, the context reasoning based correlation scheme is most suitable to be deployed in a Grid-IDS in comparison to the other described approaches. The correlation service developments are therefore based on this scheme combined with the idea to use the relationships among roles involved in Grid protocols as the context for correlation. This will be described in the following section.

B. Framework

Our protocol-aware correlation framework is designed with the consideration of two questions:

- How to design a correlation database to support the dynamic insertion of new protocol and triggering condition information?
- How to generate the correlation events and propagate them automatically?

The first question arises from the requirement of multiple protocols support in our correlation service. As we argued in section III, with so many different potential federated service protocols existing or emerging in the real world, the design of a correlation service needs to support a flexible adoption to the variety of protocols. The ideal approach should be generic. As Fig. 7 shows, in our work, the users can add new protocols or triggering conditions through the Web service interfaces of Correlation Service for monitoring and protection. Basically, the protocol information includes the protocol name as well as the roles involved in it, and any triggering condition is a combination of a protocol name, identified roles and a threshold. We will present a flexible correlation database in

section III.B.3 to support the dynamically added protocols and triggering conditions.

The second question comes from the fact that our correlation engine involves two different resource types: services and databases. The question of how to generate and propagate a correlation event that may span both services and the database is not so obvious at least from the engineering viewpoint.

We describe the high level components and their interactions as shown in Fig. 7 first, and then delve into the above two questions in the following subsections. The main components in our correlation engine are: three services and one correlation database. The Correlation Service provides the interfaces for users to add the protected federation protocol information into the correlation database. The GMAConsumer Service is a part of our WSN-GMA, and it takes charge of receiving the basic alerts from several sensors and storing them into the correlation database. The GMAProducer Service, as a part of WSN-GMA as well, will send the generated correlation events to some more advanced correlation services or to an administrator directly. The correlation database is the core of our correlation engine, and performs the dynamic information update and actual correlation work.

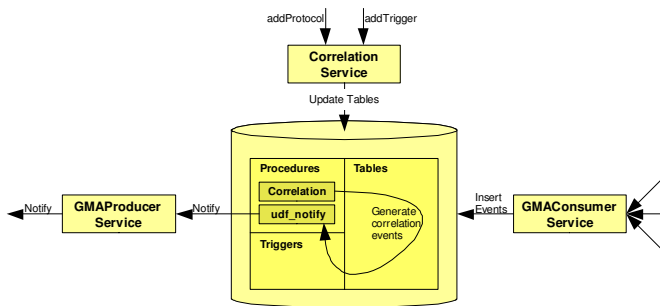


Figure 7. Components of the correlation engine

1) Correlation Event Data Structure

In order to provide the generic correlation service with the necessary information, the sensors generate and send events with the awareness of the underlying federation protocols. The alert or event format is defined by the following XML Schema specification:

```
<xsd:schema>
<xsd:element name="CorrelationEvent">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="ProtocolName"
type="xsd:string" minOccurs="1" maxOccurs="1" />
<xsd:element name="Role" minOccurs="1"
maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Name" type="xsd:string"
minOccurs="1" maxOccurs="1" />
<xsd:element name="Identity" type="xsd:anyType"
minOccurs="1" maxOccurs="1" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="Weight" type="xsd:float"
minOccurs="1" maxOccurs="1" />
<xsd:element name="Detail" type="xsd:anyType"
minOccurs="0" maxOccurs="1" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

```
minOccurs="1" maxOccurs="1" />
<xsd:element name="TimeStamp" type="xsd:dateTime"
minOccurs="0" maxOccurs="1" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

The four first-level elements that must occur include

- the federation protocol name (ProtocolName) the event is related to,
- the roles (InvolvedRoles) related to this event,
- the probability of this event (Weight; the higher the value the higher the probability) that indicates a real attack, and
- the actual event format and reason (Detail).

We treat the TimeStamp element as an optional element, because correlation technologies based on the occurrence time are not very useful once attackers get aware of these time settings and adapt the interval of two sequential attacking steps accordingly. However, in some special cases with session concepts, a certain step must take place in a certain time in order to avoid the timeout of the session. Then the TimeStamp element could be useful for correlation.

For example, the two alerts shown in Fig. 6 can be expressed as shown in Tab. I (we use the table format to make them easier to read). The details can be expressed in any format, such as the Intrusion Detection Message Exchange Format (IDMEF) [30], which is supported by many existing IDS.

TABLE I. EXAMPLE ALERT/EVENT DATA

Protocol Name	Involved Roles	Weight	Detail
Liberty ID-FF1.2	{userName=Bob, identityProvider=IdP}	1	Bob failed to log into the IP at its federation Web site.
Liberty ID-FF1.2	{useName=Bob, identityProvider=IdP, serviceName=BankingService}	2	Bob transferred an unusual amount of money to an suspicious account.

2) Service Interfaces

In order to protect different Grid protocols, we provide a Web service interface in the correlation service for users to add their protocols. Each protocol will be translated into a relationship in the database. For the role based correlation conditions, we provide another interface for users to define the correlation conditions for their protocols. These conditions will be translated into the combined operations on the relationships (Views in the database). The two Web service interfaces are below:

```
String addProtocol(
String protocolName, String[] rolesList);

String addTrigger(
String triggerName, String protocolName,
String[] rolesList, float triggerWeight);
```

When adding a new protocol through the interface above, a relationship table will be created in the database. Thus, we map each protocol into a table. For example, when the users want to add the Liberty ID-FF 1.2 protocol into the correlation service for correlation and intrusion detection, they will invoke

```
addProtocol(
    "liberty_federation",
    {"userName", "idProvider", "srvProvider"});
```

3) Correlation Database

For each new added protocol, from the concept level, the database needs to create 2 tables: one is the table to store the basic alerts (we use protocolName to stand for), and another is the table to store the correlation events (we use protocolName_events to stand for). For each added triggering condition, the database needs to insert one new item into a table which stores the triggering conditions. Once a new basic alert is inserted into the table protocolName, a correlation procedure is triggered, which correlates the items of the table according to the triggering conditions relating to this protocol. The items outweighing the threshold of any triggering condition will be aggregated together as a correlation event. In addition, in order to support multiple protocols, the correlation procedure should have a protocol name as the input parameter.

Before we describe the correlation procedure, we need to clarify some potential technical questions here first. One question is related to how to identify which protocol the incoming alert belongs to. As we mentioned, the sensors will envelope the protocol information including the protocol name and the involved roles as part of every alert and send it to the correlation service, so the correlation database can identify easily in which table to store this alert to. Another question is related to how to map different role Ids to the same name. This question is common in federated services. For example, in the alerts sample of section III.B.1, both names in the role of username are Bob. But if we consider the pseudonymous approach in Liberty ID-FF1.2 protocol, the original alerts have different pseudonymous names of Bob in fact. Therefore, before we invoke the correlation procedure, we must use some identity mapping or token services (in Liberty ID-FF1.2, IdP is an identity mapping service as well) to map these two pseudonymous names to the same role identity. The algorithm used to correlate the stored alert and event data is shown in Fig. 8.

C. Implementation

We have implemented a prototype of this correlation engine with one limitation introduced by the underlying open source database. The services are implemented as Apache Axis [31] Web services and the database is implemented in MySQL [32]. In order to implement the automatic event generation and propagation, we embedded two triggers on the insertion operation in the protocolName and protocolName_events tables respectively. Thus, once a basic alert is inserted into the protocolName table, the operation will trigger the invocation of the correlation procedure, which will insert some correlation events into the protocolName_events table, which triggers the invocation of the udf_notify procedure. udf_notify is a user defined C procedure that performs as a WSN-GMA publisher client. We have implemented it on the open source C/C++ Web

service platform gSOAP [33]. Since the WSN-GMA producer that will receive notifications from udf_notify is implemented in Axis/Java, the interoperability of these two Web service platforms is necessary. Besides the basic Web service interoperability, our implementation supports the interoperability of WS-Addressing between these two platforms as well.

A limitation in our current prototype implementation is resulting from MySQL and is related to the usage of the correlation procedure in the trigger. Since MySQL does not support the invocation of SQL statements with parameters in a trigger even from its latest version, we have to create one custom correlation procedure for each new added protocol in order to adapt to this limitation and can not reuse the one in section III.B.3 for all. Note that for most commercial databases, this kind of limitation does not exist.

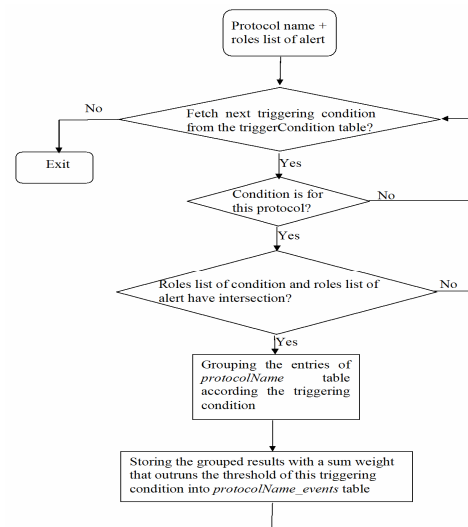


Figure 8. Correlation Algorithm

IV. INTERACTIONS' DESCRIPTION

In this section, we will give a typical interaction scenario between Grid-IDS and other components or applications. The UML sequence diagram for this interaction is shown in Fig. 9. The sensor component in this diagram can be embedded into any external component or application that needs to be protected by Grid-IDS.

In this scenario, first, a Grid-IDS manager will provide some information to the correlation service about the target Grid or Web service protocol that needs to be monitored and protected through the service methods of AddProtocol and AddTrigger (step 1, 2). After that, the manager can subscribe to this correlation service in order to receive correlation events (step 3).

Imagine there are several sensors embedded in different Grid sites, which will interact with each other using one Grid or Web service protocol. We should notice here this protocol is exactly the one registered in the first steps by the Grid-IDS manager. Each sensor will advertise its existence by invoking the add method of the DirectoryService (step 4). In the same way, the correlation service will announce its existence (step

5). In order to find all sensors related to one Grid protocol, the correlation service can search them by a topic name, which is the Grid protocol name in this case (step 6). As part of the search response, the matching sensors' addresses are returned to the correlation service. In addition, if users use the security framework presented in [25], the secure conversation tokens for each matching sensor will be generated by the DirectoryService and returned as part of the search response as well. The correlation service then subscribes to each sensor in order to receive alerts from it (step 7).

Once an alert appears, the sensor will notify the correlation service automatically (step 8), which will trigger the correlation procedure (step 9). Again, if some new correlation events are generated, the correlation service will notify the Grid-IDS manager automatically (step 10).

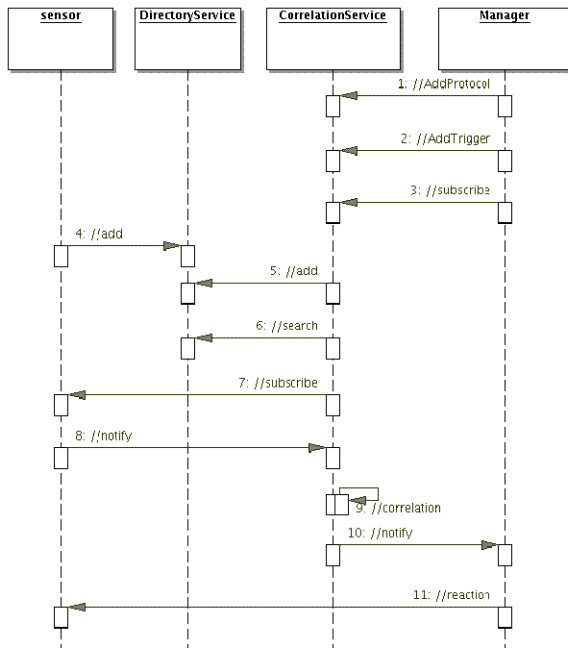


Figure 9. Interaction of Grid IDS components

As the last step, the Grid-IDS manager will decide what reactions are necessary. Optionally, some reactions can be fed back to the affected sensors for further processing, for example, shutting off the connections from the compromised users, etc (step 11).

V. TOWARDS INTRUSION TOLERANCE

Intrusion-tolerant systems must be able to continue to operate in the presence of compromised entities. A total close-down after the detection of an intrusion is in contradiction to this requirement and most possibly not tolerable for mission critical systems. Moreover, in distributed and especially in inter-enterprise information systems a total close-down is usually not acceptable since it requires coordinated manual actions including all involved administrative domains. More appropriate intrusion management and response mechanisms have to be in place in order to automatically detect changes in the security status and hence achieve the goal of an intrusion-tolerant system. To develop intrusion-tolerant SOA-based

systems both, methodologies for designing-in intrusion tolerance during system development as well as mechanisms for detecting and managing intrusions during operation are required.

To further evolve the presented intrusion detection mechanisms into an effective technology to enable distributed service eco-systems to be tolerant against intrusions, various attempts can be taken. A considerable challenge here is for example the fruitful adaptation of the intrusion detection management components to system-internal components. This enables for instance the interaction with SLA components to dynamically adapt SLA-based relationships on the changed security status or with intrusion prevention components to automatic exclude a compromised entity while keeping the whole system operating for the uninvolved authorized entities. First attempts towards an intrusion-tolerant Grid have been undertaken within the NextGRID project [34] based on the Grid-IDS and security developments. An according experiment has demonstrated a tighter interaction and mutual assistance of these components in order to automatically exclude compromised users. However, these first steps have to be considered as really basic ones, leaving a lot of open research and development questions open as discussed in the following.

To analyze and specify which intrusions are tolerable under what conditions and in which contexts, their occurrence has to be additionally considered during the risk assessment in the design phase of SOA-based systems. For each identified intrusion the probability of the occurrence and its impact to the system's assets have to be quantified as well as the necessary measures to isolate the whole system from the compromised parts while keeping the unaffected ones available. These enhancements have to be integrated into the whole security engineering process including analyses and design methodologies as well as in the security lifecycle management process for interactive reactions and refinements (see Fig. 10). The output of the risk assessments are appropriate policies, where such policies have to be expressed in some kind of policy language in order to be enforced by the intrusion prevention components. In this context the existing policy mechanisms and languages such as WS-Policy [35] or XACML [36] have to show their applicability and proof whether they are powerful enough to express the necessary statements for intrusion tolerance or not. Besides the identification of intrusions and definition of conditions under which intrusions are tolerable, interfaces between the intrusion management component and the system's intrinsic components have to be specified and implemented.

Other issues, which might include challenging research questions are the detection and management of insider intrusions. Finally, the intrusion tolerance system has to be assessed in order to analyze and evaluate if it introduces security problems itself such as vulnerabilities to Denial of Service (DoS) attacks or privacy issues.

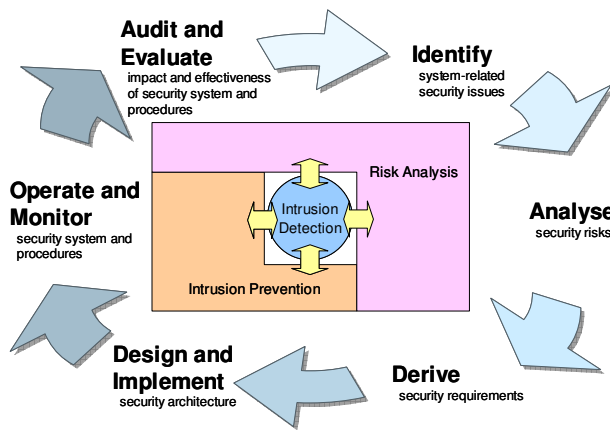


Figure 10. Obtaining intrusion tolerance properties

VI. CONCLUSION

In this paper, we presented an IDS that aims to protect federated services against cross-domain attacks. One distinguishing feature of our Grid-IDS framework compared with existing work is its service-based components and implementation. This feature makes our Grid-IDS share two main advantages of service-oriented computing: (i) easy integration with heterogeneous application environments as well as sensor types; (ii) flexible configuration and deployment in various Grid-based applications.

Our correlation approach is based on protocol context information. By generalizing each target protocol to relationships of roles and integrating protocol-aware alert information, our correlation engine can group the alerts according to the target protocol. This relationship of roles based correlation approach can be naturally implemented in any relational database.

Finally, we discussed initial attempts to evolve the developed Grid-IDS into a building block for intrusion tolerant service-oriented systems and henceforth providing a step stone towards dependable SOA. Further work will be done to investigate the discussed challenges in this context.

REFERENCES

- [1] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "Grid Services for Distributed System Integration," *Computer*, 35(6), 2002.
- [2] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, "Reference Model for Service Oriented Architecture 1.0," OASIS, 2006.
- [3] D. Booth, H. Haas, F. McCabe, E. NewComer, M. Champion, C. Ferris, and D. Orchard, "Web Service Architecture," W3C, 2004.
- [4] Enabling Grids for E-science (EGEE), <http://www.eu-egee.org/>.
- [5] TeraGrid, <http://www.teragrid.org/>.
- [6] H. Jin, "ChinaGrid: Making Grid Computing a Reality," in *Proceedings of ICADL 2004*, LNCS 3334, pp. 13-24, 2004.
- [7] T. Wason, S. Cantor, J. Hodges, J. Kemp, and P. Thompson, "Liberty ID-FF Architecture Overview," Liberty Alliance Project, 2004.
- [8] A. Nadalin (Ed.), C. Kaler (Ed.), et al., "Web Services Federation Language (WS-Federation)," 2006.
- [9] A. Alves, et al., "Web Services Business Process Execution Language Version 2.0," OASIS, 2006.
- [10] T. R. Metcalf and L. J. LaPadula, "Intrusion Detection System Requirements: A Capabilities Description in Terms of the Network

- Monitoring and Assessment Module of CSAP21," MP 00B0000046, MITRE Corporation, September 2000.
- [11] M. Wood and M. Erlinger, "Intrusion Detection Message Exchange Requirements," IETF Internet Draft, 22 October 2002.
- [12] P. A. Porras and P. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," in *Proceedings of the 20th NIST-NCSC National Information Systems Security Conference*, 1997.
- [13] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS – A Graph Based Intrusion Detection System for Large Networks," in *Proceedings of the 19th National Information Systems Security Conference*, vol 1, 1996.
- [14] J. S. Balasubramanian, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents", Coast TR 98-05, Department of Computer Science, Purdue University, 1998.
- [15] G. B. White, E. A. Fisch, and U. W. Pooch, "Cooperating Security Managers: A Peer-based Intrusion Detection System," *IEEE Network*, pp. 20-23, 1996.
- [16] C. Krügel and T. Toth, "Distributed Pattern Detection for Intrusion Detection," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2002.
- [17] F. C. Colon Osorio and X. Wang, "Applying Byzantine Agreement Protocols to the Intrusion Detection Problem in Distributed Systems," Worcester Polytechnic Institute, Security Systems Research Laboratory, WSSRL-TR-0301, 2003.
- [18] SPARTA. Security Policy Adaptation Reinforced Through Agents. IST – 12637. www.infosys.tuwien.ac.at/sparta
- [19] M. Asaka, S. Okazawa, A. TAGUCHI, and S. GOTO, "A Method of Tracing Intruders by Use of Mobile Agents," *INET'99 Conference*, 1999.
- [20] S. Fenet and S. Hassas, "A Distributed Intrusion Detection and Response System Based on Mobile Autonomous Agents Using Social Insects Communication Paradigm," *First International Workshop on Security of Mobile Multiagent Systems*, 2001.
- [21] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolski, "A Grid Monitoring Architecture," *GFD-1.7, OGF*, 2002.
- [22] Web Services Notification. OASIS Standard, 2006.
- [23] W. Vambenepe(Ed.), et al., "Web Services Distributed Management: Management Using Web Services (MUWS 1.0)," OASIS Standard, 2005.
- [24] Apache Pubsub, <http://ws.apache.org/pubsub>.
- [25] J. Wang, "A Web Services Secure Conversation Establishment Protocol Based on Forwarded Trust," *IEEE International Conference on Web Services*, 2006.
- [26] Apache Muse, <http://ws.apache.org/muse>.
- [27] J. Wang, "Defending Against Denial of Web Services Using Sessions", *IEEE/IST Workshop on: Monitoring, Attacking Detection and Mitigation (MonAM 2006)*, 2006.
- [28] P. Fabien and D. Marc, "Alert Correlation: Review of the state of the art," *Eurecom Research Report RR-03-093*, 2003.
- [29] P. Ning, Y. Cui, and D. S. Reeves, "Analyzing Intensive Intrusion Alerts Via Correlation," in *Proceedings of Recent Advances in Intrusion Detection (RAID)*, 2002.
- [30] H. Debar, D. Curry, and B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)", *IETF RFC 4765*, 2007.
- [31] Apache Axis, <http://ws.apache.org/axis/>.
- [32] MySQL, <http://www.mysql.org/>
- [33] gSOAP, <http://www.cs.fsu.edu/~engelen/soap.html>.
- [34] NextGRID, <http://www.nextgrid.org/>.
- [35] A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, and Ü. Yalcinalp, "Web Services Policy 1.5 Framework (WS-Policy)", W3C Candidate Recommendation, 2007.
- [36] T. Moses(Ed.), et al., "eXtensible Access Control Markup Language (XACML) Version 2.0", OASIS Standard, 2005.