CHAPTER 3
DESIGN METHODOLOGY

3.1 Introduction

This chapter discusses the design methodologies used for applying them to real time phishing detection scenarios. In the section 3.2 the architecture of the Phishing Detection System (PDS) is discussed, then the activity diagram depicting the process flow during the various phases of the PDS is discussed in the section 3.3. The proposed PDS algorithm is discussed in section 3.4.

3.2 System Architecture

The architecture of the PDS Monitoring system consists of the following sub-systems:

- Instant Messaging system with clients and web browser.
- PDS monitoring system applying OBIE and CBA.
- Database (Message DB, Filter word DB, Ontology DB, Phishing rules DB and Phish word DB)

The Figure 3.1 shows architecture of the PDS monitoring system in Instant Messaging System (IMS), with the interrelationship among various subsystems in order to detect the phishing words based on context of the instant message and reports it to the victim client.
Figure 3.1 Architecture of the PDS monitoring system in IMS.

The architectural sub-systems of the PDS Monitoring system as discussed in the following sub sections:

3.2.1 Instant Messaging System (IMS) with clients and web browser.

The IMS contains the following components:

- Various client machines with browsers.
- Web server containing the Instant Messaging Resources.
- IM Multiplexer for multiplexing the communications.
• Directory Server which helps in authenticating a client and contains the identities of the clients.

• Messaging server where the offline line messages are forwarded whenever the chatter starts the next session using SMTP.

3.2.2 PDS monitoring system applying OBIE and CBA.

The PDS monitoring system performs its task in two phases:

• Data preprocessing to extract domain and context using ontology (OBIE).

• Applying CBA and generate useful association rules

In the first phase the PDS monitoring system captures and stores the instant messages in the Message DB (MDB) and then performs preprocessing on it through OBIE to identify the Domain and Context of the probable phishing word.

Data preprocessing to extract domain and context using ontology (OBIE):

The preprocessing includes removal of stop words and stemming.

Stop Words Removal: From each instant message, stop words i.e. words that are not significant are removed, i.e. prepositions, conjunctions, articles, adjectives, adverbs, etc. [36]. Stop word examples are: from, into, in, for, while, a, an, the, that, these, those, under, over, about, although, how, what, when, who, whom, etc. Indexing cannot be used on stop words and the removal of these words improves the efficiency of retrieval.
**Stemming** is performed to reduce the derived or inflected words to their stem i.e. to its basic or root form. It is also referred to as conflation. Programs which perform stemming are called as stemmers or stemming algorithms [35]. Also stemming derives the stem morphologically from a completely suffixed word [34]. A stemmer identify the string "fishing", "fisher", "fishery", “fished” etc. based on the root "fish".

The extracted keywords after preprocessing i.e. removing stop words and stemming are stored in the Filter Word DB (FWDB).

**Domain identification:** After stemming is performed all the keywords obtained are in machine understandable form, which are to be mapped with the ontology to determine their respective domains through the OBIE. For the domain identification the first step is to use the Triplet Extraction Algorithm to extract the subject, predicate and object from the keywords. The NLP approach used here identifies a subject and maps it to a semantic class. It uses the predicate and object as name and value of the attribute respectively [12].

The next step is to identify the theme concept for which the concepts are to be extracted first. Concepts are the extracted tokens which are tagged as nouns and form the Concept set. The subjects identified above are populated into the subjectList. The concept which occurs the most number of times forms the MaxOccurConcepts. Identifying the Theme concept is done by performing the intersection of the three sets obtained [12].

Mathematically, Concept = [nouns]
subjectList = [subjects]

MaxOccurConcepts = [concepts]

ThemeConcept = concept ∩ subjectList ∩ MaxOccurConcepts [12]

Consider the following text passage:

“Hotel Taj Banjara at Banjara Hills offers excellent facilities and accommodation. Comprising of 4 blocks and 68 deluxe rooms, Taj Banjara offers a pleasant stay.”

Concepts = \{Hotel, Taj Banjara, Banjara Hills, facilities, accommodation, blocks, rooms, stay\} Mathematically, Concept = \{ nouns \}

subjectList = \{Taj Banjara\} subjectList = \{subjects\}

MaxOccurConcepts = \{Taj Banjara (2)\}

ThemeConcept = \{Taj Banjara\}

Themeconcept = Concept ∩ subjectList ∩ MaxOccurConcepts

Taj Banjara is identified as the theme concept

Now the Domain class can be identified by the mapping to the string Theme concept by using one of the two applicable rules: Explicit mention rule and the Implicit rule. Explicit mention rule suggests that the string which are the class names themselves. As far as the Implicit rule is concerned, it is used when there is no string match suggesting that the domain ontology lexicon. Domain ontologies are formed by experienced experts of the relevant
domain. Semantic lexicons are created related to the respective domains, which is then used to identify the domain which the message content refers [12].

In the example the string Taj Banjara refers explicitly to hotel, and thus Taj Banjara belongs to hotel instance and the domain identified is hotel.

**Context identification:** This is an enhancement to the concepts discussed from [12] where in the task was limited to domain identification. The main contribution in this work is to identify the context or intention of the instant messages exchanged between chatters. The phishing detection may be misleading if phishing alerts are generated for harmless words, which are unintentional. This may leave a bad impression on the chatter to ignore such alerts again and again. Consider for example a message “Joe is so fond of chocolates. He would kill anybody for a bar of chocolate.” Any phishing detection system would raise an alert for the word “kill”, when actually its use is harmless. Extracting the true context is the main objective of this work apart from getting the least values for the false positives and false negatives.

In continuation to the previous discussion a new *predicateList* is introduced.

Concepts = {Joe, chocolate, bar}  
*Mathematically, Concept = {nouns}*

subjectList = {chocolate}  
*subjectList = {subjects}*

predicateList = {fond, kill}

MaxOccurConcepts = {chocolate (2)}
ThemeConcept = {chocolate}

Themeconcept = Concept ∩ subjectList ∩ MaxOccurConcepts

“chocolate” is identified as the theme concept

It is found that domain for the theme concept “chocolate” does not exist in the domain ontology lexicon and comes under the Implicit rule. Thus a new Domain ontology for “eatables” is formed by experienced experts of the relevant domain using the classes, attributes and relation present in the ontology. New semantic lexicons (only a few a listed) have been created related to the “eatables” domain.

Eatables domain lexicon = {eat, dark chocolate, bar, nutty, lick, munch, kill, fond}

Note that, the words “kill” and “fond” are also included in the semantic lexicon of the domain “eatables”, as they were a part of the predicateList. The new domain is mapped with the existing domains in the Pre-defined phishing rules DB (PRDB).

If a match occurs Context = {harmful} or else Context = {harmless}.

In the above example context = “harmless” in spite of the predicate or the verb being “kill”.

The implementation of the context is possible with very simple sentences containing single values for the triplets.

The filtered keyword dataset in the FWDB, along with attribute values of
Domain ontology and the Context forms the Test dataset for the generation of classification association rules (CAR) using CBA.

**Applying CBA and generate useful association rules:**

The Classifier Based Association (CBA) method explores all the associations between attribute values and their classes in the training dataset in order to build large classifiers. The training data set used here is the pre-defined phishing rules table shown in Table 3.1.

<table>
<thead>
<tr>
<th>S.no.</th>
<th>Keyword</th>
<th>Domain</th>
<th>Ontology context</th>
<th>Threshold value</th>
<th>Phishing word</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Account no</td>
<td>Financial gain</td>
<td>Harmful</td>
<td>3</td>
<td>YES</td>
</tr>
<tr>
<td>2</td>
<td>All caps</td>
<td>Account creation tips</td>
<td>Harmful</td>
<td>5</td>
<td>YES</td>
</tr>
<tr>
<td>3</td>
<td>Pet name</td>
<td>Deceitful elicitation</td>
<td>Harmless</td>
<td>5</td>
<td>YES</td>
</tr>
<tr>
<td>4</td>
<td>credit card detail</td>
<td>Financial gain</td>
<td>Harmful</td>
<td>2</td>
<td>YES</td>
</tr>
<tr>
<td>5</td>
<td>Kids name</td>
<td>Identity access</td>
<td>Harmless</td>
<td>1</td>
<td>NO</td>
</tr>
<tr>
<td>6</td>
<td>DOB</td>
<td>Financial gain</td>
<td>Harmless</td>
<td>3</td>
<td>YES</td>
</tr>
<tr>
<td>7</td>
<td>Name</td>
<td>Not defined</td>
<td>Harmless</td>
<td>5</td>
<td>NO</td>
</tr>
<tr>
<td>8</td>
<td>Whats App</td>
<td>Not defined</td>
<td>Harmless</td>
<td>3</td>
<td>NO</td>
</tr>
<tr>
<td>9</td>
<td>Password</td>
<td>Fame and notoriety</td>
<td>Harmful</td>
<td>1</td>
<td>YES</td>
</tr>
<tr>
<td>10</td>
<td>Hack</td>
<td>Not defined</td>
<td>Harmless</td>
<td>1</td>
<td>NO</td>
</tr>
<tr>
<td>11</td>
<td>Supari</td>
<td>Not defined</td>
<td>Harmless</td>
<td>1</td>
<td>NO</td>
</tr>
<tr>
<td>12</td>
<td>Spl char</td>
<td>Account creation tips</td>
<td>Harmful</td>
<td>1</td>
<td>YES</td>
</tr>
<tr>
<td>13</td>
<td>Image</td>
<td>Not defined</td>
<td>Harmless</td>
<td>5</td>
<td>NO</td>
</tr>
</tbody>
</table>

Table 3.1 Training data for the classifier based association rules (snapshot)
The pre-defined phishing rules are framed by experts of the relevant domain having tremendous experience.

The training dataset \textit{(snapshot)} for the Classification Based Association rules is shown including the threshold values, Domain Ontology and its context. With the testing dataset rules are generated to predict the Phishing words and subsequently raising the phishing alert. If a new phishing word is identified based on threshold values based on frequently occurring words during chatting or a new phishing domain is discovered, these phishing words are appended to the Phishing Word DB (PWDB).

\textbf{3.2.3 Databases}

The backend used in the PDS monitoring system is Oracle 11g. There are total five database tables:

Message DB

Filter Word DB

Ontology DB

Phishing Rules DB

Phishing Words DB
### 3.3 System Workflow

The system workflow of the PDS monitoring algorithm initiates the steps to capture the phishing words from instant messages that are exchanged between the chatters through pre-defined phishing rules of Table 3.1 through Classification based on Association Rules (CBA) and Domain Ontology. The schematic illustration of the system workflow of the PDS monitoring algorithm is shown in figure 3.2.

![Activity diagram – System workflow of PDS Monitoring system.](image)

**Figure 3.2.** Activity diagram – System workflow of PDS Monitoring system.
3.4 PDS Monitoring algorithm

*Input*: Text messages of chatters

*Output*: Phish word Alert to Victim

*Steps:*

*Start*

1. Store Messages in *MDB*

2. Apply OBIE //Remove stop words, stemming, extract triplets

3. Get Domain ontology, Get Context

4. If found newDomain then step 5 else step 6

5. Append semantic lexicon (ODB) //through RDF

6. Store filtered words in *FWDB* //These words along with domain and context attributes will be the Test data set. Update threshold value

7. Generate CAR rules //for the test data set

8. Apply CBA //on the training data set pre-defined

(PrDB) and Test dataset to create classifier for generating phishing rules.

9. Generate Phishing rules

   if phish word = “YES” then step 10, step 11

   ELSE step 12
10. Append phish word to \((PWDB)\)

11. Raise ALERT // to Victim Client

12. Display message // original Message from \((MDB)\).

Stop.

The PDS monitoring algorithm elaborates the schematic work flow shown in the Figure 3.2 in a step by step manner.

This algorithm stores the instant messages in the MDB in step 1. Preprocessing is done on the messages through OBIE by removing stop words, performing stemming and extracting triplets (subject, predicate and object) in order to identify the Domain and Context in steps 2 and 3.

If a new domain is detected it is appended to the Ontology database \((ODB)\) i.e. semantic lexicon, through RDF in step 5, else store the keywords in filter word DB \((FWDB)\) in step 6.

In step 7, generation of Classification Association Rules \((CAR)\) is done through test data formed from keywords in FWDB and the attributes \((Domain, Context)\) obtained from step 3.

Apply Classification Based Association on the Training dataset i.e. the pre-defined phishing rule DB \((PRDB)\) and Test dataset to create classifier for generating phishing rules in step 8.

In steps 9 to 12 the generation of Phishing rules is done and if phish word is found then the phish word is appended to the phishing word DB \((PWDB)\)
and an alert is raised on the victim’s user interface, otherwise the message is displayed in its original form from the message database (MDB) on the Instant message interface.

3.5 The OBIE Architecture

Information extraction from the instant messages is required to be converted to machine understandable form in order to determine the domain and context of the instant message using Ontology Based Information Extraction (OBIE) component of the PDS monitoring system. The architecture for OBIE system is given below:

![Ontology Based Information Extraction Architecture](image)

**Figure 3.3: Ontology Based Information Extraction Architecture**

This OBIE module initially reads the instant message keywords stored in the filter word DB (FWDB). The concept of Semantic Lexicon is used to identify the semantic domain and context for the keywords being
processed through the domain inference module. The triplet algorithms are
used here to assist in the identification of the Theme concept and the
domain in the inference module [12].

The instance information is extracted by the instance extractor module. An
RDF node is created and the ontology is updated using the Jena Apache
APIs. Existing ontology editors like Protégé can be used for editing the
ontology. The Lexicon extractor module contains rules in order to learn
new lexicon symbols from the filtered keywords, and append them into
the semantic lexicon using a set of heuristics to identify the relationship
between lexical items and the existing semantic lexicon [12].

![Flowchart of OBIE domain and concept extraction workflow](image)

**Figure 3.4 OBIE domain and concept extraction workflow**

The figure 3.4 shows the workflow for the extraction of ontology instances.
3.6 Triplet algorithms

The Triplet Extraction Algorithms are used to extract the subject, predicate and object from the keywords for domain identification as discussed in section 3.2.2. The NLP approach used here identifies a subject and maps it to a semantic class. It uses the predicate and object as name and value of the attribute respectively [12].

The implementation of the triplet extraction algorithm mentioned in [37] has been done using the StanfordCoreNLP [38] Java library. The algorithms are briefly summarized for extraction of the subject, predicate and object.

**Algorithm 1**

*ExtractSubject*(string)

1. Perform a Breadth First Search (BFS) of the parse tree obtained by using StanfordCoreNLP library [38].

2. The NP subtree contains the subject, and it is the first Noun in the tree when traversed using BFS.

For complex noun compounds, the parse tree is used to extract all embedded Noun phrases (NP)[12] [40].

**Algorithm 2**

*ExtractPredicate*(string)

1. Perform a Depth First Search (DFS) of the VP subtree. The verb that
is deepest in the tree is the predicate.

The parse tree is used to extract all embedded Verb phrases (VP) [12] [39] [40].

Algorithm 3

ExtractObject (string)

1. Perform a search of the PP, ADJP subtree, and extract the first noun in the tree which is the object.

The parse tree is used to extract all embedded preposition phrases (PP), and adjective phrases (ADJP) [12] [40]

For the previously used sentence “Novotel is located in Hyderabad”, after applying the Triplet algorithm, the identified subject is Novotel, predicate is located, and object is Hyderabad. The theme concept identified here is Novotel and domain class extracted will be “hotel”.

If the above sentence is expanded into a passage “Novotel, located at Banjara hills Road no. 1, offers memorable accommodation for the guests. It has 5 blocks and 67 luxury rooms. Novotel offers a decent ambience along with culinary specialties of the Deccan.”
3.7 **Ontology Attribute Value Extraction**

Attribute values are extracted for various attributes are part of the sample hotel class. Thus

Name = Novotel

Facilities = accommodation

rooms = 67

These values are added to the hotel instance node, and the instance is appended into the ontology [12].

**Attribute Value extraction methodology**

The name value pairs are extracted from text by using pattern matching techniques, where a pattern has a few terms containing a set of constraints. Only when there is a match of pattern, the rule is executed.

Some patterns are easily identifiable and are extracted. Only the occurrence of a string is matched. For instance, “on-call” means that it is available [12].

**Ontology Update**

The extracted values are converted to RDF format, for appending it to the hotel ontology. An open-source Jena tool [52] is used for updating the ontology [12].
The RDF representation of the extracted information given in the earlier paragraph is given below:

\[
<\text{Hotel rdf:ID="Novotel">}
\]

\[
<\text{Facilities rdf:datatype="http://www.w3.org/2001/XMLSchema#string" [49]}
\]

\[
>\text{accomodation</location>}
\]

\[
<\text{rooms rdf:datatype="http://www.w3.org/2001/XMLSchema#integer" [50]}
\]

\[
>67</serves>
\]

\[
</\text{Hotel}> [12]
\]

**SPARQL queries**

After appending the data about the various identified classes, a semantic store is updated. It may be recalled that the initially the human experts create the ontology which represents a real life requirement for a domain like hotel.

SPARQL queries [51] [52] have to be run by loading this ontology in Protégé. All the querying of various triples, and retrieving the results is done using SPARQL [12].
3.8 Experimental Evaluation of CBA

The experimental evaluation of the CBA model (classifier) is done by comparing it with different classifiers like Zero-R and J48 or C4.5. For this the Weka tool [87] is used to assess the performance of generating rules from datasets which are well known [85] [86] i.e. Iris, pima-diabetes, Glass and CPU. The Table 3.2 shows the characteristic features of the above datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of Attributes</th>
<th>Class</th>
<th>No. of Class Values</th>
<th>No. of Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>5</td>
<td>class</td>
<td>3</td>
<td>150</td>
</tr>
<tr>
<td>Prima-diabetes</td>
<td>9</td>
<td>class</td>
<td>2</td>
<td>768</td>
</tr>
<tr>
<td>Glass</td>
<td>10</td>
<td>Type</td>
<td>7</td>
<td>214</td>
</tr>
<tr>
<td>CPU</td>
<td>7</td>
<td>class</td>
<td>4</td>
<td>209</td>
</tr>
</tbody>
</table>

Table 3.2 Dataset features obtained through Weka

Here in the experimental study on the datasets, the main purpose is to generate rules as much as possible with \( \text{min\_sup} \) greater than or equal to 1% and \( \text{min\_conf} \) equal to 50%.

The performance of the CBA classifier has been compared to the Zero R and the J48 (C4.5) classifiers based on the well-known datasets in Table 3.2. For CBA empirical values were taken for Single rule and Weighted by confidence also. The percentage of split for the training dataset was 66% and the remaining 33% was used by the testing dataset. The following discussions have been done for the above mentioned datasets.
<table>
<thead>
<tr>
<th>Classifier</th>
<th>Prediction mode</th>
<th>Accuracy</th>
<th>No. of Rules generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA</td>
<td>Single Rule</td>
<td>72.54%</td>
<td>56</td>
</tr>
<tr>
<td>CBA</td>
<td>Weighted</td>
<td>71.96%</td>
<td>56</td>
</tr>
<tr>
<td>J48</td>
<td>-</td>
<td>68.24%</td>
<td>23</td>
</tr>
<tr>
<td>Zero R</td>
<td>-</td>
<td>59.75%</td>
<td>7</td>
</tr>
</tbody>
</table>

Min_supp = 1% & min_conf = 50%

**Table 3.3 Comparison of Classifier performance on Iris Dataset**

The Table 3.3 shows the comparison of the classifiers on the iris dataset. The CBA performs better than the J48 and Zero R. The best accuracy is produced by the CBA with single rule.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Prediction mode</th>
<th>Accuracy</th>
<th>No. of Rules generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA</td>
<td>Single Rule</td>
<td>81.23%</td>
<td>206</td>
</tr>
<tr>
<td>CBA</td>
<td>Weighted</td>
<td>80.79%</td>
<td>206</td>
</tr>
<tr>
<td>J48</td>
<td>-</td>
<td>80.94%</td>
<td>235</td>
</tr>
<tr>
<td>Zero R</td>
<td>-</td>
<td>59.75%</td>
<td>9</td>
</tr>
</tbody>
</table>

Min_supp = 1% & min_conf = 50%

**Table 3.4 Comparison of Classifier performance on prima-diabetes Dataset**

The Table 3.4 shows the comparison of the classifiers on the prima-diabetes dataset. The J48 performs better than the CBA weighted and Zero R. The best accuracy is produced by the CBA with single rule.
Table 3.5 Comparison of Classifier performance on glass Dataset

The Table 3.5 shows the comparison of the classifiers on the glass dataset. The CBA performs better than the J48 and Zero R, but J48 is very close in margin with the CBA. The best accuracy is shared by both the prediction modes of the CBA.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Prediction mode</th>
<th>Accuracy</th>
<th>No. of Rules generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA</td>
<td>Single Rule</td>
<td>97.51%</td>
<td>34</td>
</tr>
<tr>
<td>CBA</td>
<td>Weighted</td>
<td>97.51%</td>
<td>34</td>
</tr>
<tr>
<td>J48</td>
<td></td>
<td>96.46%</td>
<td>29</td>
</tr>
<tr>
<td>Zero R</td>
<td></td>
<td>53.54%</td>
<td>3</td>
</tr>
</tbody>
</table>

Min_supp = 1% & min_conf = 50%

Table 3.6 Comparison of Classifier performance on CPU Dataset

The Table 3.6 shows the comparison of the classifiers on the CPU dataset. The CBA performs consistently better than the J48 and Zero R. The best accuracy is shared by both the prediction modes of the CBA only.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Prediction mode</th>
<th>Accuracy</th>
<th>No. of Rules generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA</td>
<td>Single Rule</td>
<td>68.22%</td>
<td>105</td>
</tr>
<tr>
<td>CBA</td>
<td>Weighted</td>
<td>68.22%</td>
<td>105</td>
</tr>
<tr>
<td>J48</td>
<td></td>
<td>66.74%</td>
<td>142</td>
</tr>
<tr>
<td>Zero R</td>
<td></td>
<td>55.17%</td>
<td>2</td>
</tr>
</tbody>
</table>

Min_supp = 1% & min_conf = 50%

In all the above comparisons, it has been quite evident that the CBA
classifier has shown consistency in its performance with regard to accuracy and the number of rules generated.

As discussed in Section 2.2, the Associative Classification (AC) uses CBA, CMAR, and CPAR to generate classifier rules, but the according to [16], the Classification based on Association (CBA) is more accurate than the C4.5 on large number of datasets. The iterative approach of Apriori is used by the CBA to create a classifier [16]. This suggests that the Apriori algorithm is a subset of the CBA, and the latter has a better performance. Hence, CBA has been used to generate phishing rules due to its better comparative results shown in this section.