Process Orchestration for Intrusion Detection System based on SOA and Event Driven Architecture Principles

K.V.S.N.Rama Rao [1] , Pandu Prudhvi [2] , Manas Ranjan Patra[3]
[1]BSIT , Hyderabad, India
[2]TechMahindra, Hyderabad, India
[3]Berhampur University, Berhampur, Orissa
{kvsnramarao@yahoo.co.in , pswamy.2009@gmail.com, mrpatra12@gmail.com }

Abstract: *As the dependency on the internet in the recent years is increasing greatly, threats and vulnerabilities were also rising in sync to it. Several security systems like Intrusion detection systems were developed to battle against these threats. But the existing intrusion detection systems were not succeeding against these threats, as they are unable to address challenges that surround different types of attacks. These systems are designed to deliver the best performance but not able to deal with some attacks because they lack service oriented architecture to support increasingly diverse clients with various network and device capabilities. It is evident that no single technique can guarantee protection against future attacks. .Hence there is a need for integrated architecture which can provide robust protection against a complete spectrum of threats. In this paper, we propose a SOA based architecture model for IDS and its process orchestration based event driven architecture.*

Keywords: *IDS, vulnerability, SOA, EDA, web services, architecture.*

1. Introduction

With the growing use of Internet, attackers are becoming active in identifying the flaws in operating systems, underlying network protocols, and different software implementations. They are able to make sophisticated attacks on information resources. As a defense it is most common to use host based solutions like antivirus software, fire walls etc. These approaches have drawbacks in being insufficiently fast to meet new threats. Now a day due to globalization, multiple stake holders are involving in the activities of any organization. For example, in the case of IT projects several stake holders like end users, customers, vendors, legal entities and many others are involved to complete the project successfully.

In such a distributed and heterogeneous setup, security policies and their implementations suffer from the inability to cope with the flexibility of multi-site and multi-organization rules and the rigidity of a strong de-militarized zone. In this paper, we discuss the service oriented approach to build intrusion detection systems. This approach has the following key features:

i) Helps in identifying and analyzing the tasks performed by an IDS at a higher-level of abstraction.

ii) Helps in designing and building independently scalable components to deal with different aspects of an attack scenario.

iii) Helps in modeling the interactions among these components in an efficient and flexible manner

iv) Helps in adding new services when necessary.

2. Intrusion Detection

Intrusion [1] is defined as set of actions aimed to compromise the security goals. Intrusion Detection is the process of identifying and responding to intrusion activities. While modeling IDS we assume that normal and intrusive activities have distinct evidence and the system activities are observable. Any IDS have few important components [1].

a)Sensor or Agent: It Monitors and analyze network activity

b) Detection Engine: It contains rules and various detection models.

c) Decision Engine: On receiving alarm from detection engine it takes appropriate action and generates report.

Any typical IDS will focus on three areas of detection methodologies.

a) Signature based detection: A signature [1] is a pattern that corresponds to a known threat. Signature-based detection is the process of comparing signatures against observed events to identify possible incidents, Examples of signatures are: A telnet attempt with a username of “root”, which is a violation of an organization’s security policy, an e-mail with a subject of “Free pictures!”, and an attachment filename of “freepics.exe”, which are characteristics of a known form of malware. Signature-based detection cannot track and

understand the state of complex communications, so it cannot detect most attacks that comprise multiple events.

Hackers [2] often attack networks through tried and tested methods from previously successful assaults. These attacks have been analyzed by network security vendors and a detailed profile, or attack signature, has been created. Signature detection techniques identify network assaults by looking for the attack fingerprint within network traffic and matching against an internal database of known threats. Once an attack signature is identified, the security system delivers an attack response, in most cases a simple alarm or alert.Success in preventing these attacks depends on an up-to-the-minute database of attack signatures,compiled from previous strikes. The drawback to systems that rely mainly, or only, on signature detection is clear: they can only detect attacks for which there is a released signature. If signature detection techniques are employed in isolation to protect networks, infrastructure remains vulnerable to any variants of known signatures, first-strike attacks, and Denial of Service attacks.

b) Anomaly-based detection: It compares definitions [1] of what activity is considered normal against observed events to identify significant deviations. This method uses profiles that are developed by monitoring the characteristics of typical activity over a period of time. The IDS then compares the characteristics of current activity to thresholds related to the profile. Anomaly-based detection methods can be very effective at detecting previously unknown threats. Common problems with anomaly-based detection are inadvertently including malicious activity within a profile, establishing profiles that are not sufficiently complex to reflect real-world computing activity, and generating many false positives.

Anomaly detection [2] techniques are required when hackers discover new security weaknesses and rush to exploit the new vulnerability. When this happens there are no existing attack signatures. The Code Red virus is an example of a new attack, or first strike, which could not be detected through an available signature. In order to identify these first strikes, IDS products can use anomaly detection techniques, where network traffic is compared against a baseline to identify abnormal—and potentially harmful—behavior. These anomaly techniques are looking for statistical abnormalities in the data traffic, as well as protocol ambiguities and atypical application activity. Today's IDS products do not generally provide enough specific anomaly information to prevent sophisticated attacks and if used in isolation, anomaly detection techniques can miss attacks that are only identifiable through signature detection.

c) Denial of Service (DoS) Detection [2]

The objective of DoS and Distributed DoS attacks is to deny legitimate users access to critical network services. Hackers achieve this by launching attacks that consume excessive network bandwidth or host processing cycles or other network infrastructure resources. DoS attacks have caused some of the world's biggest brands to disappoint customers and investors as Web sites became inaccessible to customers, partners, and users—sometimes for up to twenty-four hours. IDS products often compare current traffic behavior with acceptable normal behavior to detect DoS attacks, where normal traffic is characterized by a set of pre-programmed thresholds. This can lead to false alarms or attacks being missed because the attack traffic is below the configured threshold.

IDS can be deployed at the following places to monitor activities.

a) Host based IDS: which monitors the characteristics of a single host and the events occurring within that host for suspicious activity.
Ex: Analyze shell commands, Analyze system calls made by send mails etc.

b) Network-Based IDS: this monitors network traffic for particular network segments or devices and analyzes the network and application protocol activity to identify suspicious activity.
Ex: Watch for violations of protocols and unusual connection patterns, look into the data portions of the packets for malicious command sequences etc.

In order to robustly protect enterprise network against the complete spectrum of threats and vulnerabilities, there is a need for robust architecture. But due to the lack of superior architectural support, current IDS are facing various challenges which are discussed below.

3. Current IDS Challenges

Intrusion Detection Systems today are facing several challenges [2].

Incomplete attack coverage: IDS products typically focus on Signature, Anomaly, or Denial of Service detection. Network security managers have to purchase and integrate point solutions from separate vendors or leave networks vulnerable to attack.

Inaccurate detection: IDS products' detection capabilities can be characterized in terms of accuracy and specificity. Accuracy is often measured in true detection rate—sometimes referred to as the false negative rate—and the false-positive rate. The true detection rate specifies how successful a system is in detecting attacks when they happen. The false-positive rate tells us the likelihood that a system will misidentify benign activity as attacks. Specificity is a measure of how much detailed information about an attack is discovered when it is detected. IDS products today are lacking in both accuracy and specificity and generate too many false-positives, alerting security engineers of attacks, when nothing malicious is taking place. In some cases, IDS products have delivered tens of thousands of false-positive alerts a day. There is nothing more corrosive to network vigilance than a jumpy security system, which is continually issuing false alarms.

Detection, not prevention: Systems concentrate on attack detection. Preventing attacks is a reactive activity, often too late to thwart the intrusion.

Designed primarily for sub-100Mb/s networks: Solutions have simply not kept up with the speed and sophistication of network infrastructure and cannot accurately monitor higher-speed or switched networks.

Performance challenged: Software applications running on general purpose PC/server hardware do not have the processing power required to perform thorough analysis. These underpowered products result in inaccurate detection and packet dropping, even on low bandwidth networks.

Lack of high-availability deployment: Single port products are not able to monitor asymmetric traffic flows. Also, with networks becoming a primary mechanism to interact with customers and partners, forward-thinking organizations have developed back-up systems should their current infrastructure fail in any way. The inability of current IDS products to cope with server failovers renders them virtually useless for any mission-critical network deployment.

Poor scalability: Primarily designed for low-end deployments, today's IDS products do not scale for medium and large enterprise or government networks. Here monitored bandwidth, the number of network segments monitored, the number of sensors needed, alarm rates, and the geographical spread of the network exceed system limits.

No multiple policy enforcement: Current products generally support the selection of only one security policy for the entire system, even though the product may monitor traffic belonging to multiple administrative domains—in an enterprise this could be the finance, marketing, or HR functions. This one size fits all approach is no longer acceptable for organizations that require different security policies for each function, business unit, or geography.

Require significant IT resources: IDS products today require substantial hands-on management—for example,the simple task of frequent signature updates can take up a lot of time and skilled engineering resources,delivering a very high total cost of ownership. Concisely, one can state that many of the IDS implementations are not designed to co-operate. To address these challenges, a new architecture needs to be developed for even the most demanding enterprise networks.

Hence we propose an architecture that works for problem of a multi-site IDS for a multi-business scenario, shown in Figure.1, where each business can have a custom defined set of rules implemented at each location of choice.

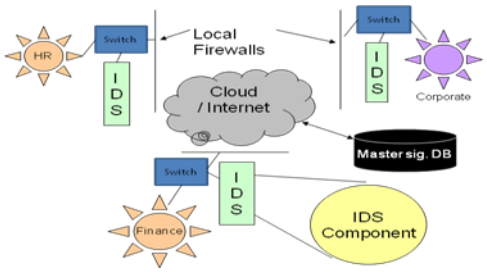

Figure 1: Distributed IDS deployed across mutli-location corporate network

4. Model Architecture for Proposed IDS

Figure 2 summarizes the architecture of the fast Ethernet IDS system designed.

The proposed solution contains a Cache component that collects network packets.

The functionality of the components are explained below.

Sampler: The Sampler randomly/heuristically picks up sample packet windows (series of contiguous packets) and sends them to the Network Packet Analyzer component. The sampling can be done in a random fashion or by using a heuristic.

Network Packet Analyzer: The Analyzer and the Pre-processing engine analyze the packets and convert them into a standard XML format by stripping the network and DLL headers. This metadata is sent for processing to the next component i.e. the "Rules Engine" which can be an SOA component.

Business Rules Engine: The Rules engine is a SOA [3] enabled component of the application that facilitates the XML packet to be checked for anomalies against suspicious activities and pre-defined business rules. This component should be able to detect packets from invalid/untrusted IPs and domains. DoS attacks, Filtering, Screening, Authentication, Trust, etc. related issues can be addressed at this component. The Rules engine should be SOA enabled to allow the organization to implement and customize the rules based on the location of the IDS on the network. For example in a large enterprise, HR may need a different set of rules implemented as against the finance and there may be some organizational rules applicable to all departments. Rules must be classified as preemptive/non-preemptive. A web-service [4] client can allow for posting of rules to be consumed and for rules to be published [5] from one instance of the IDS to another which is one of the many advantages of a SOA enabled system. The rules engine upon detecting anomaly will automatically forward to alert agent component or manual intervention component.

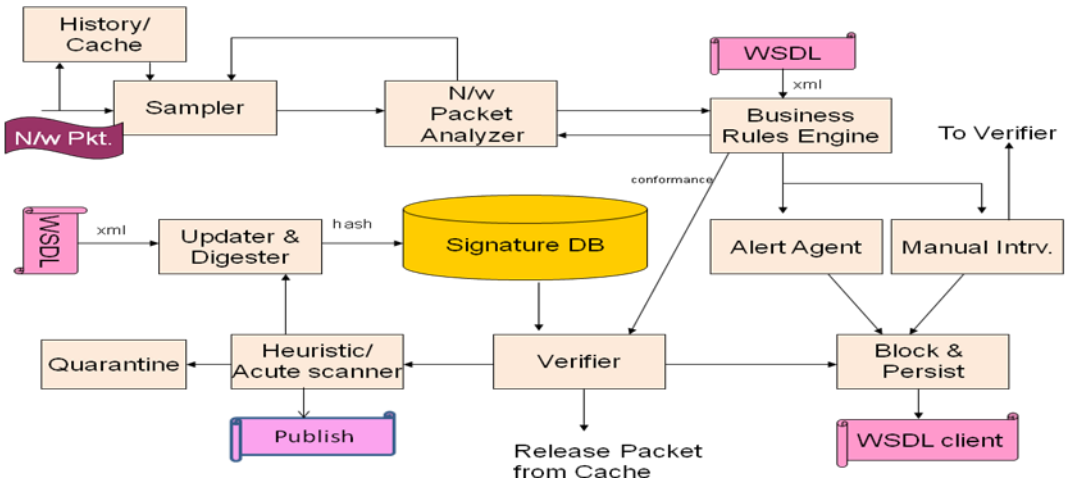

Figure 2: Proposed Model of the IDS component showing the SOA enabled external interfaces and custom built internal components

Alert Agent: If directed to alert agent component then the alerts are audited, logged, and mailed to concerned authorities.

Manual Intervention: On the other hand, if the threat is not that much harmful , they are directed to manual intervention component where they can be manually addressed by administrator of the location. The Manual intervention application may flag to either further analyze or release the associated packets to the verifier. If all packets in a sample packet window are cleared by Business Rules Engine, then the packets go for a check of known attack signatures to the verifier.

Verifier: The Verifier component checks the packets against attacks picked from a local signature database. This DB is pre-populated from external and publicly known signatures and other IDS instance detected signatures. These signatures are also batch mode synchronized between IDS instances through the Updater and Digester service component. Whenever the verifier component detects a known signature match, it immediately discards the packet and the payload. In case of innocuous packets it can inform the cache to release them. For packets that have matched a possible known attack, the packets and the payload can be sent into the heuristic and acute scanner. Since the signatures are hashed, comparing them in the verifier against new XMLs and network packet payloads becomes easy and quick to achieve the "fast Ethernet" speeds that this architecture claims.

Heuristic and Acute Scanner: This will perform further analysis to detect newer form of attacks or decisively declare a packet/source as safe. This can over a period of time detect new attacks and recover from false alarms.

Thus if a payload was marked as possibly harmful, a fuzzy logic AI agent running in the heuristic scanner can verify the safety or the hostility of the payload to a pre-determined degree of threshold (say 25% to 80%) before declaring and publishing it to other instances through the Master DB and also updating the local DB.

Updater and Digester: The updater listens for updates on a daily basis from the Master DB, which is connected on the cloud and sends web-service based publish notices to all instances. The updater then picks up these XMLs and their packet payloads and digests them using fast and compressive hashing algorithms that compact this information and store it in the local signature DB. The updater and digester component in conjunction with the Business Rules engine thus ensures that over a period of time the IDS learns to detect unknown attacks and thus can prevent them as well making it a true IDS

Master Database: It is kept updated through SOA components about attacks detected or false alarms nullified at the distributed locations. The Master Database on the next day updates all IDS instances local databases

Block and Persist: This component fires whenever the Manual intervention module marks a XML cum payload pair as suspicious or malicious or if the alerter escalates a known business rule violation. The component simply publishes the packet to be updated into the local DB via the Updater and Digester service and to the MasterDB which runs a similar Updater and Digester service

In the proposed model, several components such as Business Rules Engine, Block and persist, Heuristic and acute scanner and Updater & Digester are web services.

The advantage of using web-services clients here is that it becomes easy to update the remote DB and the local DB through common interfaces and in future to publish the same to other external service consumers as well, e.g. security provider Databases or public signature databases – on which the current system relies as well.

The best architecture to integrate web services is known to be service oriented architecture (SOA).A brief description of web service, SOA, SOAD Process are discussed in the next section.

5. Service Oriented Architecture

Service Oriented Architecture (SOA) [6] is a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks, or services.

A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. A Web service provides one way of implementing the automated aspects of a given business or technical service.

Services generally adhere to the principles of service-orientation such as abstraction, autonomy, composability, discoverability, formal contract, loose coupling, reusability, statelessness

Why SOA?

SOA helps create greater alignment between IT and line of business while generating more flexibility - IT flexibility to support greater business flexibility. Your business processes are changing faster and faster and global competition requires the flexibility that SOA can provide. SOA can help you get better reuse out of your existing IT investments as well as the new services you're developing today. SOA makes integration of your IT investments easier by making use of well-defined interfaces between services. SOA also provides an architectural model for integrating business partners', customers' and suppliers' services into an enterprise's business processes. This reduces cost and improves customer satisfaction.

SOA is a suitable architecture style when reusability, integration and agility are key concerns for an enterprise.

Basically the four tenets of Service orientation [7] are as follows
• Boundaries are explicit
• Services are autonomous
• Services share schema and contract, not class
• Compatibility is based upon policy

SOA Design Principles

- Deciding what functionality makes sense to expose as a service
- Separating and modularizing the business logic to facilitate reuse and flexibility
- Loosely coupling services to support rapid development when requirements change
- Designing an appropriate granularity of services
- Planning and implementing all the SOAD steps.

5.1 SOAD Process [8]:

The term Process means "sequence of steps required to develop or maintain software"[9].A process deals with what of developing a software while a methodology deals with how of developing a software. With the introduction of object oriented paradigm, OOAD process has been in use extensively.

Object oriented analysis and design process involves modeling real world objects based on the requirements described as a set of use cases, realizing the use cases through a process of identifying the analysis classes(boundary, control and entity) and mapping the analysis to technology elements that constitute the design classes. Classes are fine grained elements that are tightly coupled. Design classes can be implemented through programming and tested to develop the required application. But OOAD Process presents several difficulties [10].Since the OO applications granularity is at class level, there will be tight coupling and strong associations because class hierarchies are based on inheritance. .But on the other hand, services are loosely coupled.

In Service model, there will two important roles. Service Provider who exposes services and Service Consumer who consumes service. There will be a service contract between these two parties to define the type of messages they can exchange or operations they can perform. Also there will be a data contract between the client and the service. These contracts enables loose coupling. The key considerations of service model such as reusability, integration and agility will result in four types of services [11].

Client services: These deliver content to the business users that require an aggregated enterprise view. They provide presentation content to the "front-end" applications of the enterprise such as Portal, dashboard or CRM applications that provide the necessary presentation capabilities and typically are service consumers for the other services in the Enterprise.

Business Process management services (Process services): These allow for externalization of business processes in an orchestrable fashion resulting in agility for the enterprise.

Business Application services (Activity services): These are reusable business level services that can be orchestrated as part of a configured business process.

Data Services (Entity services): These encapsulate access to data in various sources such as ERP, legacy, a data warehouse or a system external to the organizational context

These four services are integrated by Enterprise service bus. Considering these 4 services, SOAD process consists the following steps.

Step1: Gather objectives and business requirements of the application.

Step2: Perform Business Process Modeling (BPM) that involves identification of business processes and workflows that applications in the enterprise would need to support to meet the business objectives. The business process model provides the workflows that may be expressed as Business Process Execution Language(BPEL), configured and orchestrated to generate the Business process services. The business process model generated through BPEL is the key artifact of SOAD Process. This will serve as an input to develop four services which were discussed above.

Step3: Implementation: A technology stack is chosen for implementation of services.

The scope of this paper covers the first two steps of SOAD process. In the following sections, we present Business process modeling generated through Business Process Execution Language (BPEL) for the IDS model architecture described in section 4.BPEL generates work flows and process orchestrations.

6. Process Modeling For IDS Using BPEL:

The process modeling will be done for the architecture that is explained in section 4. In that architecture several components are web services.

Hence the next task in SOAD process is to perform Business Process Modeling using BPEL which expresses the workflows.

Process Orchestration diagram is shown in figure 3.

The corresponding BPEL code given below.

IDSProcess.bpel

<process name="IDSProcess"

targetNamespace="http://ids.security.com/IDS/idsServices/IDSProcess"

xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"

xmlns:client="http://ids.security.com/IDS/idsServices/IDSProcess"

xmlns:ora="http://schemas.oracle.com/xpath/extension"

xmlns:bpelx="http://schemas.oracle.com/bpel/extension"

xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"

xmlns:ns1="http://oracle.com/sca/soapservice/Application1/Project1/NetWrokPacketAnalyser"

xmlns:ns2="http://xmlns.oracle.com/pcbpel/adapter/jms/Application1/Project1/AlertManagerService"

xmlns:task="http://xmlns.oracle.com/bpel/workflow/task"

xmlns:taskservice="http://xmlns.oracle.com/bpel/workflow/taskService"

xmlns:wfcommon="http://xmlns.oracle.com/bpel/workflow/common"

xmlns:ns3="http://oracle.com/sca/soapservice/Application1/Project1/VerifierService"

xmlns:ns4="http://oracle.com/sca/soapservice/Application1/Project1/DigesterService"

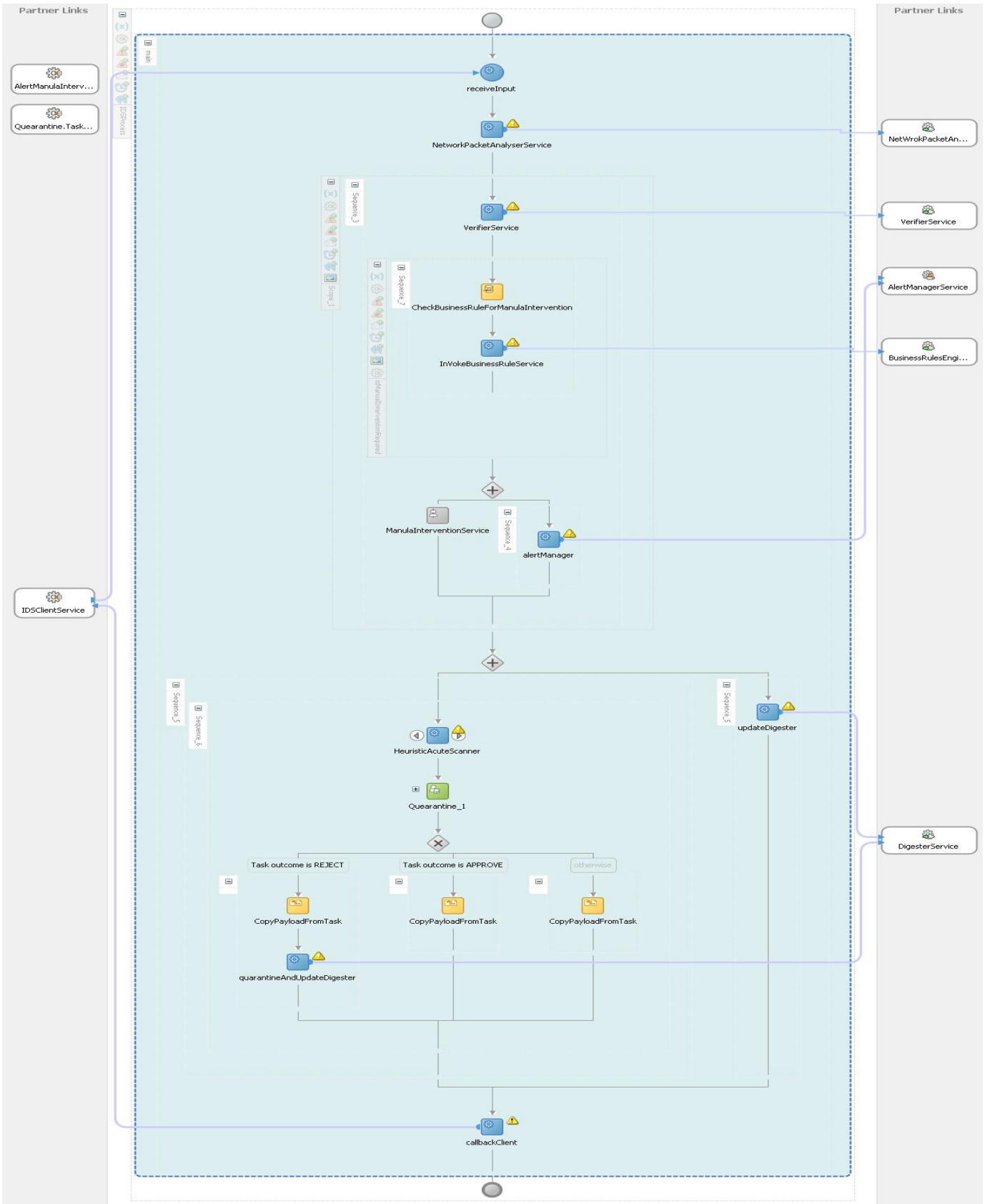

Figure 3: Business Process modeling for IDS

```
xmlns:ns5="http://oracle.com/sca/soapservice/Application1
/Project1/BusinessRulesEngineService">

  <!—<!--
   PARTNERLINKS
    <!--
   The 'client' role represents the requester of this service.
It is  used for callback. The location and correlation
information associated  with the client role are
automatically set using WS-Addressing. - - >

Link myRole="execute_ptt"
name="NetWorkPacketAnalyser"

partnerLinkType="ns1:NetWorkPacketAnalyser"/>
    <partnerLink myRole="alertManager_role"
name="AlertManagerService"
            partnerLinkType="ns2:alertManager_plt"/>
 -->
  <partnerLinks>
```

Request from IDS client service are received as input by a proxy server. The proxy manager will select a sample of packets and directs them to Network packet analyzer through sendPayload() method. The Network Analyzer will connect to its partner links. and convert the packets into a standard XML format by stripping the network and DLL headers. This Meta data is sent for validation against business rules.

The Rules engine is SOA enabled to allow the organization to implement and customize the rules based on the location of the IDS on the network. For example in a large enterprise, HR may need a different set of rules implemented as against the finance and there may be some organizational rules applicable to all departments. A web-service client can allow for posting of rules to be consumed and for rules to be published from one instance of the IDS to another which is one of the many advantages of a SOA enabled system.

The rules engine upon detecting anomaly will automatically forward to alert agent component or manual intervention component.

The orchestration logic is represented below

```
<sequence name="main">
    <!-- Receive input from requestor. (Note: This
maps to operation defined in IDSProcess.wsdl) -->
    <receive name="receiveInput"
partnerLink="IDSClientService"
portType="client:IDSProcess" operation="process"
variable="inputVariable" createInstance="yes"/>

    <!--
     Asynchronous callback to the requester. (Note:
the callback location and correlation id is
transparently handled using WS-addressing.)
    -->
 <invoke name="NetworkPacketAnalyserService"
partnerLink="NetWrokPacketAnalyser"/>
    <scope name="Scope_1">
      <bpelx:annotation>
        <bpelx:general>
          <bpelx:property
name="userLabel">Business Rules
</bpelx:property>
        </bpelx:general>
      </bpelx:annotation>
      <sequence>
        <invoke name="VerifierService"
partnerLink="VerifierService"/>
        <scope
name="isManualInterventionRequired">
          <bpelx:annotation>
            <bpelx:pattern
patternName="bpelx:decide"></bpelx:pattern>
          </bpelx:annotation>
          <sequence>
            <bpelx:checkpoint
name="CheckBusinessRuleForManulaIntervention"/
>
            <invoke
name="InVokeBusinessRuleService"

partnerLink="BusinessRulesEngineService"/>
          </sequence>
        </scope>
        <flow name="Rules">
          <sequence>
            <invoke name="alertManager"

partnerLink="AlertManagerService"/>
          </sequence>
```

The high level design of Rules engine service is shown in the Figure 4.

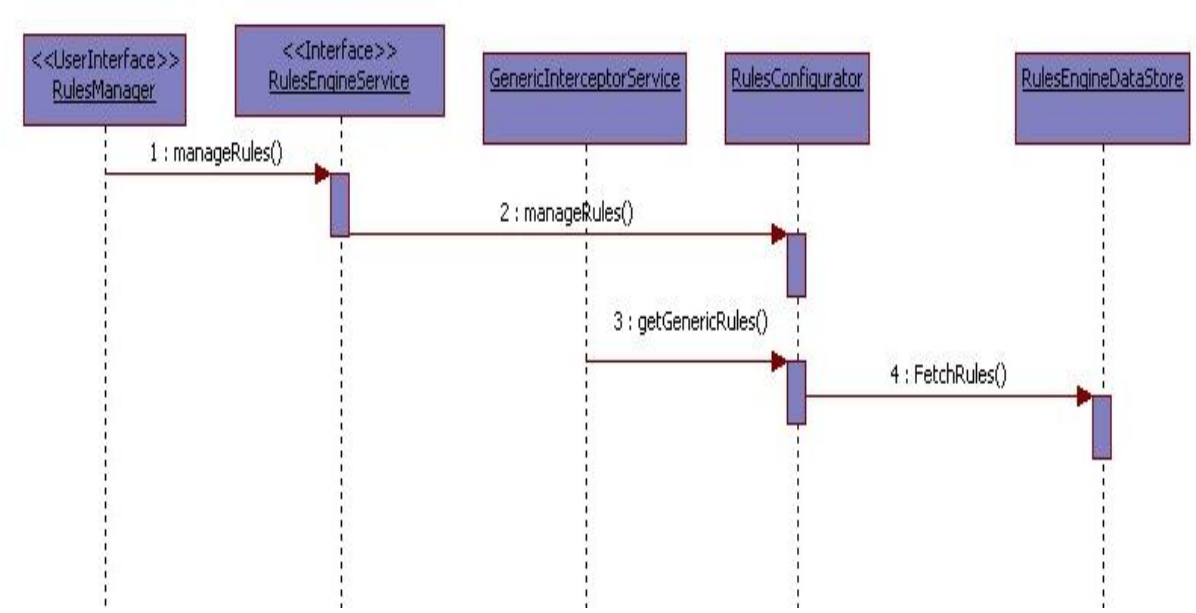

Figure 4: Rules Engine Service

The Rules Manager Interface will contact the Rules Engine Service Interface by using the method manageRules(). This service will invoke Rules configarator. Concurrently another service called Generic Interceptor Service will request for generic rules from the RulesConfigarator through getGenericRules() method. RulesConfigarator will in turn contact RulesEngineDataStore through fetchRules() method and will fetch both generic and specific rules. Hence by using this service, each administrative domain in an enterprise like HR, Marketing etc can customize their own business rules in conjunction with enterprise wide business rules. So the use of this service has conquered one of the challenges of IDS.

After passing the business rules check, the pay load will be directed to verifier for second level of checking. If this check is successful and found that the packet is not harmful, packet will be released otherwise it will be sent for heuristic and acute scanner.

The corresponding orchestration logic is presented below.

```
<sequence name="ManulaInterventionService">
        <sequence>
                <scope
name="AlertManulaInterventionToVerifier_1"

xmlns="http://schemas.xmlsoap.org/ws/2003/03/busi
ness-process/"

xmlns:wf="http://schemas.oracle.com/bpel/extension/
workflow"

wf:key="AlertManulaInterventionToVerifier_1_glob
alVariable">
                <bpelx:annotation
xmlns:bpelx="http://schemas.oracle.com/bpel/extensi
on">
                <bpelx:pattern
patternName="bpelx:workflow"></bpelx:pattern>
                </bpelx:annotation>
                <variables>
                <variable
name="initiateTaskInput"

messageType="taskservice:initiateTaskMessage"/>
```

```
                        <variable
name="initiateTaskResponseMessage"

messageType="taskservice:initiateTaskResponseMes
sage"/>
                    </variables>
                    <sequence>
                       <assign
name="AlertManulaInterventionToVerifier_1_Assig
nTaskAttributes">
                          <copy>
                             <from
expression="number(3)"/>
                             <to
variable="initiateTaskInput"
                                part="payload"

query="/taskservice:initiateTask/task:task/task:priorit
y"/>
                          </copy>
                          <copy>
                             <from>
                                <payload
xmlns="http://xmlns.oracle.com/bpel/workflow/task"
/>
                             </from>
                             <to
variable="initiateTaskInput"
                                part="payload"

query="/taskservice:initiateTask/task:task/task:payloa
d"/>
                          </copy>
                       </assign>
                       <invoke
name="initiateTask_AlertManulaInterventionToVerif
ier_1"

partnerLink="AlertManulaInterventionToVerifier.Ta
skService_1"

portType="taskservice:TaskService"
                          operation="initiateTask"

inputVariable="initiateTaskInput"

outputVariable="initiateTaskResponseMessage"/>
                       <receive
name="receiveCompletedTask_AlertManulaIntervent
ionToVerifier_1"

partnerLink="AlertManulaInterventionToVerifier.Ta
skService_1"

portType="taskservice:TaskServiceCallback"
operation="onTaskCompleted"

variable="AlertManulaInterventionToVerifier_1_glo
balVariable"
                          createInstance="no"/>
                    </sequence>
                 </scope>
                 <switch
name="humanIntervention">
                    <case
condition="bpws:getVariableData('AlertManulaInter
ventionToVerifier_1_globalVariable', 'payload',
'/task:task/task:systemAttributes/task:state') =
'COMPLETED' and
bpws:getVariableData('AlertManulaInterventionToV
erifier_1_globalVariable', 'payload',
'/task:task/task:systemAttributes/task:outcome') =
'APPROVE'">
                       <bpelx:annotation>
                          <bpelx:pattern>Task
outcome is APPROVE</bpelx:pattern>
                          <bpelx:general>
                             <bpelx:property
name="userLabel">Task

outcome
                                is

APPROVE</bpelx:property>
                          </bpelx:general>
                       </bpelx:annotation>
                       <sequence>
                          <assign/>
                          <invoke
name="alertManager"

partnerLink="AlertManagerService"/>
                       </sequence>
```

The high level design of verifier service is shown in figure 5.

In this service, the incoming pay load signature is checked with the pre-populated database from external and publicly known signatures and other IDS instance detected signatures. This will be achieved by using signatureVerification() method .

The updater Service listens for updates on a daily basis from the Master DB, which is connected on the cloud and sends web-service based publish notices to all instances. The updater then picks up these XMLs and their packet payloads and digests them using fast and compressive hashing algorithms

that compact this information and store it in the local signature DB as signatures.

So by using this service, one more challenge of IDS can be conquered such as frequent signature updates taking up a lot of time and skilled engineering resources, delivering a very high total cost of ownership.

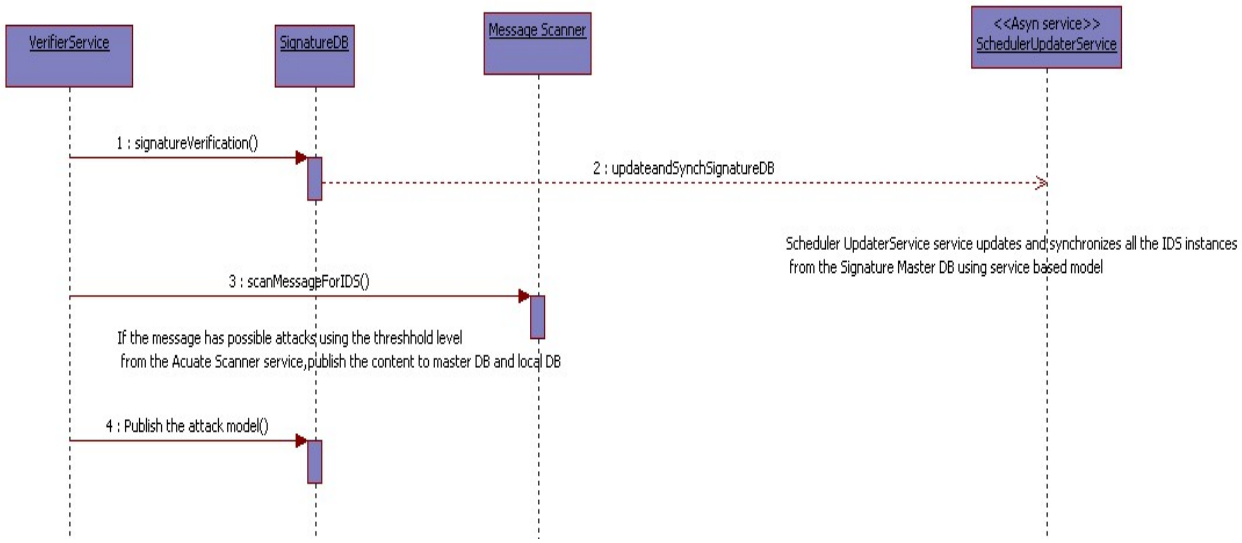

Figure 5: Verifier Service

Another functionality of this service is scanning messages for possible attacks. As already mentioned the verifier checks the hash of the XML and the payload from the DB against the incoming XML and payload of the sample window, which enables to detect a match for a possible harmful packet. For packets that have matched a possible known attack, the packets and the payload can be sent into the heuristic and acute scanner that can perform further analysis to detect newer form of attacks or decisively declare a packet/source as safe. This can over a period of time detect new attacks and recover from false alarms. Thus one more challenge of IDS such as lacking in both accuracy and specificity and generate too many false alarms is conquered.

Thus if a payload was marked as possibly harmful, a fuzzy logic AI agent running in the heuristic scanner can verify the safety or the hostility of the payload to a pre-determined degree of threshold (say 25% to 80%) before declaring and publishing it to other instances through the Master DB and also updating the local DB.

The Orchestration logic for Heuristics scanner and digester is presented below

```
 <sequence name="Sequence_5">
          <invoke name="updateDigester"
partnerLink="DigesterService"/>
        </sequence>
        <sequence name="Sequence_5">
          <sequence>
            <invoke
name="HeuristicAcuteScanner"/>
            <scope name="Quearantine_1"

xmlns="http://schemas.xmlsoap.org/ws/2003/03/busi
ness-process/"

xmlns:wf="http://schemas.oracle.com/bpel/extension/
workflow"

wf:key="Quearantine_1_globalVariable">
              <bpelx:annotation
xmlns:bpelx="http://schemas.oracle.com/bpel/extensi
on">
                <bpelx:pattern
patternName="bpelx:workflow"></bpelx:pattern>
              </bpelx:annotation>
              <variables>
                <variable name="initiateTaskInput"

messageType="taskservice:initiateTaskMessage"/>
                <variable
name="initiateTaskResponseMessage"
```

```
messageType="taskservice:initiateTaskResponseMes
sage"/>
                </variables>
                <sequence>
                   <assign
name="Quearantine_1_AssignTaskAttributes">
                      <copy>
                         <from
expression="number(3)"/>
                         <to
variable="initiateTaskInput"
                            part="payload"

query="/taskservice:initiateTask/task:task/task:priorit
y"/>
                      </copy>
                      <copy>
                         <from>
                            <payload
xmlns="http://xmlns.oracle.com/bpel/workflow/task"
/>
                         </from>
                         <to
variable="initiateTaskInput"
                            part="payload"

query="/taskservice:initiateTask/task:task/task:payloa
d"/>
                      </copy>
                   </assign>
                   <invoke
name="initiateTask_Quearantine_1"

partnerLink="Quearantine.TaskService_1"

portType="taskservice:TaskService"
                      operation="initiateTask"

inputVariable="initiateTaskInput"

outputVariable="initiateTaskResponseMessage"/>
                   <receive
name="receiveCompletedTask_Quearantine_1"

partnerLink="Quearantine.TaskService_1"

portType="taskservice:TaskServiceCallback"

operation="onTaskCompleted"

variable="Quearantine_1_globalVariable"
                      createInstance="no"/>
                </sequence>
             </scope>
             <switch name="taskSwitch">
                <case
condition="bpws:getVariableData('Quearantine_1_gl
obalVariable', 'payload',
'/task:task/task:systemAttributes/task:state') =
'COMPLETED' and
bpws:getVariableData('Quearantine_1_globalVariabl
e', 'payload',
'/task:task/task:systemAttributes/task:outcome') =
'REJECT'">
                   <bpelx:annotation>
                      <bpelx:pattern>Task outcome is
REJECT</bpelx:pattern>
                      <bpelx:general>
                         <bpelx:property
name="userLabel">Task
                                                outcome
is

REJECT</bpelx:property>
                      </bpelx:general>
                   </bpelx:annotation>
                   <sequence>
                      <assign/>
                      <invoke

partnerLink="DigesterService"

name="quarantineAndUpdateDigester"/>
                   </sequence>
                </case>
                <case
condition="bpws:getVariableData('Quearantine_1_gl
obalVariable', 'payload',
'/task:task/task:systemAttributes/task:state') =
'COMPLETED' and
bpws:getVariableData('Quearantine_1_globalVariabl
e', 'payload',
'/task:task/task:systemAttributes/task:outcome') =
'APPROVE'">
                   <bpelx:annotation>
                      <bpelx:pattern>Task outcome is
APPROVE</bpelx:pattern>
                      <bpelx:general>
                         <bpelx:property
name="userLabel">Task
outcome is
APPROVE</bpelx:property>
                      </bpelx:general>
                   </bpelx:annotation>
                   <sequence>
                      <assign/>
                   </sequence>
```

All the period, the Master Database is kept updated through SOA components about attacks detected or

false alarms nullified at the distributed locations. The Master Database on the next day updates all IDS instances local databases. So this makes the IDS as a true IDPS (Intrusion Detection and prevention system) because if the attack is detected at one location, the attack model is published to all other locations in the enterprise through a service.

Hence this proposed model has conquered one more challenge of IDS such as Current IDS are concentrating on detection, but not on prevention

6.1 Sub Processes Identification

As specified in [8], the business process model generated is the key artifact in SOAD process and serves as input to four sub processes. There are four sub processes.

1) Activity Services
2) Business Process services
3) Client Services
4) Data Services

To develop any of the 4 services discussed above, a series of generic steps.

- Service Identification.
- Analysis and design.
- Technology selection.
- Coding and Testing.
- Integrating services.

In the context of our IDS model, we have identified the 4 services as follows.

Activity Services: These are the applications that support our IDS tool activities. These applications not only support current state objectives but also meet the future state objectives. For example Different Algorithms (Pattern matching, Genetic, Intelligent), log generators, graph generators are identified as Activity services.

Business Process Services: Business Process model workflows are expressed as BPEL, configured and orchestrated to generate business process services. In our IDS model Business Rules, Heuristic & acute scanner, Updater & Digester are some of the Business Process services.

Client Services: These services facilitate the delivery of content through various channels such as web, mobile etc. For our IDS model a User Interface to customize business rules, displaying existing rules, capturing values in the form fields, conversion policy of converting a string into numerical value may be some of the examples of client services.

Data Services: These services define the way to store and access core data of the enterprise. For our IDS model adding new signatures to database, converting data to normalized structure and ensuring that database is maintained in clustered structure, access to master database are few examples of data services.

The SOAD process diagram along with four services are shown in figure 6.

For any IDS, several algorithms such as pattern matching algorithms, Dos detection algorithms etc are necessary for defense against severe threats. Activity services will maintain such algorithms and retrieve appropriate algorithm during the flow of events. Also they maintain log generators which will record all the activities of an attack. Graph generators will use this data and generate attack graphs. On the other hand, the client services will deal with presentation of an user interface for entry, display of rules and any conversion policies. The conversion policies are necessary as the host and network packet orders are different. The data services such as data normalization, cluster maintenance and interactions between local and master databases will aid us in maintaining the database in normalized form.

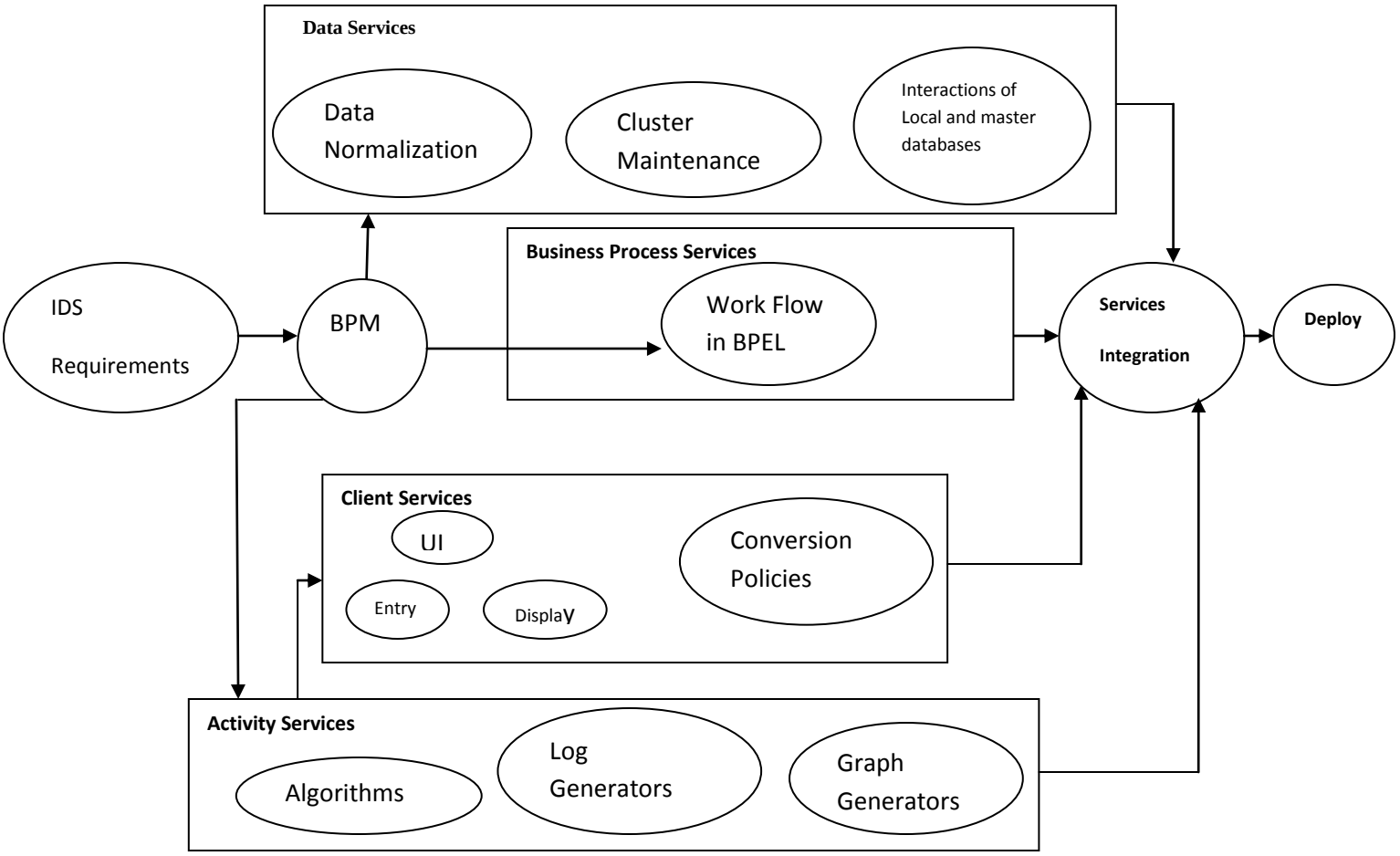

Figure 6: Sub-Processes Interaction

This will ensure that a rule will be present in the database only once. Also for efficient rule processing, the data is maintained in structured form such that related keywords are available in a single cluster. The important task of data service is to maintain the interactions with local database and master database. Synchronizing the data from several databases will be critical. The local database should update itself a new signature and inform the same to master database. The master database after a specified period of time, should update the local databases of other domains with all the new signatures. All these interactions between client, activity and data services will be synchronized by Business process services. The architecture is highly scalable and greatly interoperable.

7. RELATED WORK

In [12], based on danger theory, authors have proposed a four layer model of Immune based intrusion detection system. The first layer is danger sense layer which handles alert correlation problems, and to construct an intrusion scenario that would be detected by reacting to the balance of various types of alerts. The second layer is danger computation layer which calculates computes danger according to the intrusion alert 5-Tuple.The third layer is immune response layer which will detect the abnormal behavior. Fourth layer is spot disposal layer which will remove the dangerous behaviors. In [13], authors have proposed a Intrusion detection Intelligent agent system where several intelligent agents are integrated for providing in depth defense strategy against intrusions.

The main goals of this approach are its distributed architecture, scalability, efficiency and the use of intelligent agents. In [14], In order to enhance the availability and practicality of intelligent intrusion detection system based on machine learning in high-speed network, an improved fast inductive learning method for intrusion detection (FILMID) is designed and implemented. Accordingly, an efficient intrusion detection model based on FILMID algorithm is presented. In [15], a design scheme of intrusion detection system based on pattern matching algorithm is proposed. Also authors aimed at several key modules of intrusion detection system, a detailed analysis of data acquisition module, protocol processing module, feature matching module, log record module and intrusion response module is also given in this paper. Data acquisition module is responsible for capturing various types of hardware frames from network flow and handing these hardware frames to data pretreatment module and then the data pretreatment module strips off hardware frame heads and checks the integrity of messages. Based on application protocols hardware frames are sent to response protocol analyzing and processing modules respectively. For example, TELNET protocol has a process of packet. The pattern matching algorithm will judge for intrusion. If intrusion is found, alarm is given by intrusion response module and attack log is recorded. If no intrusion, it will make a detailed record of protocol operation log.

Conclusion

The proposed architecture and its design can manage the distributed system components efficiently. It allows new computing resources and services to be added dynamically. Most of the challenges faced by current IDS are addressed by the proposed architecture. Our future work aims at developing algorithms that would allow global distribution of various processing components.

References

[1] Rebecca Bace and Peter Mell "Intrusion Detection systems" NIST Special Publication on Intrusion Detection Systems
[2] McAfee network protection solutions "Next generation intrusion detection systems
[3]IBM Red book "Patterns: SOA Foundation Service Creation Scenario"
[4] Ali Arsanjani "How to identify, specify, and realize services for your SOA "
[5] Jim Amsden "Service realization"
[6] http://www-01.ibm.com/software/solutions/soa/
[7] Evdemon, J, 2005, "The Four Tenets of Service Orientation"http://www.bpminstitute.org/articles/article/article/the-four-tenets-of-service-orientation.html
[8]Shankar k," Service oriented analysis and design process for the enterprise"**,** 7th WSEAS International Conference on applied computer science, Venice, Italy, November 21-23, 2007,Pgs 366-371 , ISBN ~ ISSN:1750-5117 , 978-960-6766-18-3,ACM
[9] Humphrey, Watts. "A Discipline for Software Engineering. Reading", MA: Addison-Wesley Publishing Company, Inc., 1995.
[10]Zimmermann, O.; Krogdahl,P.; Gee, C., 2004, "Elements of Service-Oriented Analysis and Design".http://www.ibm.com/developerworks/webservices/library/ws-soad1/
[11]Kambhampaty, S,chandra, S. "Service Oriented Architecture for Enterprise Applications", WSEAS Transactions on Business and Economics. Issue 3, Volume 2, July 2005.
[12]Haidong Fu, Xiguo Yuan, Liping Hu," Design of a Four-layer Model Based on Danger Theory and AIS for IDS"2007,IEEE.
[13] Amine Berqia and Gustavo Nacsimento," A Distributed Approach For Intrusion Detection Systems"2004, IEEE**.**
[14] Wu Yang, Wei Wan , Lin Guo, Le-Jun Zhang," An Efficient Intrusion Detection Model Based On Fast Inductive Learning "Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007,IEEE.
[15] Zhang Hu," Design of Intrusion Detection System Based on a New Pattern Matching Algorithm" International Conference on Computer Engineering and Technology 2009 IEEE.

Author Biographies

K.V.S.N.Rama Rao was born in Andhra Pradesh, India and has attained his Masters in Computer Applications. from Andhra University and currently pursuing PhD in Berhampur University. His research areas are network security and Intrusion detection systems.

Pandu Prudvi has around 15 years of experience in software development field and currently working as SOA architect in Mahindra Satyam, Hyderabad. Has rich expertise in service oriented architecture and web services.

Manas Ranjan Patra was born in Orissa and has attained his Doctorate from Hyderabad Central University and currently working in Berhampur University. His research areas are network security, Intrusion detection systems and intelligent systems.