1. INTRODUCTION

DATA breaches are one of the most devastating cyber incidents. The Privacy Rights Clearinghouse reports 7,730 data breaches between 2005 and 2017, accounting for 9,919,228,821 breached records. The Identity Theft Resource Center and Cyber Scout reports 1,093 data breach incidents in 2016, which is 40% higher than the 780 data breach incidents in 2015. The United States Office of Personnel Management (OPM) reports that the personnel information of 4.2 million current and former Federal government employees and the background investigation records of current, former, and prospective federal employees and contractors (including 21.5 million Social Security Numbers) were stolen in 2015. The monetary price incurred by data breaches is also substantial. IBM reports that in year 2016, the global average cost for each lost or stolen record containing sensitive or confidential information was $158. Net Diligence reports that in year 2016, the median number of breached records was 1,339, the median per-record cost was $39.82, the average breach cost was $665,000, and the median breach cost was $60,000.

While technological solutions can harden cyber systems against attacks, data breaches continue to be a big problem. This motivates us to characterize the evolution of data breach incidents. This not only will deep our understanding of data breaches, but also shed light on other approaches for mitigating the damage, such as insurance. Many believe that insurance will be useful, but the development of accurate cyber risk metrics to guide the assignment of insurance rates is beyond the reach of the current understanding of data breaches (e.g., the lack of modeling approaches)

Recently, researchers started modeling data breach incidents. Mallart and Serinette studied the statistical properties of the personal identity losses in the United States between year 2000 and 2008. They found that the number of breach incidents dramatically increases from 2000 to July 2006 but remains stable thereafter. Edwards et al. analyzed a dataset containing 2,253 breach incidents that span over a decade (2005 to 2015). They found that neither the size nor the frequency of data breaches has increased over the years. Wheatley et al. analyzed a dataset that is combined from and corresponds to organizational breach incidents between year 2000 and 2015. They found that the frequency of large breach incidents (i.e., the ones that breach more than 50,000 records) occurring to US firms is independent of time, but the frequency of large breach incidents occurring to non-US firms exhibits an increasing trend

The present study is motivated by several questions that have not been investigated until now, such as: Are data breaches caused by cyber-attacks increasing, decreasing, or stabilizing? A principled answer to this question will give us a clear insight into the overall situation of cyber threats. This question was not answered by previous studies. Specifically, the dataset analyzed in only covered the time span from 2000 to 2008 and does not necessarily contain the breach incidents that are caused by cyber-attacks; the dataset analyzed in is more recent but contains two kinds of incidents: negligent breaches (i.e., incidents caused by lost, discarded, stolen devices and other reasons) and malicious breaching. Since negligent breaches represent more human errors than cyber-attacks, we do not consider them in the present study. Because the malicious breaches studied in contain four sub-categories: hacking (including malware), insider, payment card fraud, and unknown, this study will focus on the hacking sub-category

(called hacking breach dataset thereafter), while noting that the other three sub-categories are interesting on their own and should be analyzed separately

A. Our Contributions

In this paper, we make the following three contributions. First, we show that both the hacking breach incident interarrival times (reflecting incident frequency) and breach sizes should be modeled by stochastic processes, rather than by distributions. We find that a particular point process can adequately describe the evolution of the hacking breach incidents inter-arrival times and that a particular ARIMA model can adequately describe the evolution of the hacking breach sizes, where ARIMA is acronym for "Autoregressive Integrated Moving Average We show that this stochastic process model can predict the inter-arrival times and the breach sizes.

Second, we discover a positive dependence between the incidents inter-arrival times and the breach sizes, and show that this dependence can be adequately described by a particular copula. We also show that when predicting inter-arrival times and breach sizes, it is necessary to consider the dependence; otherwise, the prediction results are not accurate. Third, we conduct both qualitative and quantitative trend analyses of the cyber hacking breach incidents. We find that the situation is indeed getting worse in terms of the incidents inter-arrival time because hacking breach incidents become more and more frequent, but the situation is stabilizing in terms of the incident breach size, indicating that the damage of individual hacking breach incidents will not get much worse. We hope the present study will inspire more investigations, which can offer deep insights into alternate risk mitigation approaches. Such insights are useful to insurance companies, government agencies, and regulators because they need to deeply understand the nature of data breach risks

2. LITERATURE SURVEY

A) Prior Works Closely Related to the Present Study: Maillart and Sornette analyzed a dataset of 956 personal identity loss incidents that occurred in the United States between year 2000 and 2008. They found that the personal identity losses per incident, denoted by X, can be modeled by a heavy tail distribution $Pr(X > n) \sim n^{-\alpha}$ where $\alpha = 0.7 \pm 0.1$. This result remains valid when dividing the dataset per type of organizations: business, education, government, and medical institution. Because the probability density function of the identity losses per incident is static, the situation of identity loss is stable from the point of view of the breach size.

Edwards et al. analyzed a different breach dataset of 2,253 breach incidents that span over a decade (2005 to 2015). These breach incidents include two categories: negligent breaches (i.e., incidents caused by lost, discarded, stolen devices, or other reasons) and malicious breaching (i.e., incidents caused by hacking, insider and other reasons). They showed that the breach size can be modeled by the log-normal or log-skewnormal distribution and the breach frequency can be modeled by the negative binomial distribution implying that neither the breach size nor the breach frequency has increased over the years

Wheatley et al. analyzed an organizational breach incidents dataset that is combined from and spans over a decade (year 2000 to 2015). They used the Extreme Value Theory to study the maximum breach size, and further modeled the large breach sizes by a doubly truncated Pareto distribution. They also used linear regression to study the frequency of the data breaches and found that the frequency of large breaching incidents is independent of time for the United States organizations but shows an increasing trend for non-US organizations.

There are also studies on the dependence among cyber risks. Böhme and Kataria studied the dependence between cyber risks of two levels: within a company (internal dependence) and across companies (global dependence). Herath and Herath used the Archimedean copula to model cyber risks caused by virus incidents and found that there exists some dependence between these risks. Mukhopadhyay et al used a copula-based Bayesian Belief Network to assess cyber vulnerability. Xu and Hua investigated using copulas to model dependent cyber risks. Xu et al. used copulas to investigate the dependence encountered when modeling the effectiveness of cyber defense early-warning. Peng et al. investigated multivariate cybersecurity risks with dependence. Compared with all these studies mentioned above, the present paper is unique in that it uses a new methodology to analyze a new perspective of breach incidents (i.e., cyber hacking breach incidents). This perspective is important because it reflects the consequence of cyber hacking (including malware). The new methodology found for the first time, that both the incidents inter-arrival times and the breach sizes should be modeled by stochastic processes rather than distributions, and that there exists a positive dependence between them

B) Other Prior Works Related to the Present Study: Eling and Loperfido analyzed a dataset from the point of view of actuarial modeling and pricing. Bagchi and Udo used a variant of the Gompertz model to analyze the growth of computer and Internet-related crimes. Condon et. al used the ARIMA model to predict security incidents based on a dataset provided by the Office of Information Technology

at the University of Maryland. Zhan et al. analyzed the posture of cyber threats by using a dataset collected at a network telescope. Using datasets collected at a honeypot, Zhan et al., exploited their statistical properties including long-range dependence and extreme values to describe and predict the number of attacks against the honeypot; a predictability evaluation of a related dataset is described in. Peng et al. used a marked point process to predict extreme attack rates. Bakdash et al. extended these studies into related cybersecurity scenarios. Liu et al. investigated how to use externally observable features of a network (e.g., mismanagement symptoms) to forecast the potential of data breach incidents to that network. Sen and Borle studied the factors that could increase or decrease the contextual risk of data breaches, by using tools that include the opportunity theory of crime, the institutional anomie theory, and the institutional theory

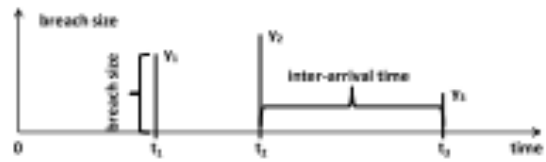


Fig. 1. Illustrative description of cyber hacking breach incidents.

2.1 Description of Cyber hacking issues

C) Paper Outline The rest of the paper is organized as follows. In Section II we describe the dataset and research questions. In Section III we present a basic analysis of the dataset. In Section IV we develop a novel point process model for analyzing the dataset. In Section V, we discuss the prediction performance of the proposed model. In Section VI we present qualitative and quantitative trend analyses. In Section VII we conclude our paper with future research directions. We defer formal description of the main statistical notions to the Appendix, and discuss their intuitive meanings when they are mentioned for the first time

3. SYSTEM ANALYSIS

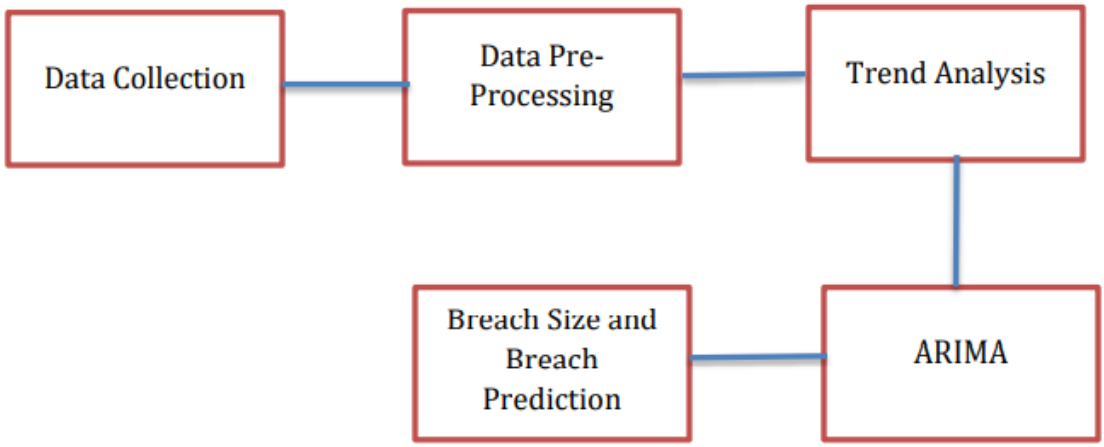
3.1 Existing System

Previously, the dataset analyzed for only covered the time span from 2000 to 2008 and does not necessarily contain the breach incidents that are caused by cyber-attacks and the another dataset analyzed in is more recent, but contains two kinds of incidents: negligent breaches (i.e., incidents caused by lost, discarded, stolen devices and other reasons) and malicious breaching. Since negligent breaches represent more human errors than cyber-attacks, we do not consider them. Because the malicious breaches studied contain four sub- categories: hacking (including malware), insider, payment card fraud, and unknown Recently, researchers started modeling data breach incidents and studied the statistical properties of the personal identity losses in the States between year 2000 and 2008. They found that the number of breach incidents dramatically increased from 2000 to July 2006 but remained stable thereafter and then analyzed a dataset containing 2,253 breach incidents that span over a decade They found that neither the size nor the frequency of data breaches has increased over the years and also analyzed a dataset that is combined from corresponds to organizational breach incidents for years They found that the frequency of large breach incidents is more occurring to US firms is independent of time, but the frequency of large breach incidents occurring to non-US firms exhibits an increasing trend.

3.2 Proposed System

In this project initially started with the both the hacking breach incident interarrival times (reflecting incident frequency) and breach sizes should be modeled by stochastic processes, rather than by distributions. We find that a particular point process can adequately describe the evolution of the hacking breach incidents inter-arrival times and that a particular ARIMA (Time Series forecasting) model can adequately describe the evolution of the hacking breach sizes, where ARIMA is acronym for "Autoregressive and Moving Average". We show that this stochastic process model can predict the inter-arrival times and the breach sizes and then we discover a positive dependence between the incidents inter-arrival times and the breach sizes. We also show that when predicting inter-arrival times and breach sizes, it is necessary to consider the dependence; otherwise, the prediction results and also we both qualitative and quantitative trend analyses of the cyber hacking breach incidents are done and find that the situation is indeed getting worse in terms of the incidents inter-arrival time because hacking breach incidents become more and more frequent, but the situation is stabilizing in terms of the incident breach size, indicating that the damage of individual hacking breach incidents will not get much worse.

Block Diagram:



3.2 Block Diagram

SYSTEM CONFIGURATION:

3.3 Modules Description

1. UPLOAD DATA
2. ACCESS DETAILS
3. USER PERMISSIONS
4. DATA ANALYSIS

3.4 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

3.5. Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased

3.6 Operational Feasibility

Are you into the production of "things"? Perhaps, your answer would be yes. We naturally don't call them things; instead, we call them products, services, or systems. Using the term "things" sounds foreign because you can't just drop them into an area without touching them. They need to be connected to an existing service or business. These "things" are an extension of the organization where they are produced.

3.7 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system

3.8 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

4. System Requirements Specification

4.1 Introduction

The project involved analyzing the design of few applications to make the application more users friendly. To do so, it was important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. To make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers

4.2 Purpose

Typing the user needs to do. To make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

4.3 Functional Requirements

For developing the application, the following are the Software Requirements:

1. Python
2. anaconda

Operating Systems supported

1. Windows 7
2. Windows XP
3. Windows 8

Technologies and Languages used to Develop

1. Python

Debugger and Emulator

- Any Browser (Particularly Chrome)

Hardware Requirements

For developing the application, the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB

Space on Hard Disk: minimum 512MB

4.4 Non-Functional Requirements

- Any Browser (Particularly Chrome)

4.5 Input & Output Design

4.5.1 Input design:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.

Methods for preparing input validations and steps to follow when error occur

OBJECTIVES:

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

4.5.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.

Confirm an action

4.6 Hardware Requirements

For developing the application the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB

Space on Hard Disk: minimum 512MB

4.7 Software Requirements

For developing the application the following are the Software Requirements:

3. Python
4. anaconda

Operating Systems supported

4. Windows 7
5. Windows XP
6. Windows 8

Technologies and Languages used to Develop

Python

5.System Design

5.1. System Design

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system designis decided. During this phase, the details of the data of a module are usually specified in a high-level design description language, which is independent of the target language in whichthe software will eventually be implemented.

In system design the focus is on identifying the modules, where as during detailed designthe focus is on designing the logic for each of the modules. In other works, in system designthe attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearlyspecified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software

5.2 UML Diagrams (9 types)

Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagrams are not a new concept but it is used widely in different forms in different industries.

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system. You can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way. There are two broad categories of diagrams and they are again divided into

subcategories –

Structural Diagrams

Behavioral Diagrams

Structural Diagrams

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable.

These static parts are represented by classes, interfaces, objects, components, and nodes. The four structural diagrams are −

Class diagram

Object diagram

Component diagram

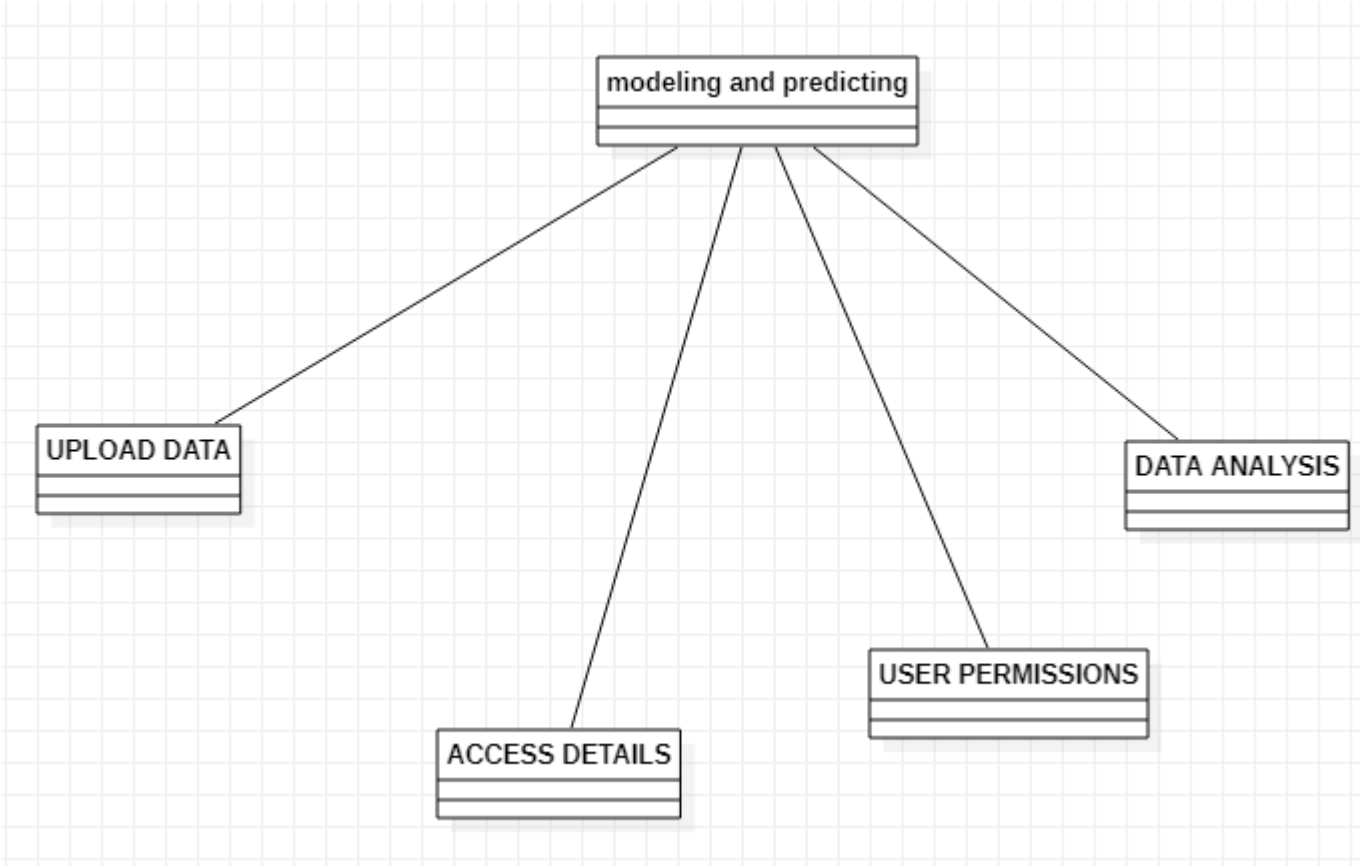
Deployment diagram

Class Diagram

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. Active class is used in a class diagram to represent the

concurrency of the system.

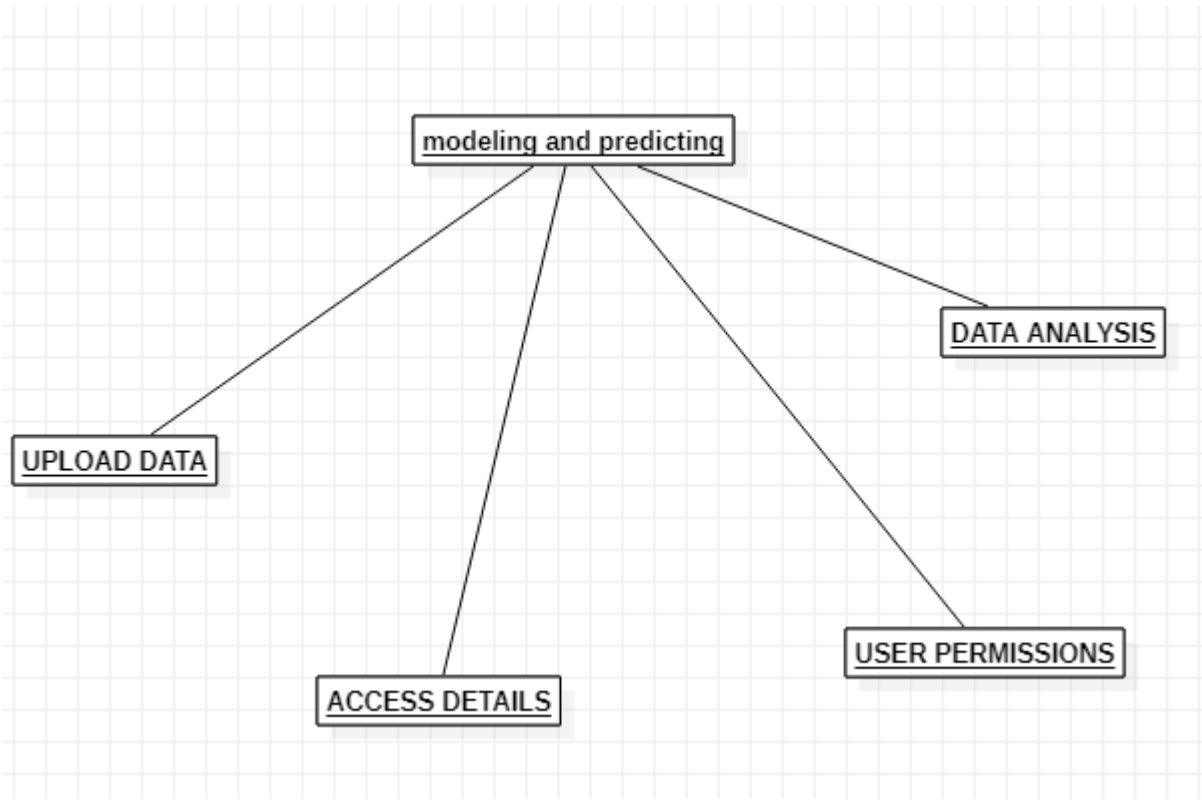
Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose. is is the most widely used diagram at the time of system construction.



Object Diagram

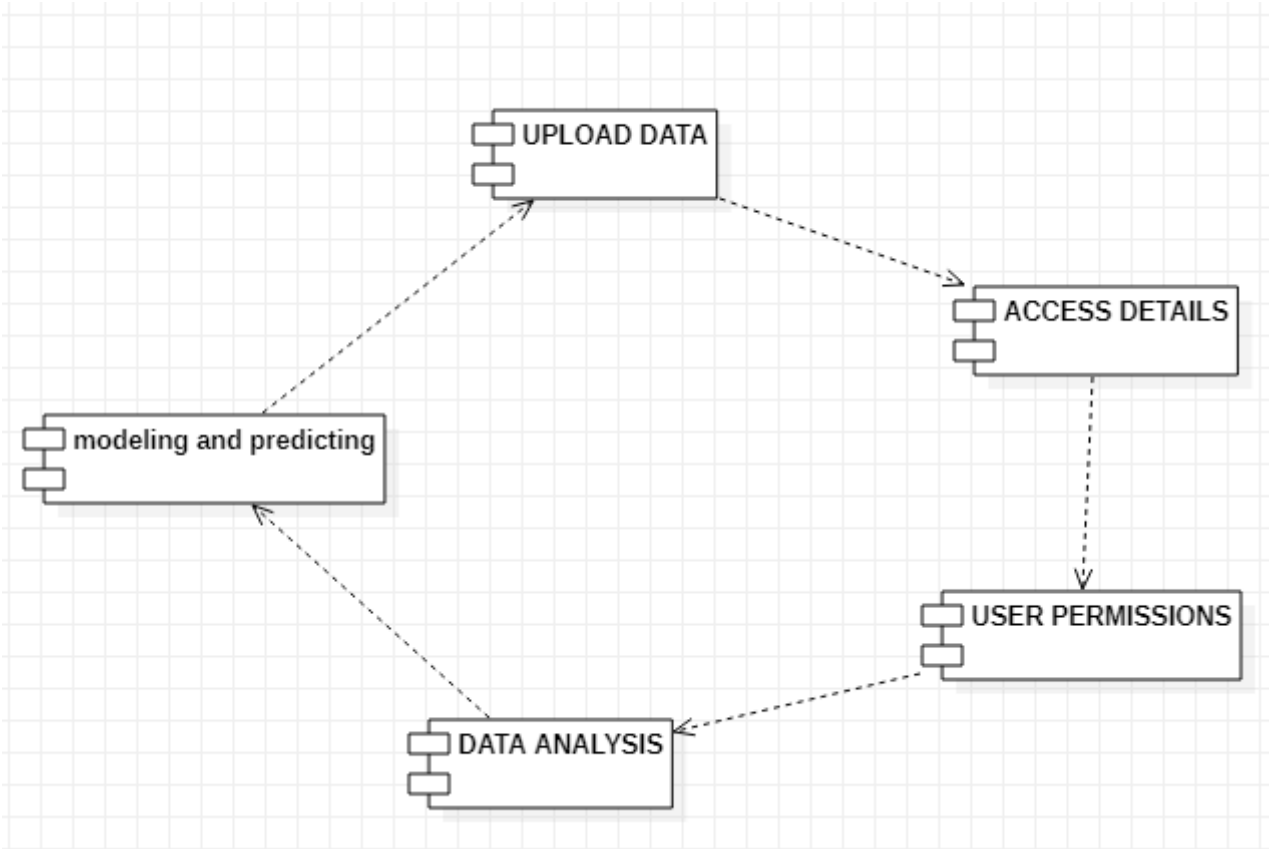
Object diagrams can be described as an instance of class diagram. Thus, these diagrams are more close to real-life scenarios where we implement a system. Object diagrams are a set of objects and their relationship is just like class diagrams. They also represent the static view of the system.

The usage of object diagrams is similar to class diagrams but they are used to build prototype of a system from a
practical perspective.



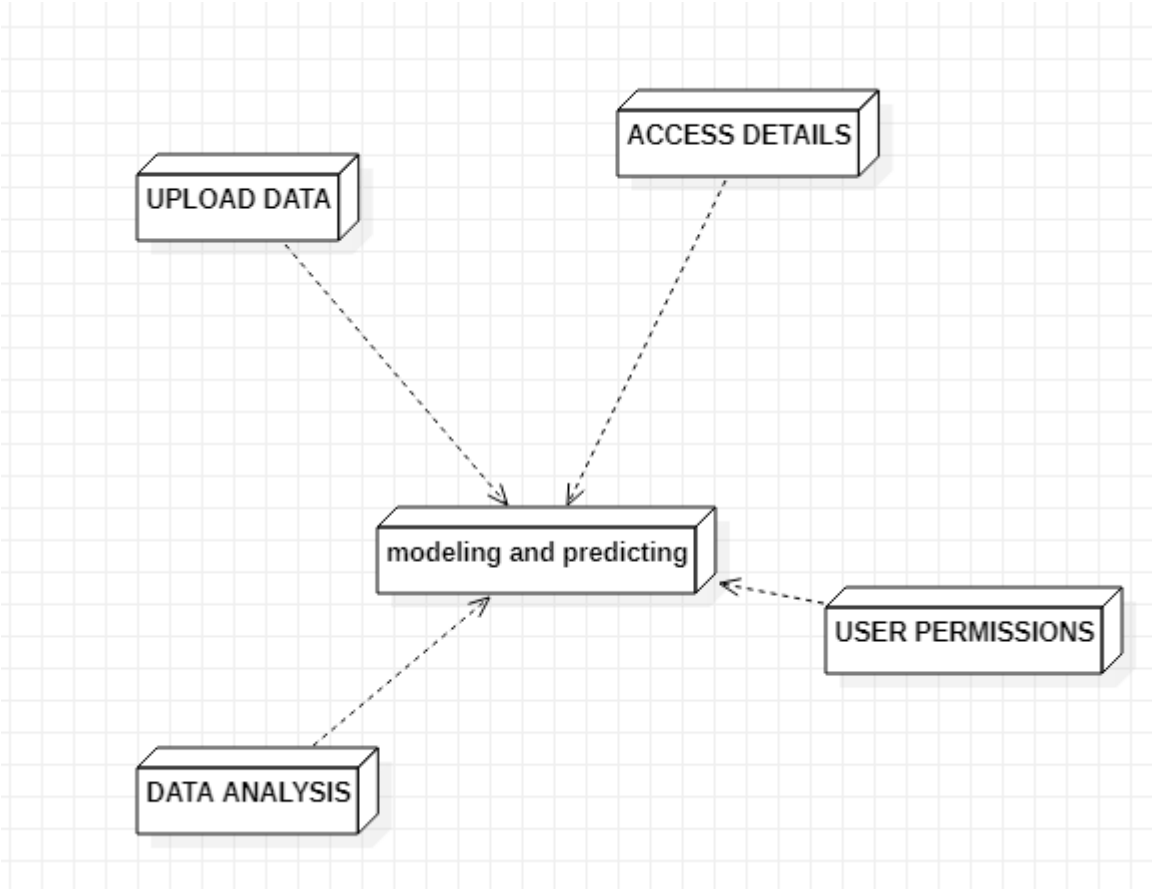
Component Diagram

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system. During the design phase, software artifacts (classes, interfaces, etc.) of a system are arranged in different groups depending upon their relationship. Now, these groups are known as components. Finally, it can be said component diagrams are used to visualize the implementation.



Deployment Diagram

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team. Note − If the above descriptions and usages are observed carefully then it is very clear that all the diagrams have some relationship with one another. Component diagrams are dependent upon the classes, interfaces, etc. which are part of class/object diagram. Again, the deployment diagram is dependent upon the components, which are used to make component diagrams.



Behavioral Diagrams

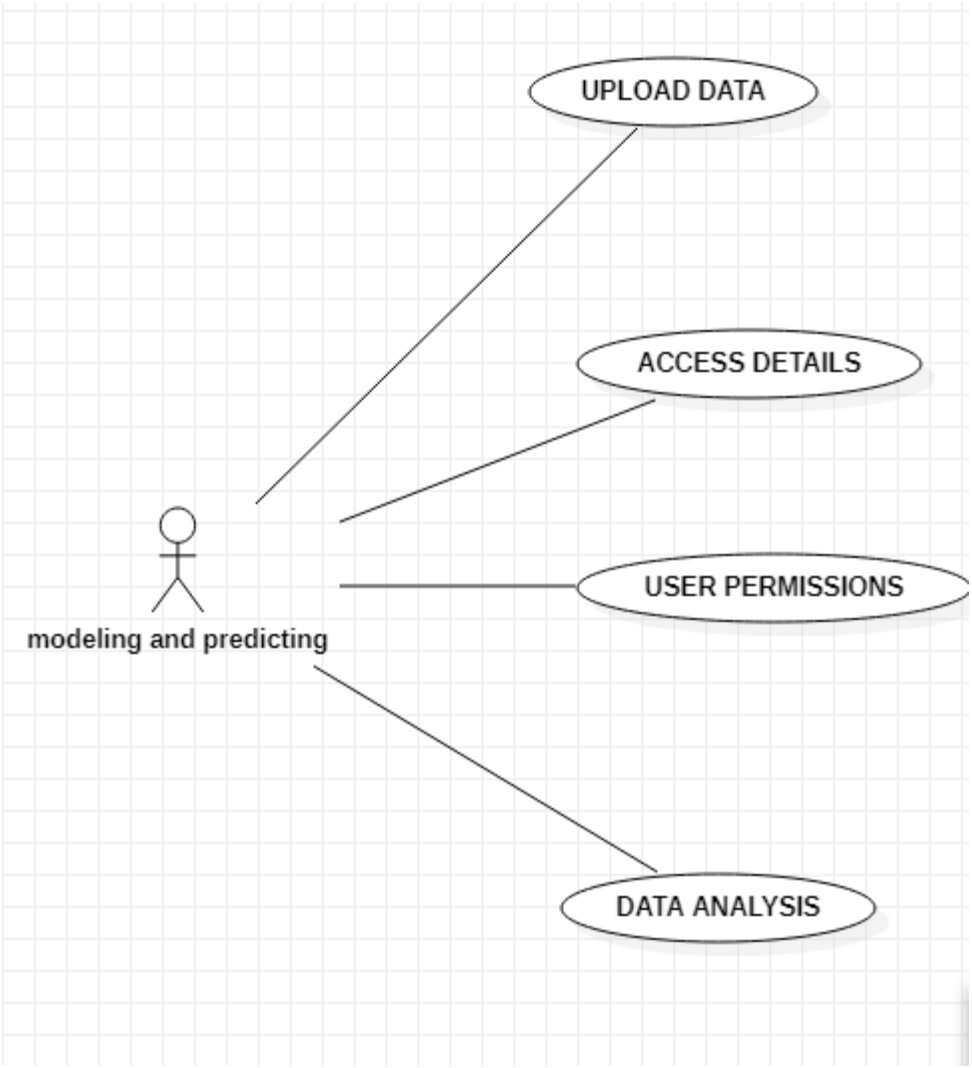
Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered.

Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

UML has the following five types of behavioral diagrams −
Use case diagram
Sequence diagram
Collaboration diagram
Statechart diagram
Activity diagram

Use Case Diagram

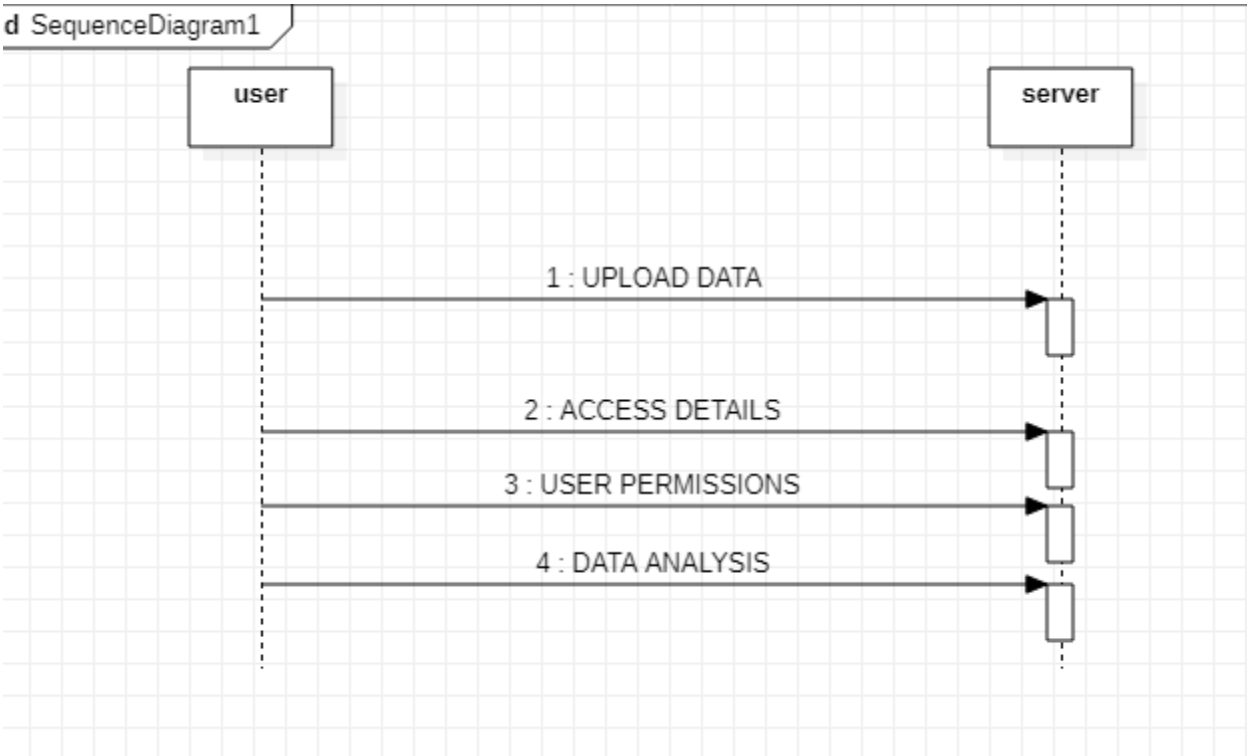
Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.



Sequence Diagram

A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.

Interaction among the components of a system is very important from implementation and execution perspective. Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

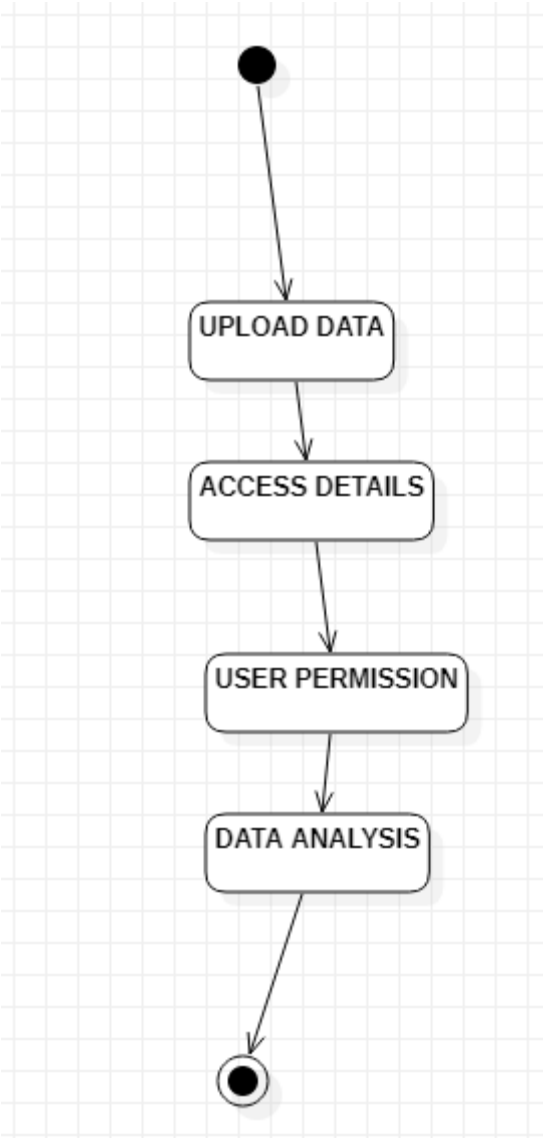


Collaboration Diagram

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links. The purpose of collaboration diagram is similar to sequence diagram. However, the specific purpose of collaboration diagram is to visualize the organization of objects and their interaction.

Statechart Diagram

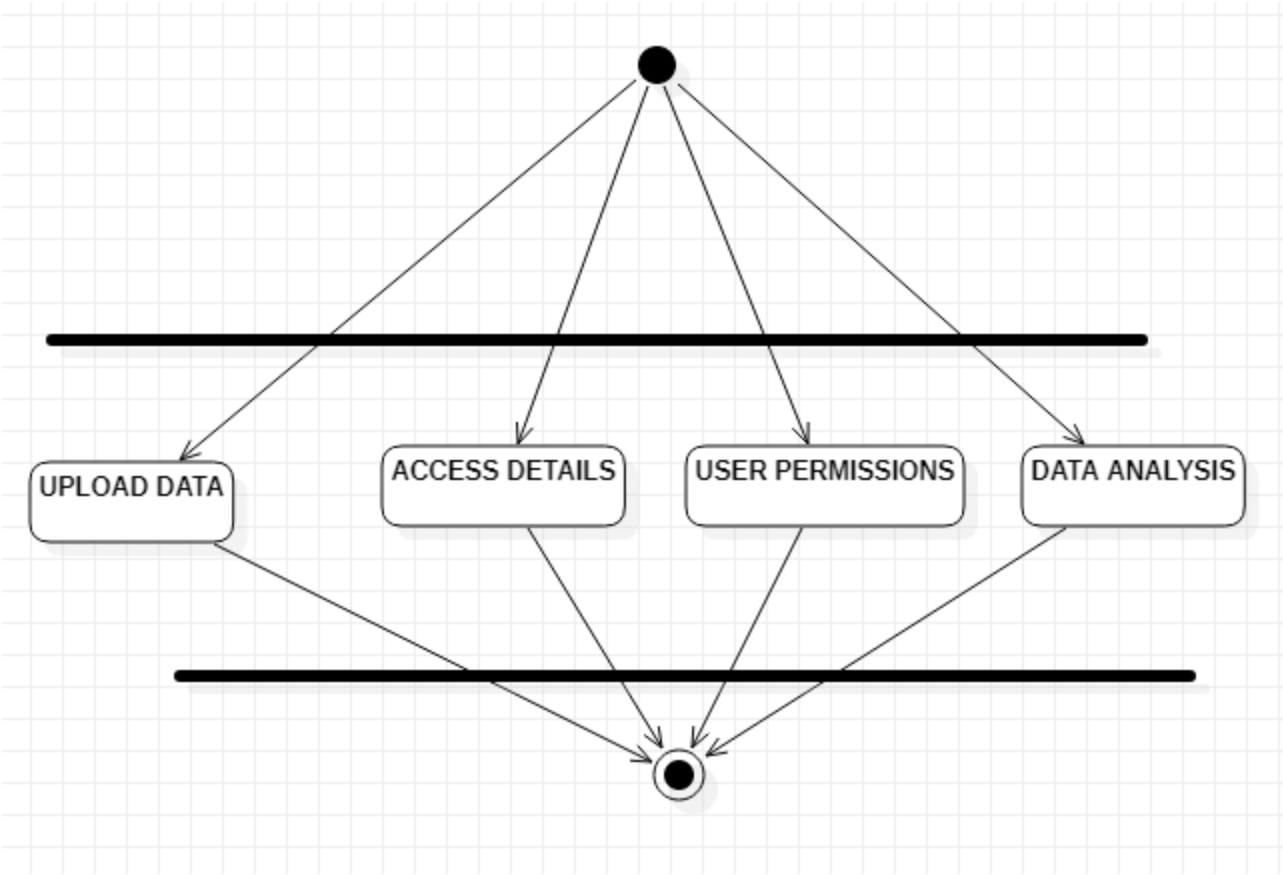
Any real-time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. Statechart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface, etc. State chart diagram is used to visualize the reaction of a system by internal/external factors.



Activity Diagram

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system. Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

Note − Dynamic nature of a system is very difficult to capture. UML has provided features to capture the dynamics of a system from different angles. Sequence diagrams and collaboration diagrams are isomorphic, hence they can be converted from one another without losing any information. This is also true for Statechart and activity diagram

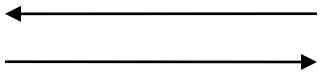


5.3 data flow diagrams

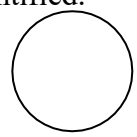
A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are asfollows:

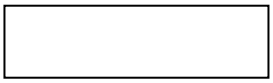
1. Dataflow: Data move in a specific direction from an origin to a destination.



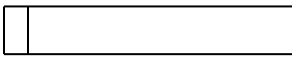
2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.



3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.



4. Data Store: Here data are stored or referenced by a process in the System.



6. Implementation

Modules Description

1. UPLOAD DATA
2. ACCESS DETAILS
3. USER PERMISSIONS
4. DATA ANALYSIS

7. TECHNOLOGY DESCRIPTION

PYTHON:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python

print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable

$./test.py
```

This produces the following result −

```
Hello, Python!
```

8. CODING

Modelling and Predicting Cyber hacking Breaches using Time Series Analysis

Importing required libraries

In [1]:
```
import warnings
import itertools
import numpy as np
import matplotlib.pyplot as plt

warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')

import pandas as pd
import statsmodels.api as sm
import matplotlib

matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
```

In [2]:
```
warnings.filterwarnings(action='once')
```

Loading Data

In [3]:
```
data=pd.read_csv("cyber_breach_data.csv")
```

```
D:\Anaconda\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `s
hould_run_async` will not call `transform_cell` automatically in the future. Pl
ease pass the result to `transformed_cell` argument and any exception that happ
en during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

In [4]:
```
data.head()
```

Out[4]:

	Date Made Public	Company	City	State	Type of breach	Type of organization	Total Records	Description of incident	Informatio Sourc
0	1-10-2005	George Mason University	Fairfax	Virginia	HACK	EDU	32,000	Names, photos, and Social Security numbers of ...	Datalos DI
1	1-18-2005	University of California, San Diego	San Diego	California	HACK	EDU	3,500	A hacker breached the security of two Universi...	Datalos DI
2	1-22-2005	University of Northern Colorado	Greeley	Colorado	PORT	EDU	15,790	A hard drive was lost or stolen. It contained ...	Datalos DI
3	2-12-2005	Science Applications International Corp. (SAIC)	San Diego	California	STAT	BSO	45,000	On January 25 thieves broke \n into...	Datalos DI
4	2-15-2005	ChoicePoint	Alpharetta	Georgia	INSD	BSO	1,63,000	Fraudsters who presented themselves as legitim...	Securit Breac Lette

Preprocessing the Data

In [5]:
```
time_series=data.loc[data['Type of breach'] == "HACK", ["Date Made Public","Total
time_series = time_series.dropna(subset=["Date Made Public"])
time_series["Total Records"]=time_series["Total Records"].str.replace(",","")
time_series["Total Records"] = pd.to_numeric(time_series["Total Records"], errors
drop_indices=time_series["Total Records"][time_series["Total Records"]>30000].ind
drop_index=time_series["Total Records"][time_series["Total Records"]==0].index
time_series=time_series.drop(drop_indices)
time_series=time_series.drop(drop_index)
time_series.index=range(1406)
```

```
D:\Anaconda\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `s
hould_run_async` will not call `transform_cell` automatically in the future. Pl
ease pass the result to `transformed_cell` argument and any exception that happ
en during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

In [6]:
```
time_series = time_series.groupby('Date Made Public')['Total Records'].sum().rese
time_series = time_series.set_index('Date Made Public')
time_series.index = pd.to_datetime(time_series.index)
time_series.index
```

```
D:\Anaconda\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `s
hould_run_async` will not call `transform_cell` automatically in the future. Pl
ease pass the result to `transformed_cell` argument and any exception that happ
en during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

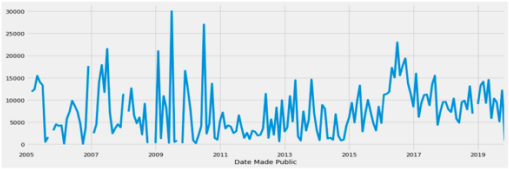
Out[6]:
```
DatetimeIndex(['2019-08-11', '2019-08-15', '2019-08-16', '2019-08-22',
               '2019-08-23', '2019-08-27', '2019-08-29', '2019-08-30',
               '2019-09-03', '2019-09-06',
               ...
               '2006-09-07', '2011-09-07', '2017-09-07', '2018-09-07',
               '2010-09-08', '2014-09-08', '2016-09-08', '2017-09-08',
               '2014-09-09', '2015-09-09'],
              dtype='datetime64[ns]', name='Date Made Public', length=980, freq
=None)
```

Re-sampling the data

In [7]:
```
y = time_series['Total Records'].resample('MS').mean()
# time_series
```

Original Plot of No.of.Breaches

In [8]:
```
y.plot(figsize=(15, 6))
plt.show()
```



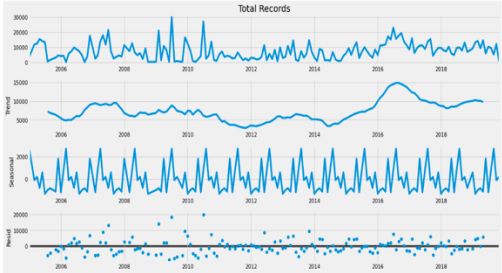
In [9]:
```
y=y.dropna()
```

```
D:\Anaconda\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `s
hould_run_async` will not call `transform_cell` automatically in the future. Pl
ease pass the result to `transformed_cell` argument and any exception that happ
en during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Using statistical methods to decompose data into seasonality and trend

In [10]:
```
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8

decomposition = sm.tsa.seasonal_decompose(y,freq=12, model='additive')
fig = decomposition.plot()
plt.show()
```

```
<ipython-input-10-b6bcd6af9ddc>:4: FutureWarning: the 'freq'' keyword is deprec
ated, use 'period' instead
  decomposition = sm.tsa.seasonal_decompose(y,freq=12, model='additive')
```



In [3]:
```
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)

D:\Anaconda\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `s
hould_run_async` will not call `transform_cell` automatically in the future. Pl
ease pass the result to `transformed_cell` argument and any exception that happ
en during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Generating all possible p,d,q values to fit into ARIMA model

Performing Grid search for optimal values

```
In [12]: for param in pdq:
             for param_seasonal in seasonal_pdq:
                 try:
                     mod = sm.tsa.statespace.SARIMAX(y,
                                                     order=param,
                                                     seasonal_order=param_seasonal,
                                                     enforce_stationarity=False,
                                                     enforce_invertibility=False)

                     results = mod.fit()
                     print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.a
                 except:
                     continue
```

```
D:\Anaconda\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarni
ng: A date index has been provided, but it has no associated frequency informat
ion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'

ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:3591.518657843917
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:3303.220306216598
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:3287.024369114365 7
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:2974.3106294140307
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:3277.994601451446
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:3217.5023939847206
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:3019.58811545881 8
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:2975.9956577541716
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:3516.590014501853
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:3262.0643440064614
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:3268.0079282329625
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:2963.6620910972224
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:3291.4889535702146
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:3231.011467011 71
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:3025.1105351637643
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:2965.404817271921
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:3478.5961941084315
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:3237.4834286182627
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:3369.6182618879704
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:3007.9438708903062
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:3257.6260551573673
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:3235.525530804661
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:3075.5032677162426
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:2995.4285984632625
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:3371.2342888743424
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:3132.589286135246
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:3252.5987733804177
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:2942.678982012003
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:3171.6002528007575
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:3132.2437566417348
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:2996.4144041400264
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:2944.20327513753
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:3474.756348365636
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:3237.4500873671495
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:3288.354290730482
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:2983.6061792202945
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:3237.44203491878 5
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:3212.55123563125
```

```
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:3002.2398983289746
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:2968.617156358664
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:3393.4499767361394
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:3153.537749953487
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:3267.3640483187837
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:2945.039654731687
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:3173.134309365685
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:3154.034101662 3024
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:2997.8711138271537

D:\Anaconda\lib\site-packages\statsmodels\base\model.py:566: ConvergenceWarnin
g: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "

ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:2913.3972357845228
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:3433.41047515983 7
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:3194.0201398716 3
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:3323.5697147998903
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:2991.392804140946 3
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:3193.879397589173
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:3189.026529234512
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:3024.3299778415985
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:2991.934279524743
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:3373.0518527163226
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:3134.538003995417
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:3254.4000128772295
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:2900.336909776821
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:3154.22220458144
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:3134.1257277450927
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:2992.1007716331956
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:2945.8414262897186
```

Modelling the most optimal paramters

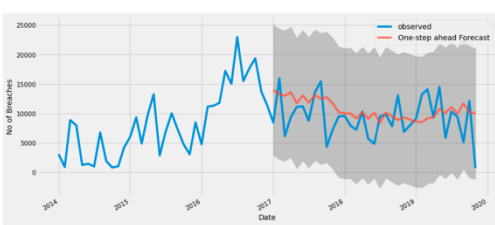
```
In [15]: mod = sm.tsa.statespace.SARIMAX(y,
                                         order=(1, 1, 1),
                                         seasonal_order=(0, 1, 1, 12),
                                         enforce_stationarity=False,
                                         enforce_invertibility=False)

         results = mod.fit()
```

```
D:\Anaconda\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `s
hould_run_async` will not call `transform_cell` automatically in the future. Pl
ease pass the result to `transformed_cell` argument and any exception that happ
en during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
D:\Anaconda\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: ValueWarni
ng: A date index has been provided, but it has no associated frequency informat
ion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
```

Plotting the forecast of ARIMA model and observed results

```
In [17]: pred = results.get_prediction(start=pd.to_datetime('2017-01-01'), dynamic=False)
         pred_ci = pred.conf_int()

         ax = y['2014':].plot(label='observed')
         pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsiz

         ax.fill_between(pred_ci.iloc[:, 0],
                         pred_ci.iloc[:, 1], color='k', alpha=.2)

         ax.set_xlabel('Date')
         ax.set_ylabel('No of Breaches')
         plt.legend()

         plt.show()
```

```
D:\Anaconda\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `s
hould_run_async` will not call `transform_cell` automatically in the future. Pl
ease pass the result to `transformed_cell` argument and any exception that happ
en during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```



Evaluating the model using metrics using MSE, RMSE, MAE

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ – *predicted value of y*

$\bar{y}$ – *mean value of y*

```
In [4]: y_forecasted = pred.predicted_mean
        y_truth = y['2017-01-01':]

        mse = ((y_forecasted - y_truth) ** 2).mean()
        print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-a69f324b5fbc> in <module>
----> 1 y_forecasted = pred.predicted_mean
      2 y_truth = y['2017-01-01':]
      3
      4 mse = ((y_forecasted - y_truth) ** 2).mean()
      5 print('The Mean Squared Error of our forecasts is {}'.format(round(mse,
2)))

NameError: name 'pred' is not defined
```

```
In [19]: print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(m
```

```
The Root Mean Squared Error of our forecasts is 3837.42
```

```
In [20]: mean_error=(y_forecasted-y_truth).sum()/len(y_truth)
         print('Mean Error:',mean_error)
```

```
Mean Error: 1394.8844736572496
```

9. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTS:

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually, and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered

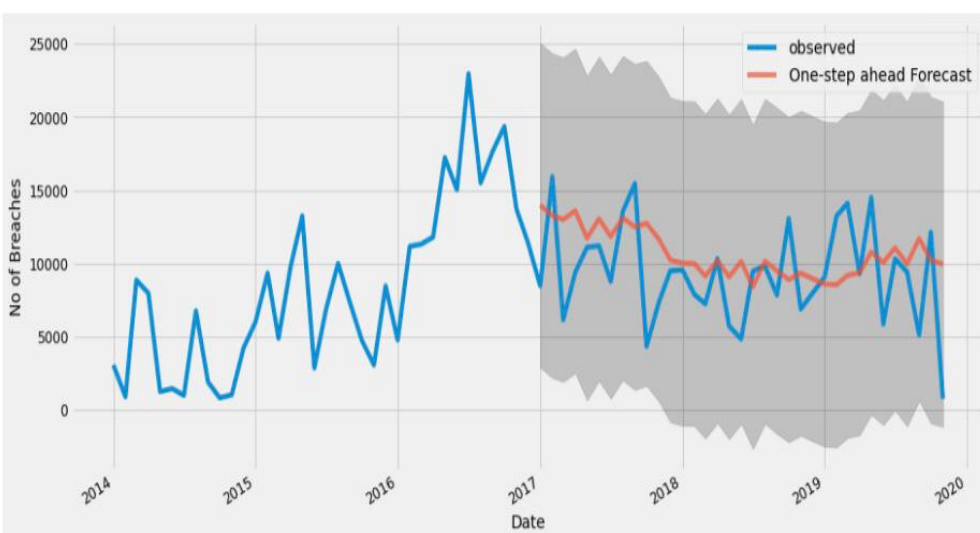
10. OUTPUT SCREENS (FORMS & REPORTS)

6/17/2021 Modelling_predicting_cyber_breach - Jupyter Notebook

In [4]:
```
data.head()
```

Out[4]:

	Date Made Public	Company	City	State	Type of breach	Type of organization	Total Records	Description of incident	Informatio Sourc
0	1-10-2005	George Mason University	Fairfax	Virginia	HACK	EDU	32,000	Names, photos, and Social Security numbers of ...	Datalos DI
1	1-18-2005	University of California, San Diego	San Diego	California	HACK	EDU	3,500	A hacker breached the security of two Universi...	Datalos DI
2	1-22-2005	University of Northern Colorado	Greeley	Colorado	PORT	EDU	15,790	A hard drive was lost or stolen. It contained ...	Datalos DI
3	2-12-2005	Science Applications International Corp. (SAIC)	San Diego	California	STAT	BSO	45,000	On January 25 thieves broke \n into...	Datalos DI
4	2-15-2005	ChoicePoint	Alpharetta	Georgia	INSD	BSO	1,63,000	Fraudsters who presented themselves as legitim...	Securit Breac Lette



11. FUTURE ENHANCEMENT

11. Future Enhancement

Modeling and predicting cyber hacking breaches is an important, yet challenging, problem. In this we initiate the study of modeling and predicting cyber hacking breaches.in the present study we proposed a stochastic process model to predict both hacking breach incident inter arrival times and breach sizes. Here we will use both qualitative and quantitative trend analysis on the data set.

12. CONCLUSION

We analyzed a hacking breach dataset from the incident inter arrival time and the breach size and showed that they both should be modeled by stochastic process rather than distribution because they exhibit autocorrelations. Here we are using a ARIMA to process for the evaluation of breaching incidents. we conducted both qualitative and quantitative analysis to draw further insights. The earlier the breach is detected, we will either stop any damage to the data or prevent damaging data

13. REFERENCES

[1] P. R. Clearinghouse. Privacy Rights Clearinghouse's Chronology of Data Breaches. Accessed: Nov. 2017. [Online]. Available: https://www.privacyrights.org/data-breaches.

[2] ITR Center. Data Breaches Increase 40 Percent in 2016, Finds New Report From fraud Resource Center and CyberScout. Accessed: Nov. 2017. [Online]. Available: http://www.idtheftcenter.org/2016databreaches.html.

[3] C. R. Center. Cybersecurity Incidents. Accessed: Nov. 2017. [Online]. Available: https://www.opm.gov/cybersecurity/cybersecurity-incidents.

[4] IBM Security. Accessed: Nov. 2017. [Online]. Available: https://www.ibm.com/security/data-breach/index.html.

[5]NetDiligence. The 2016 Cyber Claims Study. Accessed: Nov. 2017. [Online]. Available: https://netdiligence.com/wp-content/uploads/2016/ 10/P02_NetDiligence2016-Cyber-Claims-Study-ONLINE.pdf.

[6] M. Eling and W. Schnell, "What can we realize cyber risk and cyber risk insurance?" J. Risk Finance, vol. 17, no. 5, pp. 474–491, 2016.

[7] T. Maillart and D. Sornette, "Heavy-tailed distribution of cyber-risks," Eur. Phys. J. B, vol. 75, no. 3, pp. 357–364, 2010.

[8] R. B. Security. Datalossdb. Accessed: Nov. 2017. [Online]. Available: https://blog.datalossdb.org