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 andCyber 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. NetDiligence.

Manuscript received November 22, 2017; revised March 16, 2018 and April 23, 2018; accepted April 28, 2018. Date of publication May 16, 2018; date of current version May 23, 2018. This work was supported in part by ARL under Grant W911NF-17-2-0127. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mauro Conti. (Corresponding author: Shouhuai Xu.) M. Xu is with the Department of Mathematics, Illinois State University, Normal, IL 61761 USA. K. M. Schweitzer and R. M. Bateman are with the U.S. Army Research Laboratory South (Cyber), San Antonio, TX 78284 USA. S. Xu is with the Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: shxu@cs.utsa.edu).

Colour versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org, 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 characterise 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 modelling approaches). Recently, researchers started modelling data breach incidents. Maillart and Sornette 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. analysed 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. analysed a dataset that is combined from and and corresponds to organisational 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 stabilising? A principled answer to this question will gives a clear insight into the overall situation of cyber threats. This question was not answered by previous studies.

Specifically, the dataset analysed 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 analysed 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 analysed separately.

ABSTRACT

Analysing cyber incident data sets is an important method for deepening our understanding of the evolution of the threat situation. This is a relatively new research topic, and many studies remain to be done. In this paper, we report a statistical analysis of a breach incident data set corresponding to 12 years (2005–2017) of cyber hacking activities that include malware attacks. We show that, in contrast to the findings reported in the literature, both hacking breach incident *inter-arrival times* and *breach sizes* should be modelled by stochastic processes, rather Than by distributions because they exhibit autocorrelations. Then, we propose particular stochastic process models to, respectively, fit the inter-arrival times and the breach sizes. We also show that these models can predict the inter-arrival times and the breach sizes. In order to get deeper insights into the evolution of hacking breach incidents, we conduct both qualitative and quantitative trend analyses on the data set. We draw a set of cyber security insights, including that the threat of cyber hacks is indeed getting worse in terms of their frequency, but not in terms of the magnitude of their damage.

2. LITERATURE SURVEY

1.Prior Works Closely Related to the Present Study:

Maillart and Sornette analysed 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 modelled 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 organisation's: 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. analysed 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 modelled by the log-normal or log-skew normal distribution and the breach frequency can be modelled by the negative binomial distribution,

Wheatley et al. analysed an organisational breach incidents dataset that is combined from and and spans over a decade (year 2000 to 2015). They used the Extreme Value Theory to study the maximum breach size, and further modelled 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 organisations, but shows an increasing trend for non-US organisation's.

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 Archimedeancopula 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 modelling 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 analyse 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 modelled by stochastic processes rather than distributions, and that there exists a positive dependence between them.

2) Other Prior Works Related to the Present Study:

Eling and Loperfido analysed a dataset from the point of view of actuarial modelling and pricing. Bagchi and Udo used a variant of the Gompertz model to analyse 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. analysed 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.

3.SYSTEM ANALYSIS

3.1 Existing system

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 stabilising? 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 analysed 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 analysed 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 [9] 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 analysed separately.Recently, researchers started modelling data breach incidents. Maillart and Sornette 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. analysed 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., analysed a dataset that is combined from corresponds to organisational 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.

3.2 Proposed system

In this paper, we make the following three contributions. First, we show that both the hacking breach incident inter-arrival times (reflecting incident frequency) and breach sizes should be modelled 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 ARMA-GARCH model can adequately describe the evolution of the hacking breach sizes, where ARMA is acronym for "AutoRegressive and Moving Average" and GARCH is acronym for "Generalised AutoRegressive Conditional Heteroskedasticity." We show that these stochastic process models can predict the inter-arrival times and the breach sizes.

To the best of our knowledge, this is the first paper showing that stochastic processes, rather than distributions, should be used to model these cyber threat factors. 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. To the best of our knowledge, this is the first work showing the existence of this dependence and the consequence of ignoring it. 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 stabilising 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 in-depth to understand the nature of data breach risks.

3.3 System Specification

Software Requirements:

- Operating System : Windows 7.
- Coding Language : Python.
- Front-End : Python.
- Designing : Html, css, javascript.
- Data Base : MySQL.

Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 14’ Colour Monitor.
- Mouse : Optical Mouse.
- Ram : 512mb

4.SYSTEM STUDY

4.1. FEASIBILITY STUDY

The feasibility of the project is analysed 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:

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organisation. 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 customised products had to be purchased.

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.

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.

5.MODULES

5.1. UPLOAD DATA

The data resource to database can be uploaded by both administrator and authorised user. The data can be uploaded with key in order to maintain the secrecy of the data that is not released without knowledge of user. The users are authorised based on their details that are shared to admin and admin can authorise each user. Only Authorised users are allowed to access the system and upload or request for files.

5.2. ACCESS DETAILS

The access of data from the database can be given by administrators. Uploaded data are managed by admin and admin is the only person to provide the rights to process the accessing details and approve or unapproved users based on their details.

5.3.USER PERMISSIONS

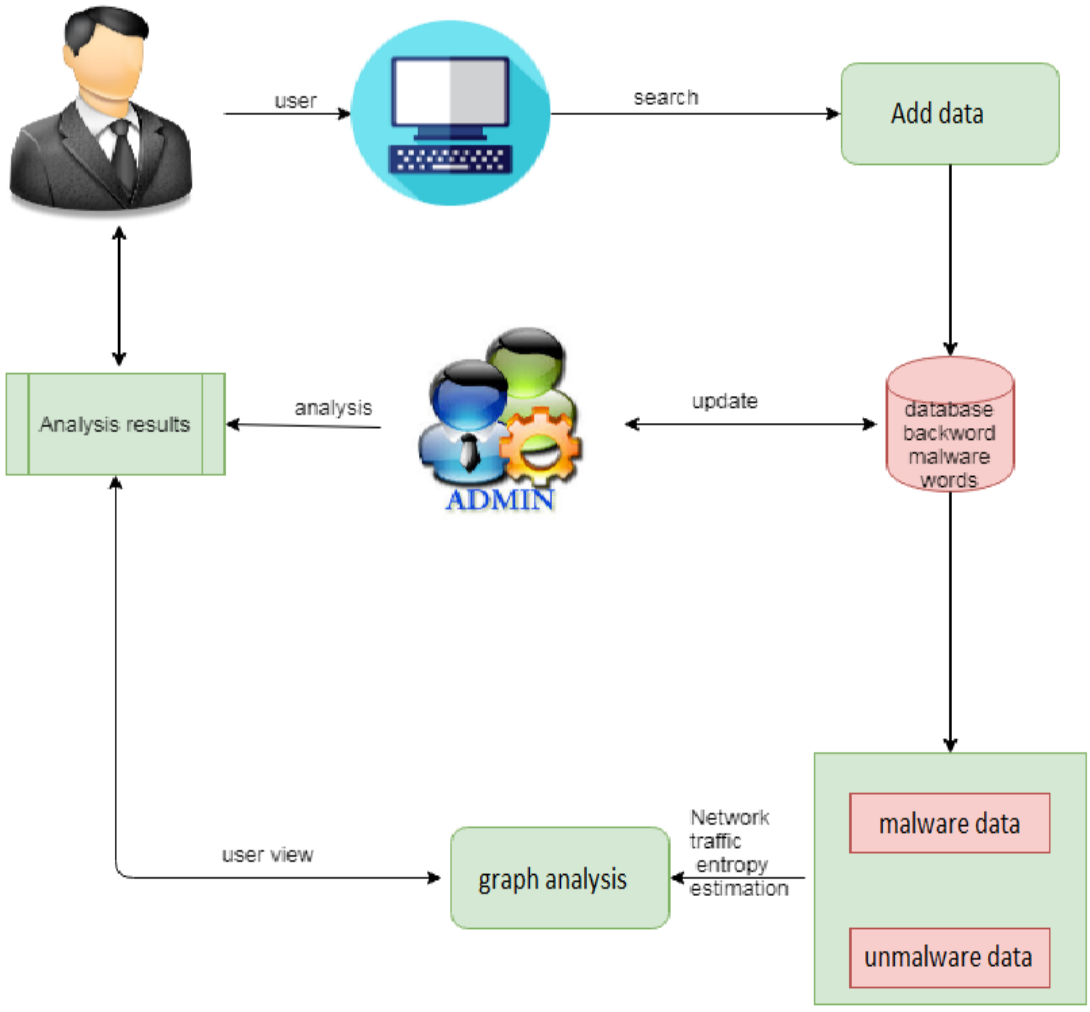
The data from any resources are allowed to access the data with only permission from administrator. Prior to access data, users are allowed by admin to share their data and verify the details which are provided by user. If user is access the data with wrong attempts then, users are blocked accordingly. If user is requested to unblock them, based on the requests and previous activities admin is unblock users.

5.4.DATA ANALYSIS

Data analyses are done with the help of graph. The collected data are applied to graph in order to get the best analysis and prediction of dataset and given data policies. The dataset can be analysed through this pictorial representation in order to better understand of the data details.

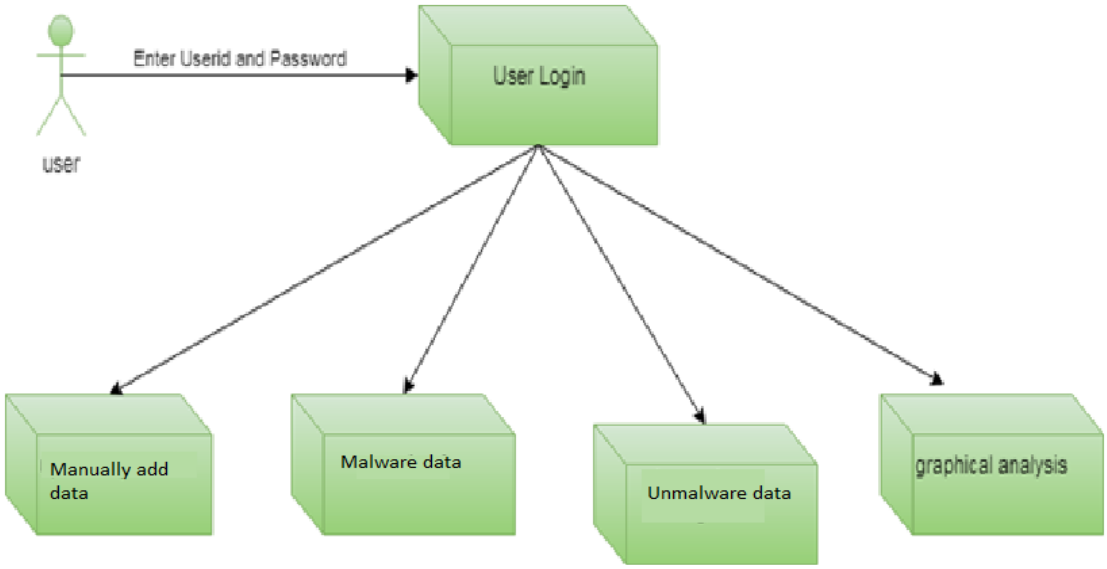
6.SYSTEM DESIGN

6.1 Architecture Diagram

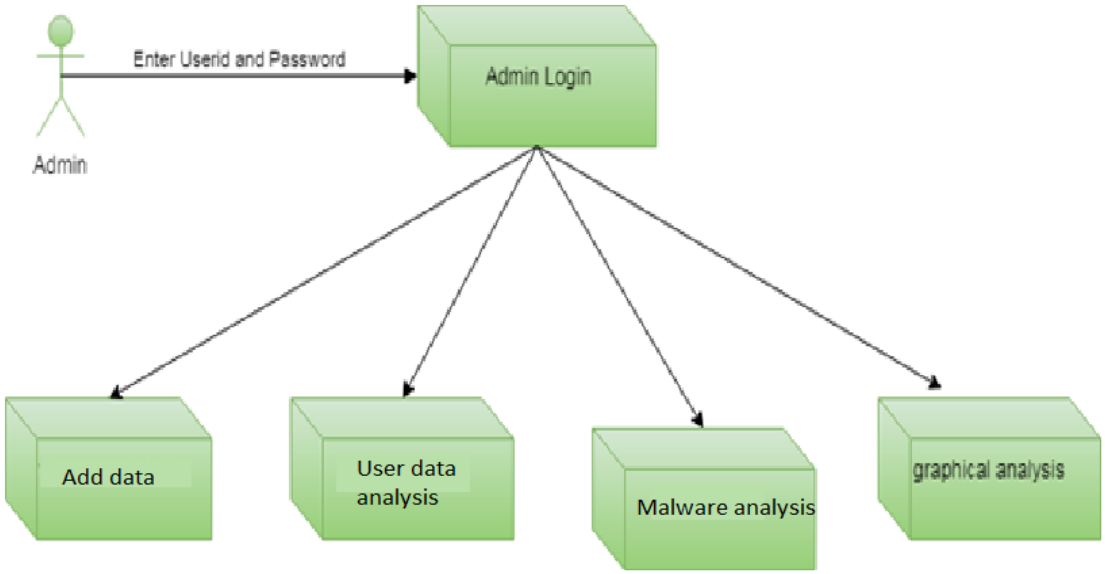


6.2 COMPONENT DIAGRAM

a.User

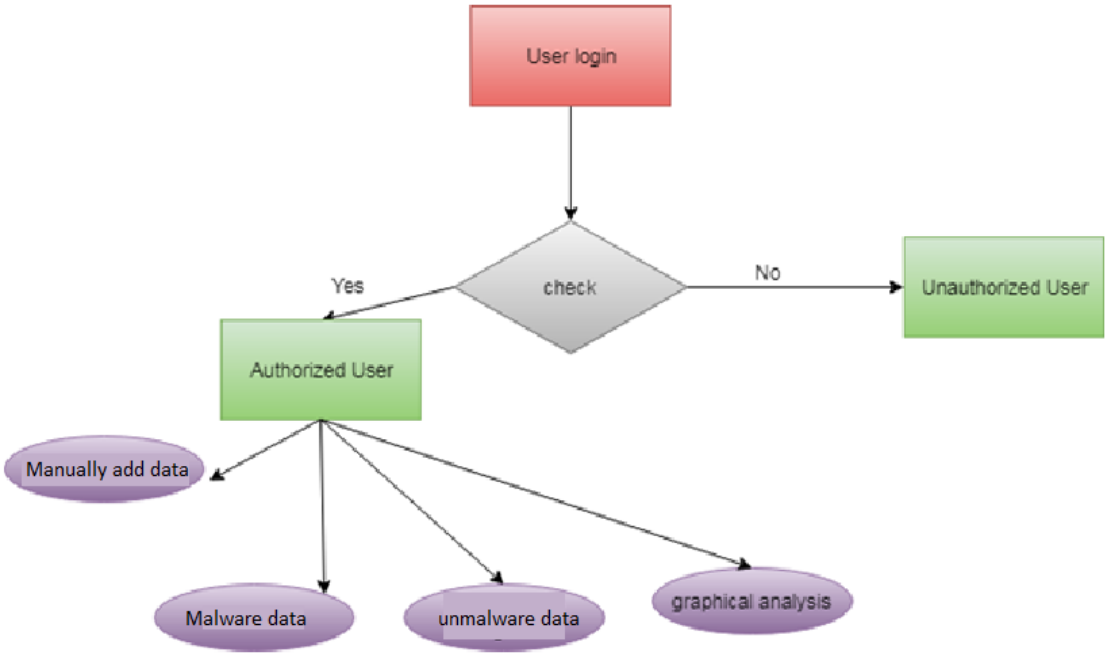


b.Admin



6.3 ER DIAGRAM

a.User



b.Admin

UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardised general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualisation, Constructing and documenting the artefacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

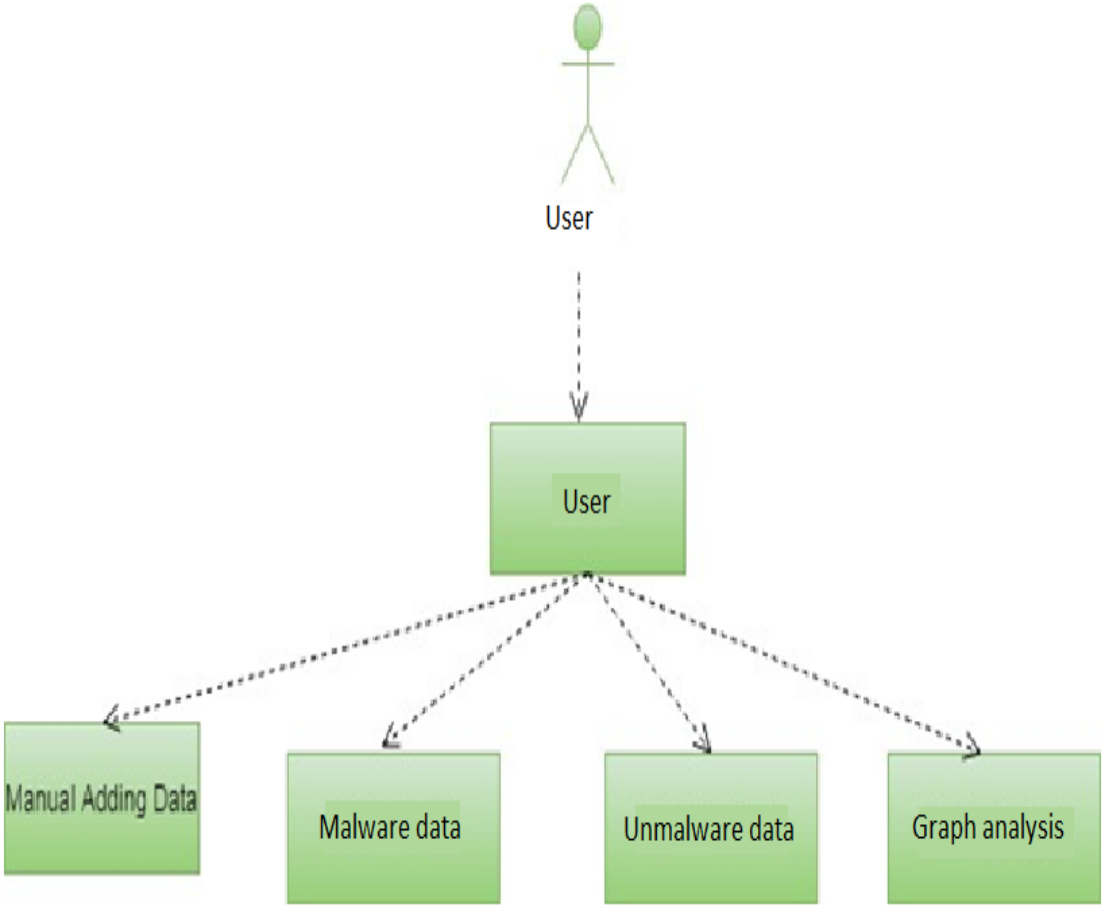
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialisation mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

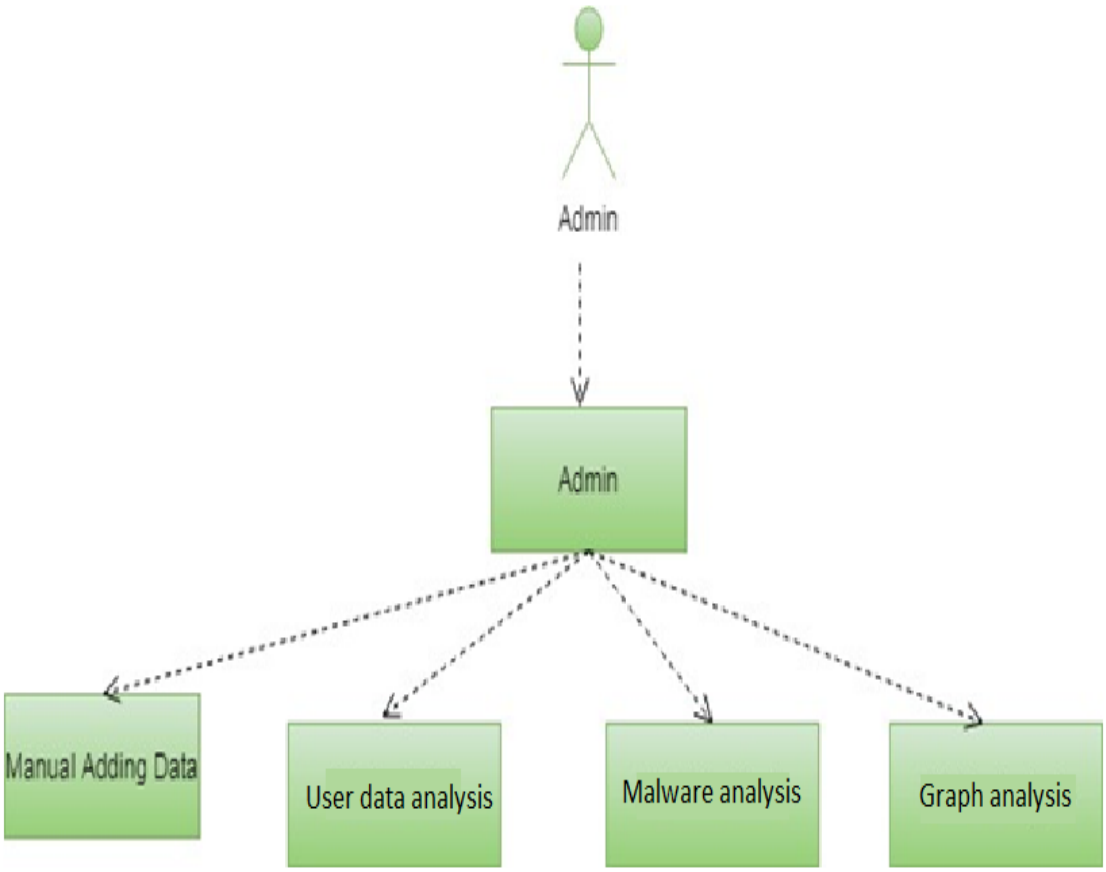
6.4 USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

a.User

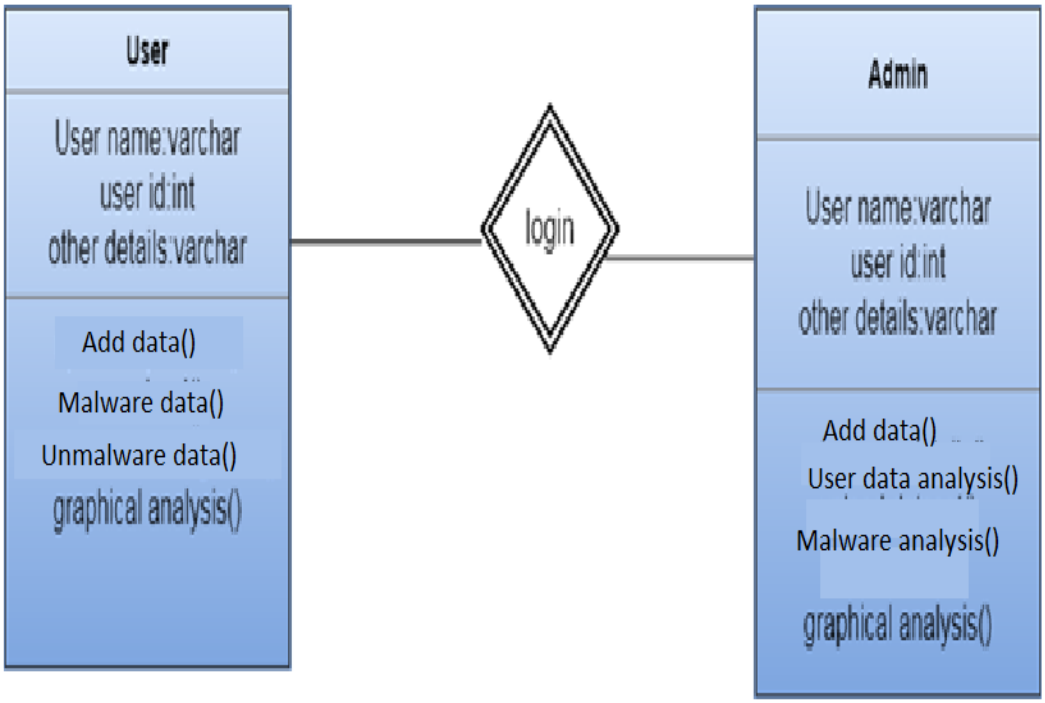


b.Admin


Admin
Admin
Manual Adding Data
User data analysis
Malware analysis
Graph analysis

6.5 CLASS DIAGRAM

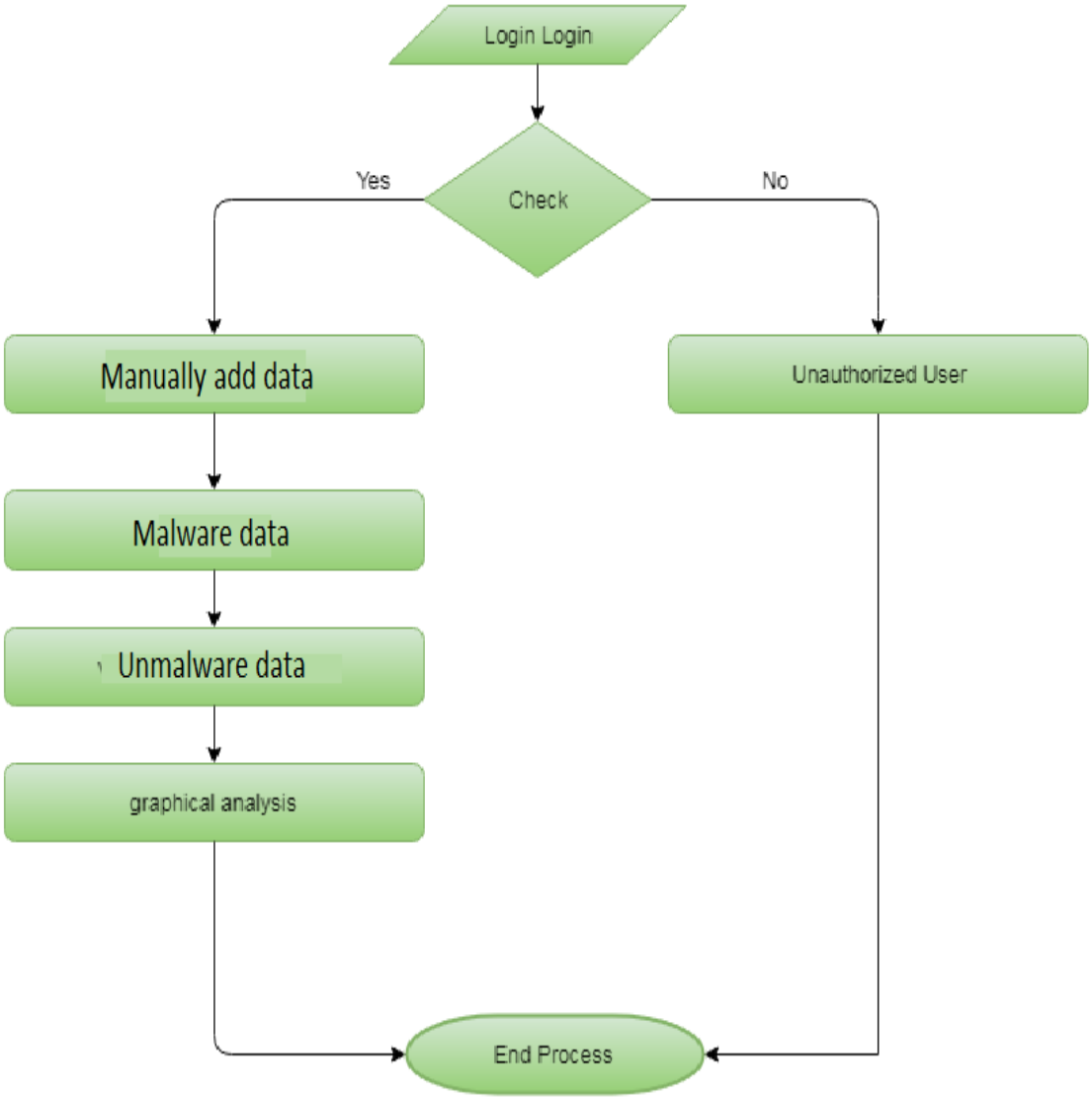
In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



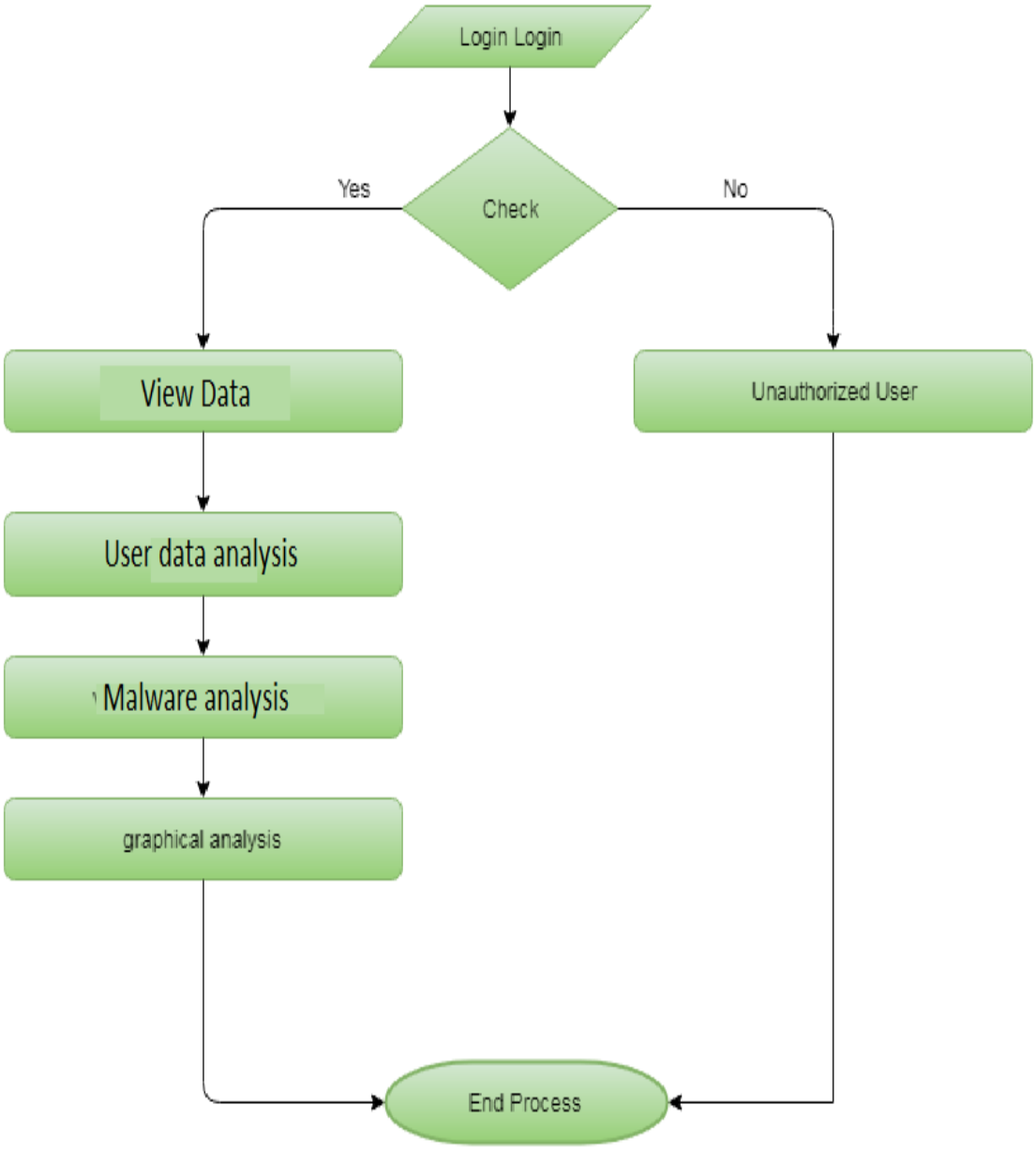
6.6 DATA FLOW DIAGRAM

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modelling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

a.User


Login Login
Check
Yes
No
Manually add data
Unauthorized User
Malware data
Unmalware data
graphical analysis
End Process

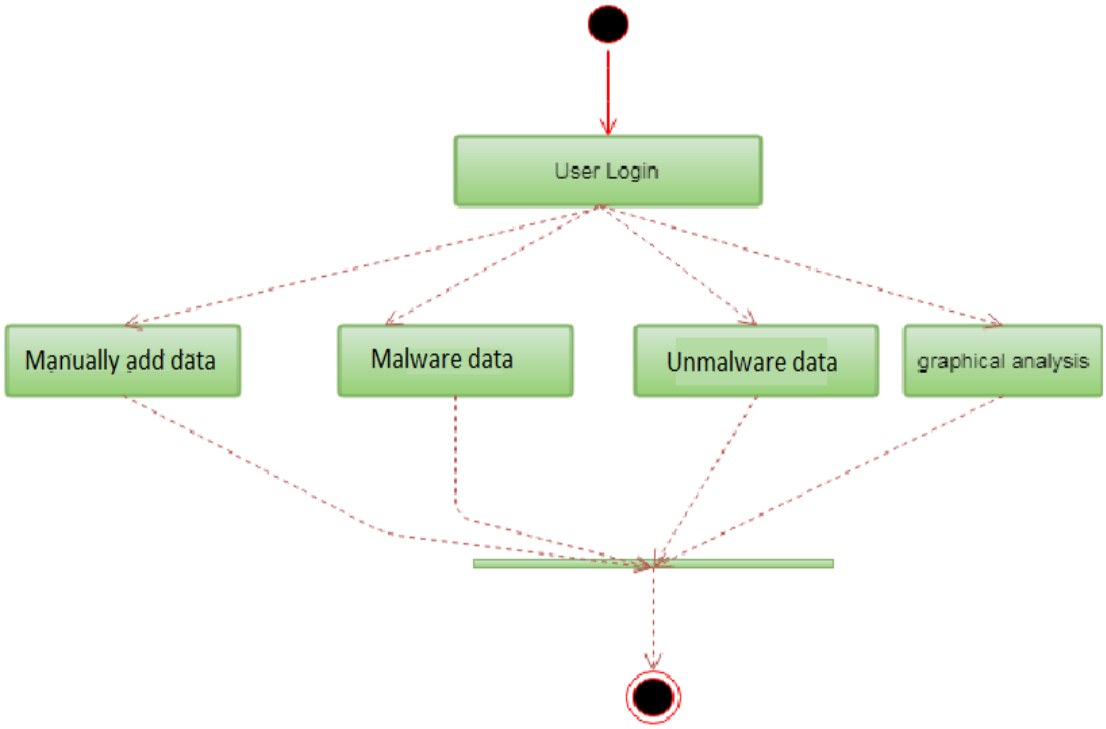
b.Admin


Login Login
Check
Yes
No
View Data
Unauthorized User
User data analysis
Malware analysis
graphical analysis
End Process

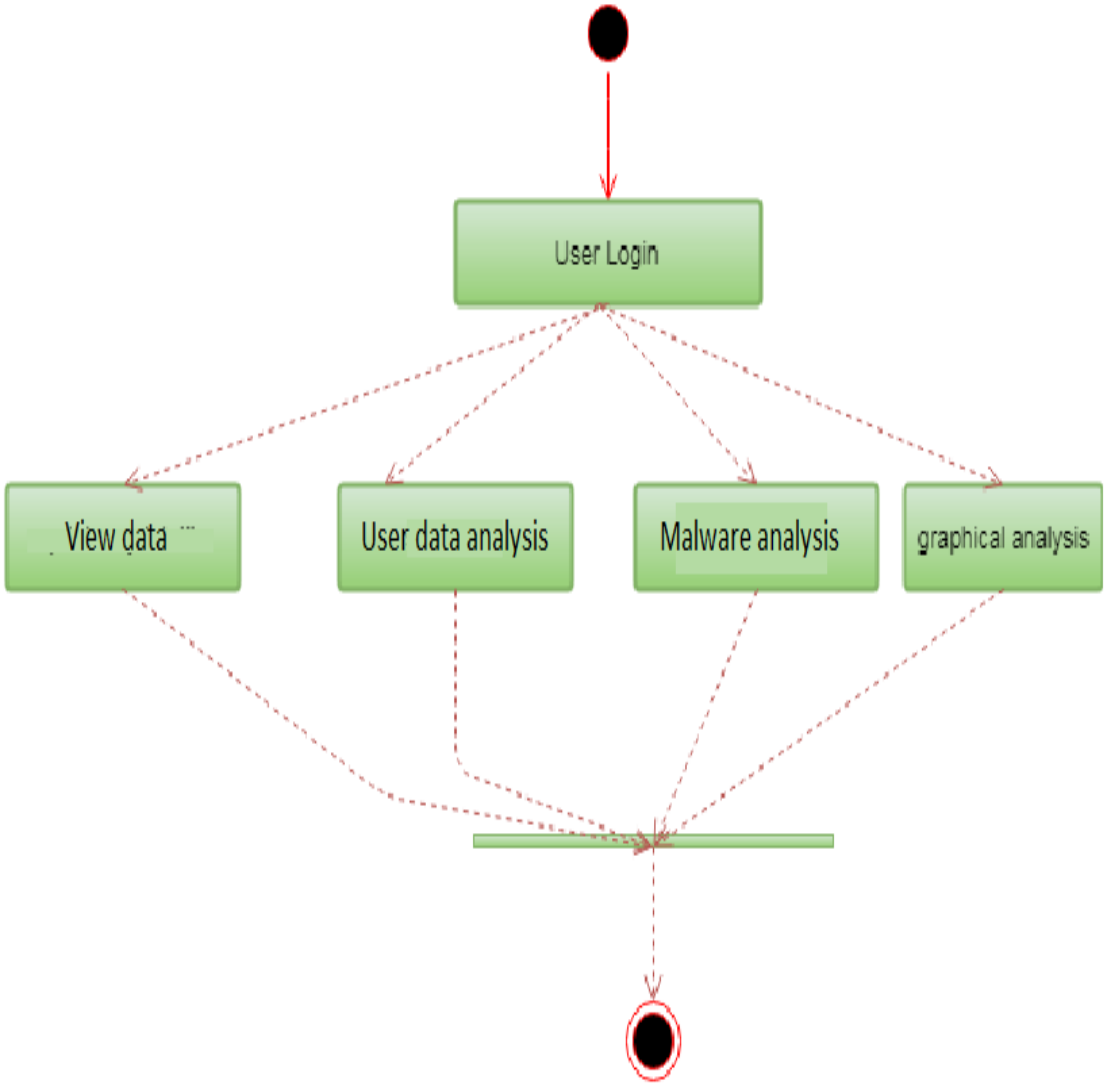
6.7 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

a.User



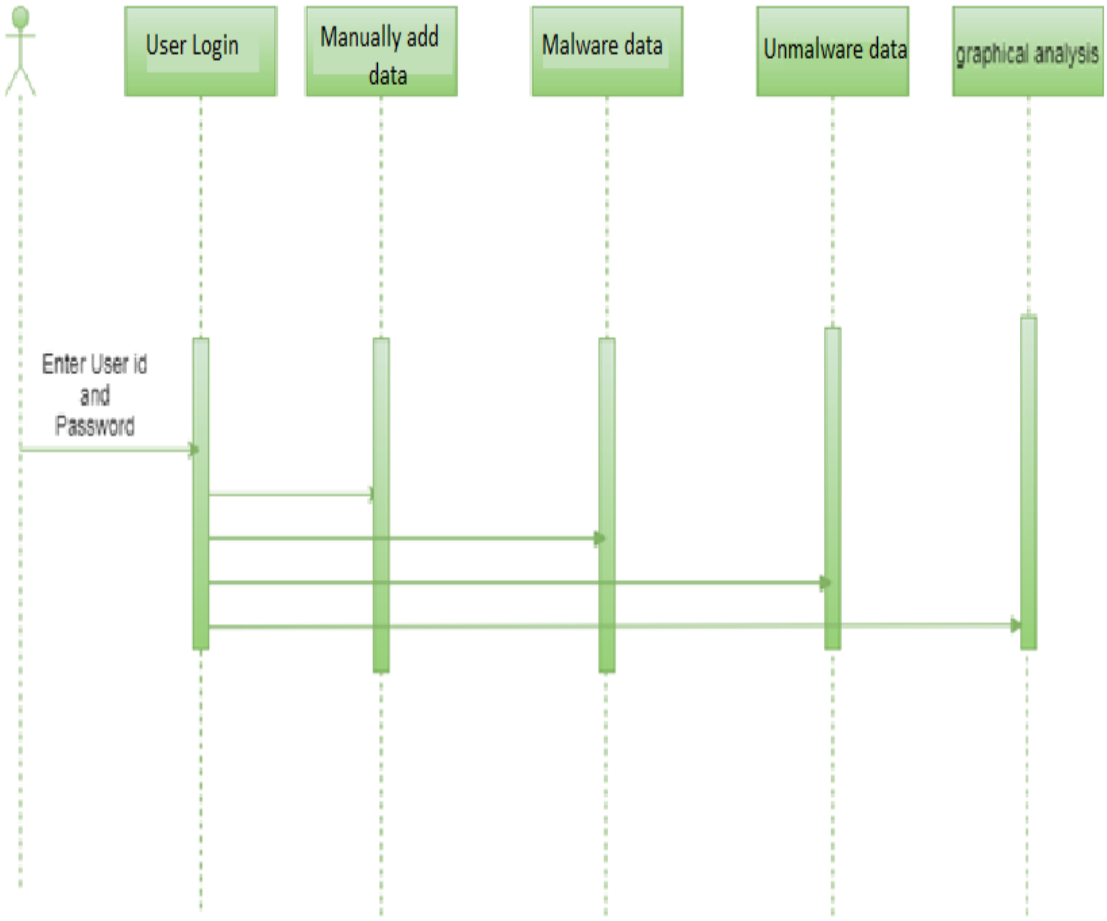
b.Admin


User Login
View data
User data analysis
Malware analysis
graphical analysis

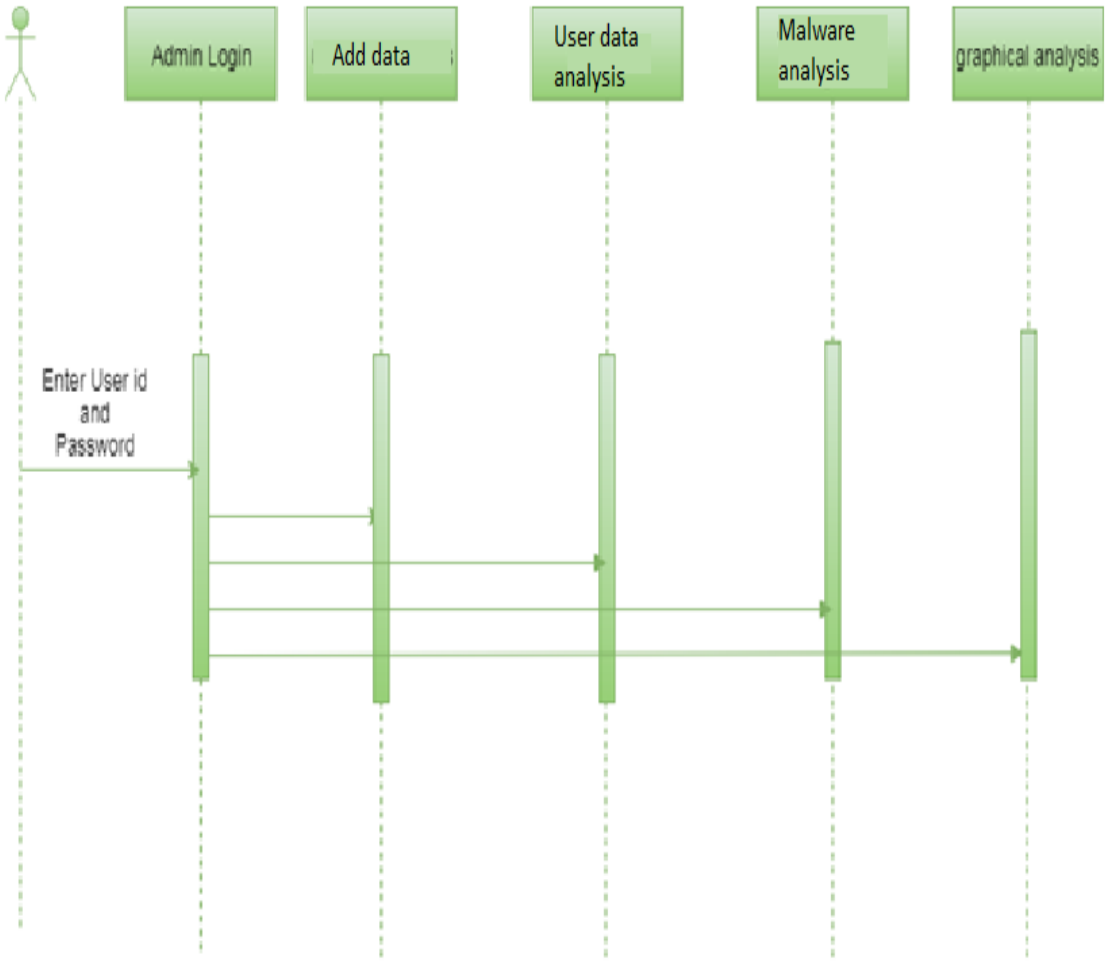
6.8 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

a.User



b.Admin


Admin Login
Add data
User data analysis
Malware analysis
graphical analysis
Enter User id and Password

7.IMPLEMENTATION

7.1. Software Environment

What is Python?

- Python is a High level, structured, open-source programming language that can be used for a wide variety of programming tasks.
- Python within itself is an interpreted programming language that is automatically compiled into byte code before execution.
- It is also a dynamically typed language that includes (but does not require one to use) object-oriented features.
- NASA has used Python for its software systems and has adopted it as the standard scripting language for its Integrated Planning System.
- Python is also extensively used by Google to implement many components of its Web Crawler and Search Engine & Yahoo! for managing its discussion groups.

History of Python

- Python was created by Guido Van Rossum.
- The design began in the late 1980s and was first released in February 1991.

Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late 70s. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

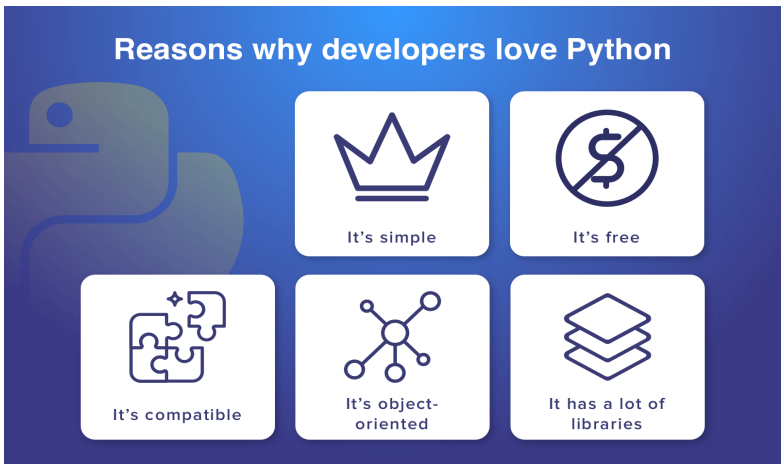
Python Version History

Implementation started - December 1989

Internal releases - 1990

Version No.	Date of Released
0.9	February 20, 1991
1.0	January, 1994
2.0	October 16, 2000
3.0	December 3, 2008
3.1	June 27, 2009
3.2	February 20, 2011
3.3	September 29, 2012
3.4	March 16, 2014
3.5	September 13, 2015
3.6	December 23, 2016
3.7	June 27, 2018

Features of Python Programming



1. **A simple language which is easier to learn**

    - Python has a very simple and elegant syntax.
    - It's much easier to read and write Python programs compared to other languages like: C++, Java, C#.
    - Python makes programming fun and allows you to focus on the solution rather than syntax.
    - If you are a newbie, it's a great choice to start your journey with Python.

2. **Free and open-source**

    - You can freely use and distribute Python, even for commercial use.
    - Not only you can use and distribute software's written in it, you can even make changes to the Python's source code.
    - Python has a large community constantly improving it in each iteration.

3. **Portability**

    - You can move Python programs from one platform to another and run it without any changes.
    - It runs seamlessly on almost all platforms including Windows, Mac OS and Linux.

4. **Extensible and Embeddable**

    - Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.
    - This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

5. **A high-level, interpreted language**

    - Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.
    - Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

6. **Large standard libraries to solve common tasks**

    - Python has several standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself.
    - For example: Need to connect MySQL database on a Web server? You can use MySQL dB library using import MySQL db.
    - Standard libraries in Python are well tested and used by hundreds of people. So, you can be sure that it won't break your application.

7. **Object-oriented**

    - Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.
    - With OOP, you can divide these complex problems into smaller sets by creating objects.

DJANGO

Django is a Web framework written in Python.

A Web framework is a software that supports the development of dynamic Web sites, applications, and services.

It provides a set of tools and functionalities that solves many common problems associated with Web development, such as security features, database access, sessions, template processing, URL routing, internationalisation, localisation, and much more.

Using a Web framework, such as Django, enables us to develop secure and reliable Web applications very quickly in a standardised way.

The development of Django is supported by the Django Software Foundation, and it's sponsored by companies like JetBrains and Instagram.

Who's Using Django?

It's good to know who is using Django out there, so to have an idea what you can do with it. Among the biggest Web sites using Django we have: Instagram, Disqus, Mozilla, Bitbucket, Last.fm, National Geographic.

Installation

The first thing we need to do is install some programs on our machine so to be able to start playing with Django. The basic setup consists of installing

- **Python**
- **Virtualenv**
- **Django**

Using virtual environments is not mandatory, but it's highly recommended.

Installing Virtualenv

we are going to use **pip**, a tool to manage and install Python packages, to install **virtualenv**.

In the Command Prompt, execute the command below:

```
pip install virtualenv
```

From now on, everything we install, including Django itself, will be installed inside a Virtual Environment.

```
mkdir myproject
cd myproject
```

This folder is the higher level directory that will store all the files and things related to our Django project, including its virtual environment.

let's start by creating our very first virtual environment and installing Django.

Inside the **myproj** folder:

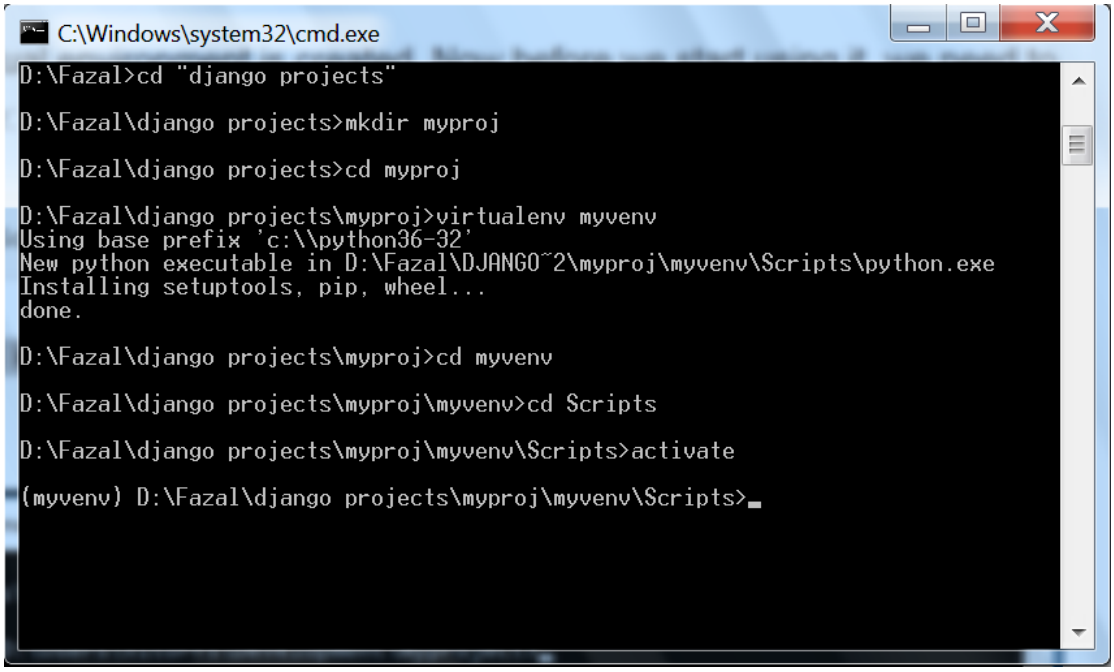
```
virtualenv myvenv
```

Our virtual environment is created.

Now before we start using it, we need to activate:

```
myvenv\Scripts\activate
```

You will know it worked if you see **(venv)** in front of the command line, like this:



to deactivate the **venv** run the command below:

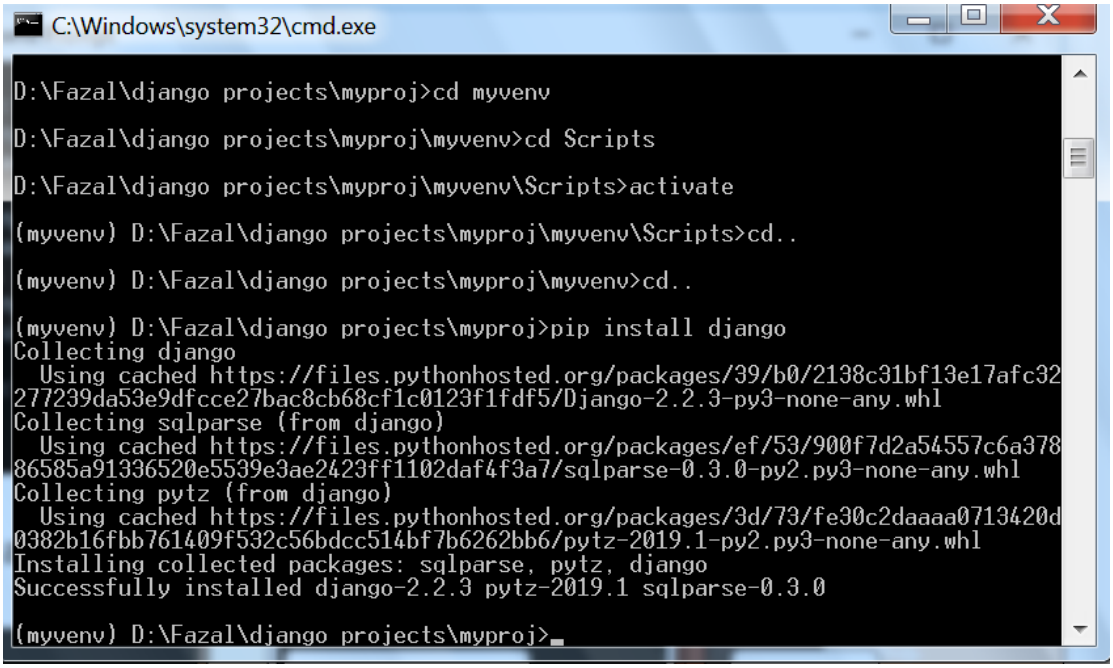
```
venv\Scripts\deactivate.bat
```

But let's keep it activated for the next steps.

Installing Django

Now that we have the **venv** activated, run the following command to install Django:

```
pip install django
```



Starting a New Project

To start a new Django project, run the command below:

```
django-admin startproject myproject
```

The command-line utility **django-admin** is automatically installed with Django.

After we run the command above, it will generate the base folder structure for a Django project.

Our initial project structure is composed of five files:

- **manage.py**: a shortcut to use the **django-admin** command-line utility. It's used to run management commands related to our project.

  We will use it to run the development server, run tests, create migrations and much more.

- **__init__.py**: this empty file tells Python that this folder is a Python package.
- **settings.py**: this file contains all the project's configuration.
- **urls.py**: this file is responsible for mapping the routes and paths in our project.

  For example, if you want to show something in the URL /about/, you have to map it here first.

- **wsgi.py**: this file is a simple gateway interface used for deployment.

  You don't have to bother about it. Just let it be for now.

Django comes with a simple web server installed.

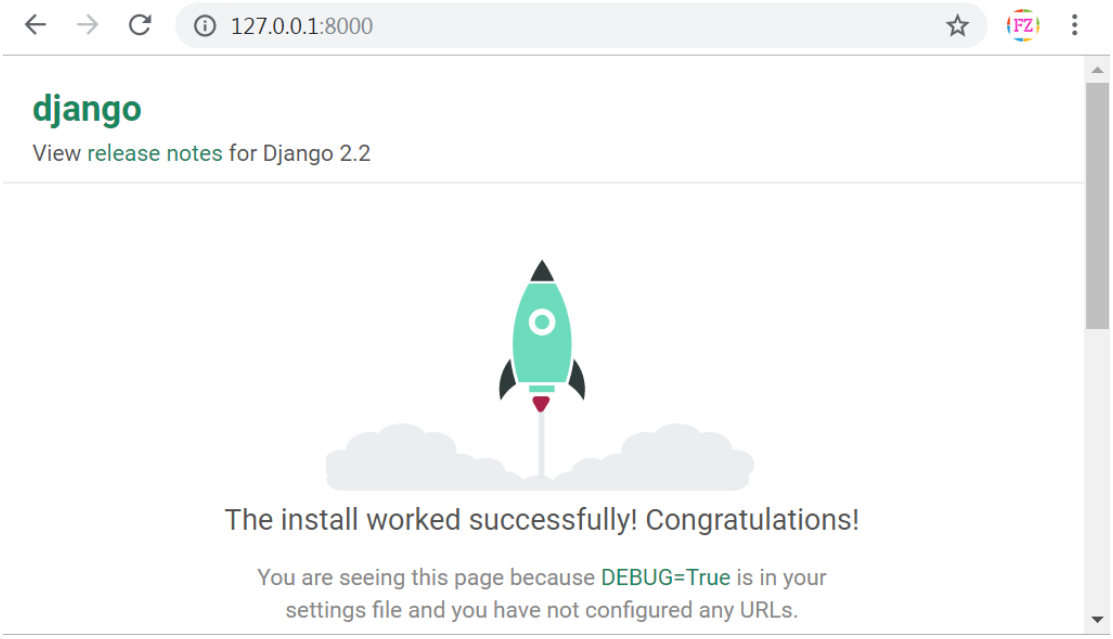
It's very convenient during the development, so we don't have to install anything else to run the project locally.

We can test it by executing the command:

```
python manage.py runserver
```

For now, you can ignore the migration errors; we will get to that later.

Now open the following URL in a Web browser: **http://127.0.0.1:8000** and you should see the following page:



Hit CTRL + BREAK to stop the development server.

Django Apps

In the Django philosophy we have two important concepts:

- **app**: is a Web application that does something.

  An app usually is composed of a set of models (database tables), views, templates, tests.

- **project**: is a collection of configurations and apps.

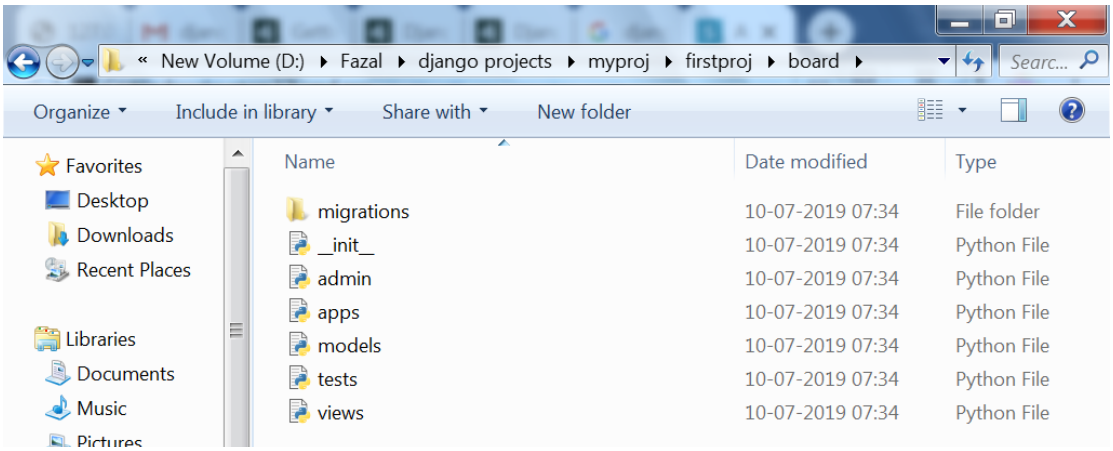
  One project can be composed of multiple apps, or a single app.

It's important to note that you can't run a Django **app** without a **project**. Simple websites like a blog can be written entirely inside a single app, which could be named **blog** or **weblog** for example.

let's create a simple Web Forum or Discussion Board. To create our first app, go to the directory where the **manage.py** file is and executes the following command:

```
django-admin startapp boards
```

Notice that we used the command **startapp** this time.



So, let's first explore what each file does:

- **migrations/**: here Django store some files to keep track of the changes you create in the **models.py** file, so to keep the database and the **models.py** synchronized.
- **admin.py**: this is a configuration file for a built-in Django app called **Django Admin**.
- **apps.py**: this is a configuration file of the app itself.
- **models.py**: here is where we define the entities of our Web application. The models are translated automatically by Django into database tables.
- **tests.py**: this file is used to write unit tests for the app.
- **views.py**: this is the file where we handle the request/response cycle of our Web application.

Now that we created our first app, let's configure our project to *use* it.

To do that, open the **settings.py** and try to find the `INSTALLED_APPS` variable:

settings.py

```
INSTALLED_APPS = [
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
```

```
'django.contrib.messages',
'django.contrib.staticfiles',
]
```

As you can see, Django already come with 6 built-in apps installed. They offer common functionalities that most Web applications need, like authentication, sessions, static files management (images, javascripts, css, etc.) and so on.

Hello, World!

Let's write our first **view**. We will explore it in great detail in the next tutorial. But for now, let's just experiment how it looks like to create a new page with Django.

Open the **views.py** file inside the **boards** app, and add the following code:

views.py

```
fromdjango.httpimportHttpResponse

defhome(request):
returnHttpResponse('Hello, World!')
```

Views are Python functions that receive an `HttpRequest` object and returns an `HttpResponse` object. Receive a *request* as a parameter and returns a *response* as a result. That's the flow you have to keep in mind!

So, here we defined a simple view called **home** which simply returns a message saying **Hello, World!**.

Now we have to tell Django *when* to serve this view. It's done inside the **urls.py** file:

urls.py

```
fromdjango.conf.urlsimporturl
fromdjango.contribimportadmin

fromboardsimportviews

urlpatterns=[
```

```
url(r'^$',views.home,name='home'),
url(r'^admin/',admin.site.urls),
]
```

If you compare the snippet above with your **urls.py** file, you will notice I added the following new line: `url(r'^$', views.home, name='home')` and imported the **views** module from our app **boards** using `from boards import views`.

As I mentioned before, we will explore those concepts in great detail later on.

But for now, Django works with **regex** to match the requested URL. For our **home** view, I'm using the `^$` regex, which will match an empty path, which is the homepage (this url: **http://127.0.0.1:8000**). If I wanted to match the URL **http://127.0.0.1:8000/homepage/**, my url would be:

```
url(r'^homepage/$', views.home, name='home').
```

Let's see what happen:

```
python manage.py runserver
```

In a Web browser, open the http://127.0.0.1:8000 URL:

8. OUTPUT SCREENS

Fig: user login page

Fig: User Registration

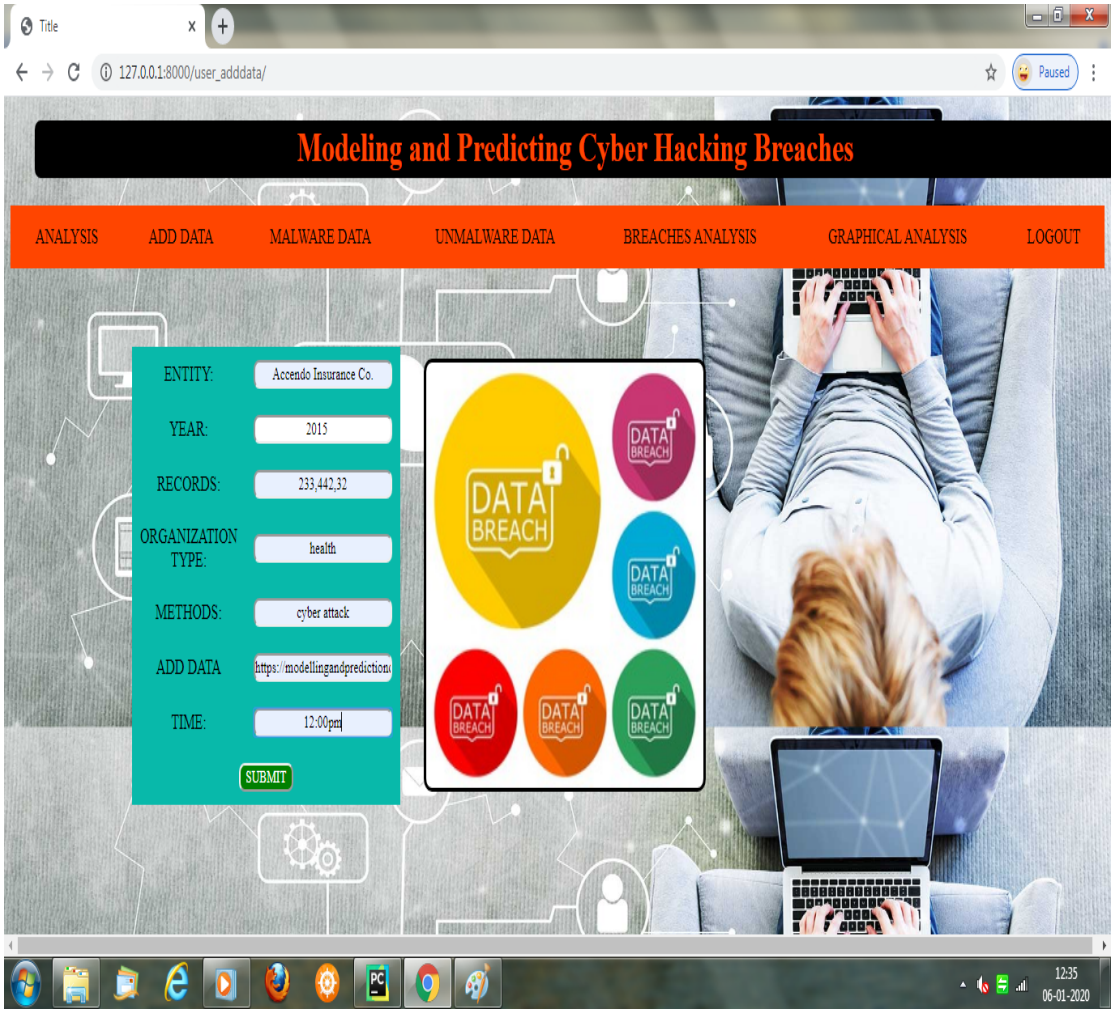


Fig: User entering data

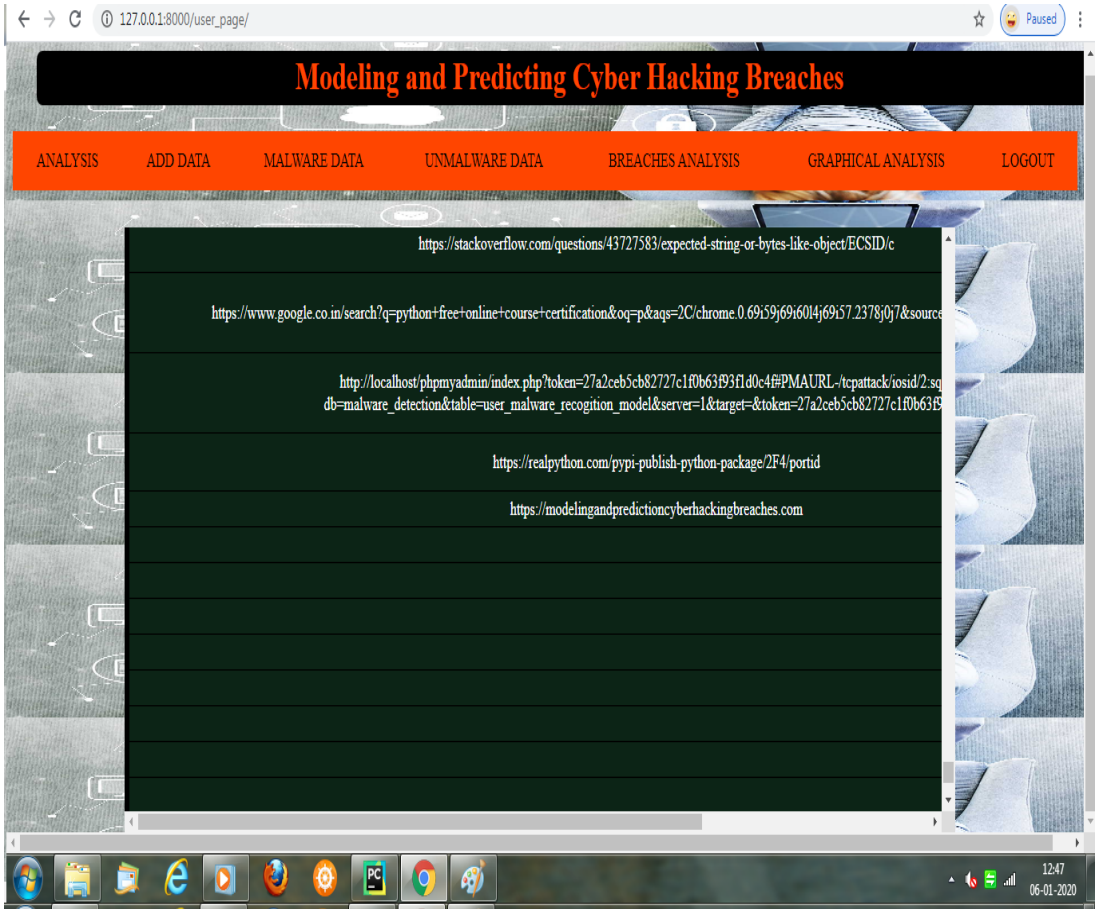


Fig: User checking data

Fig: User checking malware data

Fig: Data analysis

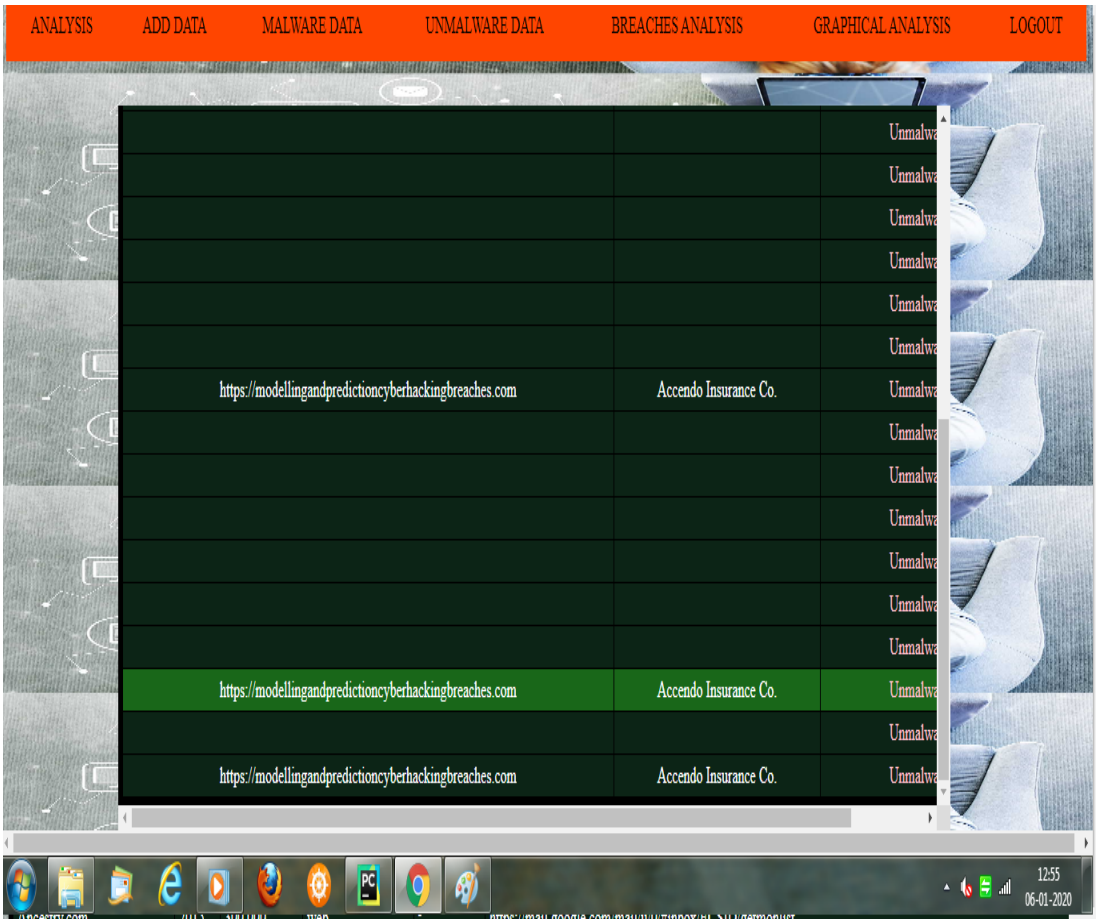


Fig: Malware data

Fig: Breach analysis

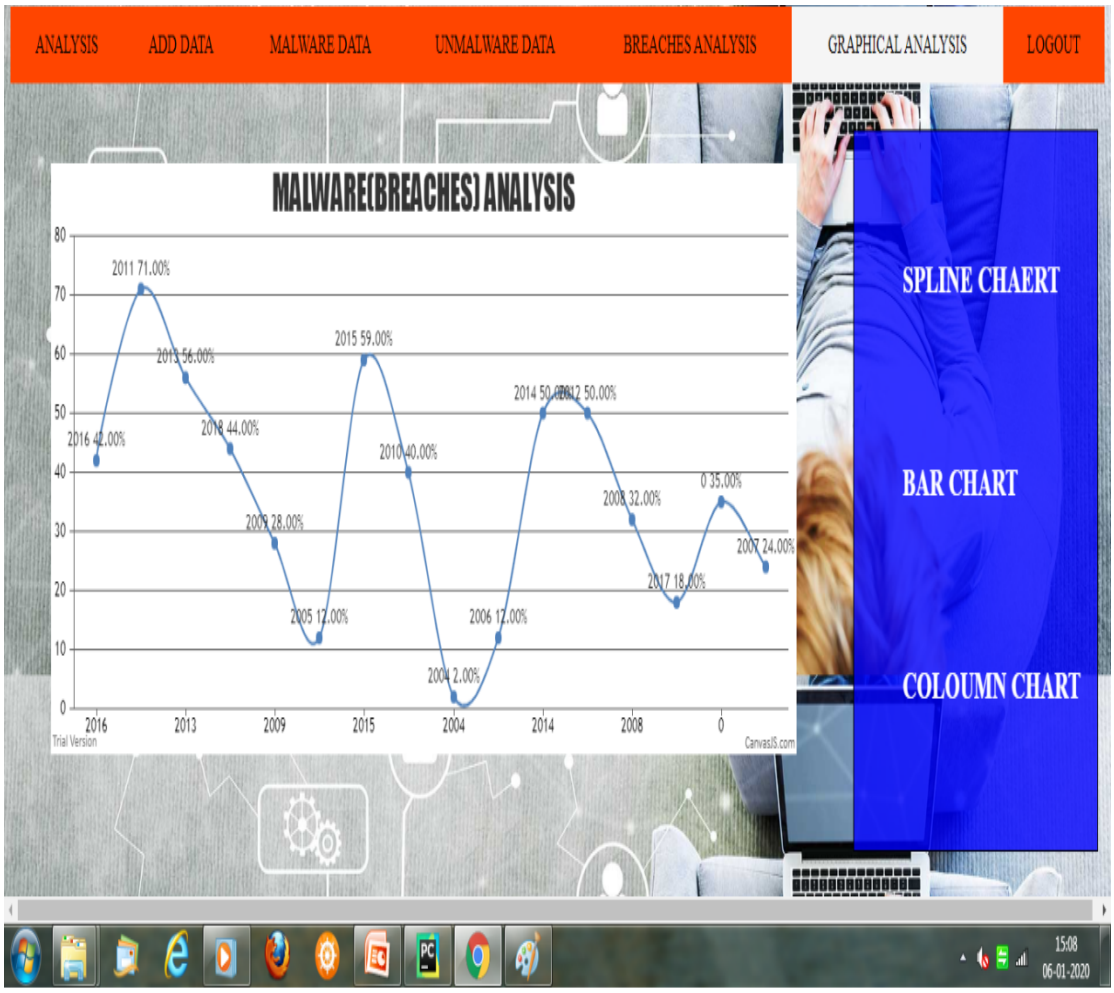


Fig: Graphical analysis

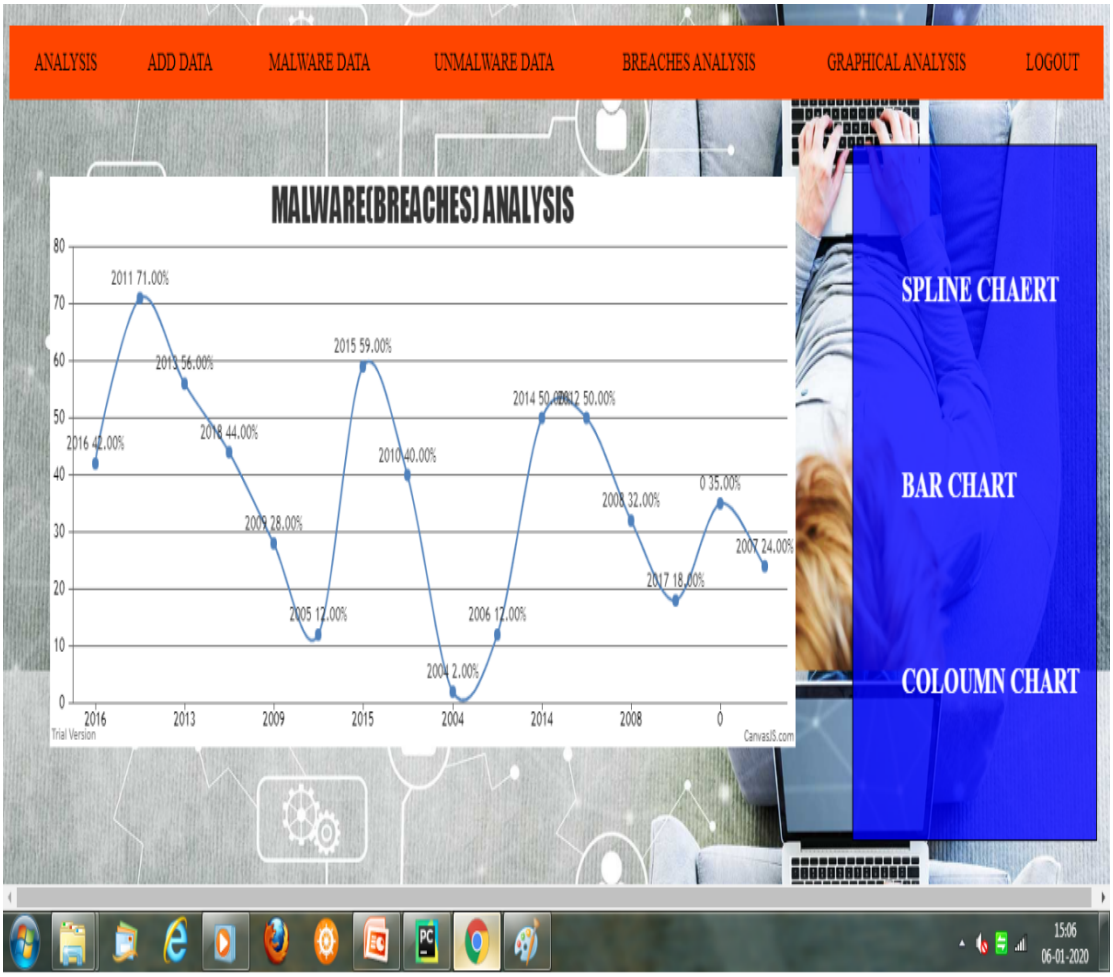


Fig: Spline chart

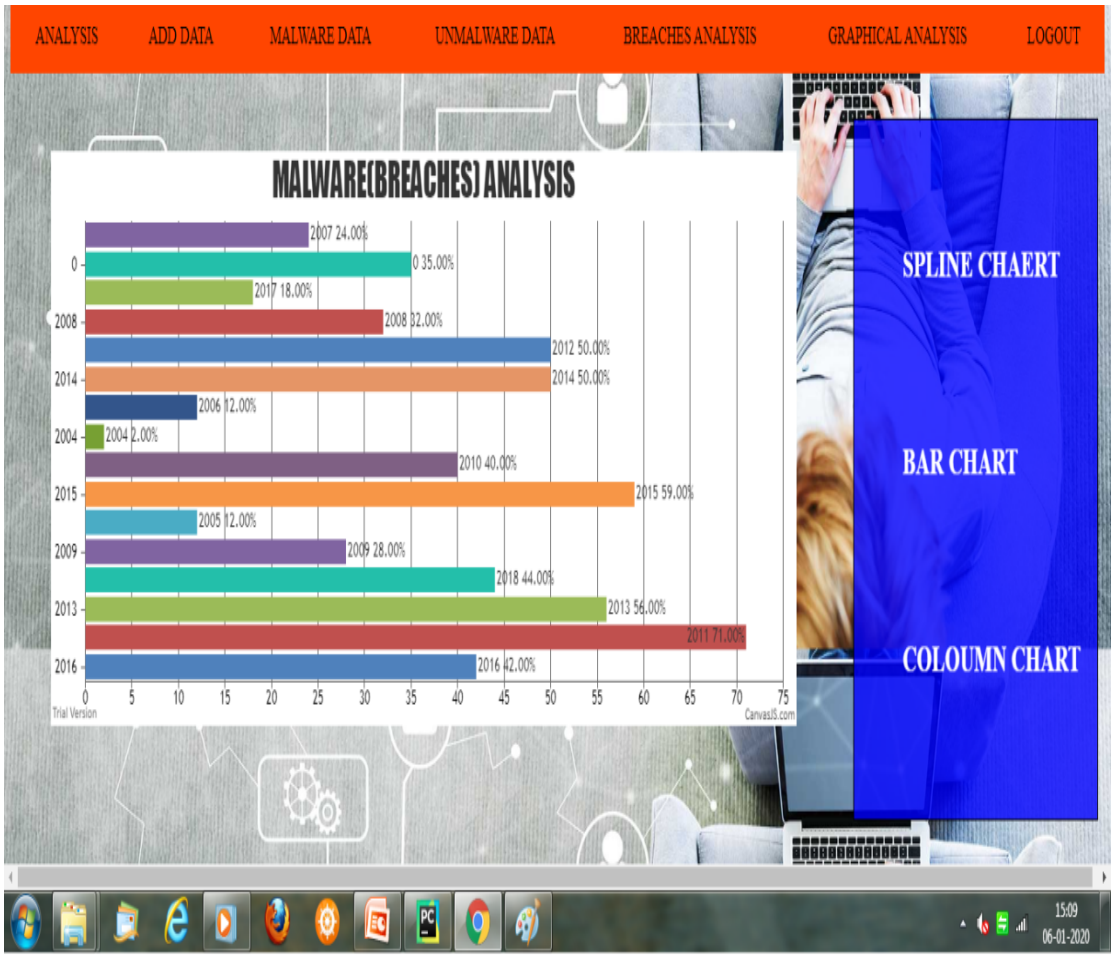


Fig:Bar chart

Fig: Coloumn chart

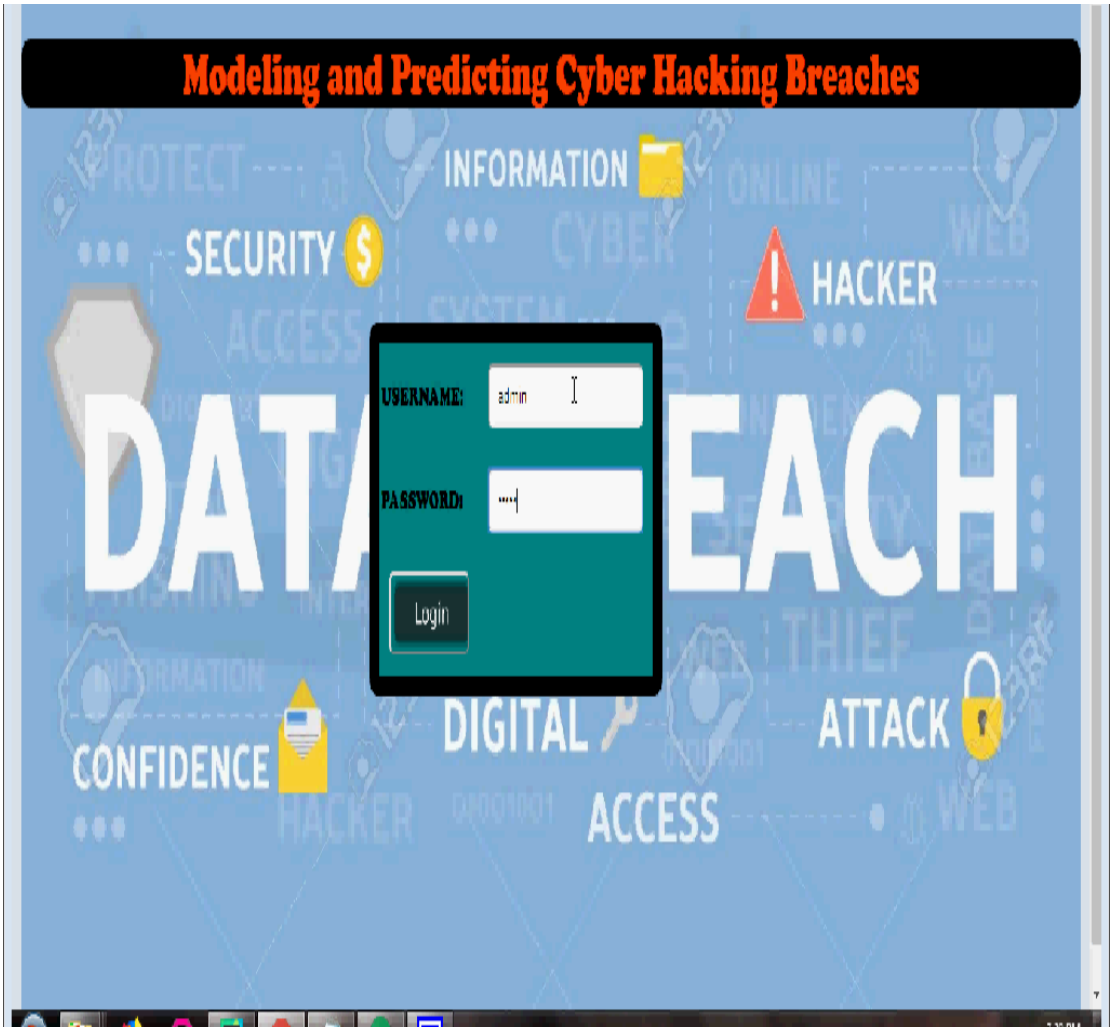


Fig: Admin login

Fig: User datails analysis

MALWARE NAME	NETWORK TRAFIC POSSITION	METHOD
Man-in-the-middle (MitM) attack	hacked	48
Phishing and spear phishing attacks	poor security	4
Drive-by attack	hacked	36
Password attack	lost / stolen media	10
SQL injection attack	hacked	34
Cross-site scripting (XSS) attack	lost / stolen media	10
Eavesdropping attack	lost / stolen media	8
Birthday attack	poor security	10
Teardrop attack	hacked	34
Phishing and spear phishing attacks	inside job, hacked	2
Drive-by attack	accidentally published	4
Password attack	hacked	34
Cross-site scripting (XSS) attack	poor security	4

Fig: Admin analysis

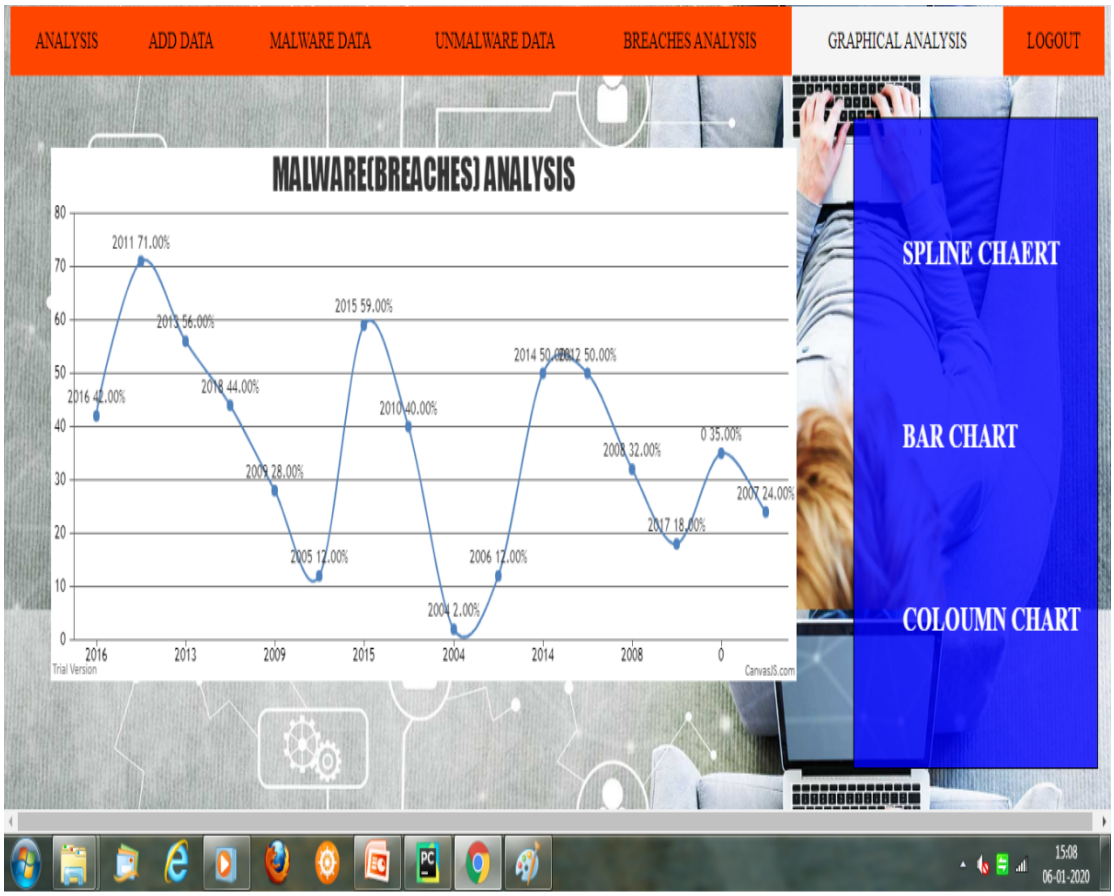


Fig: Graphical analysis

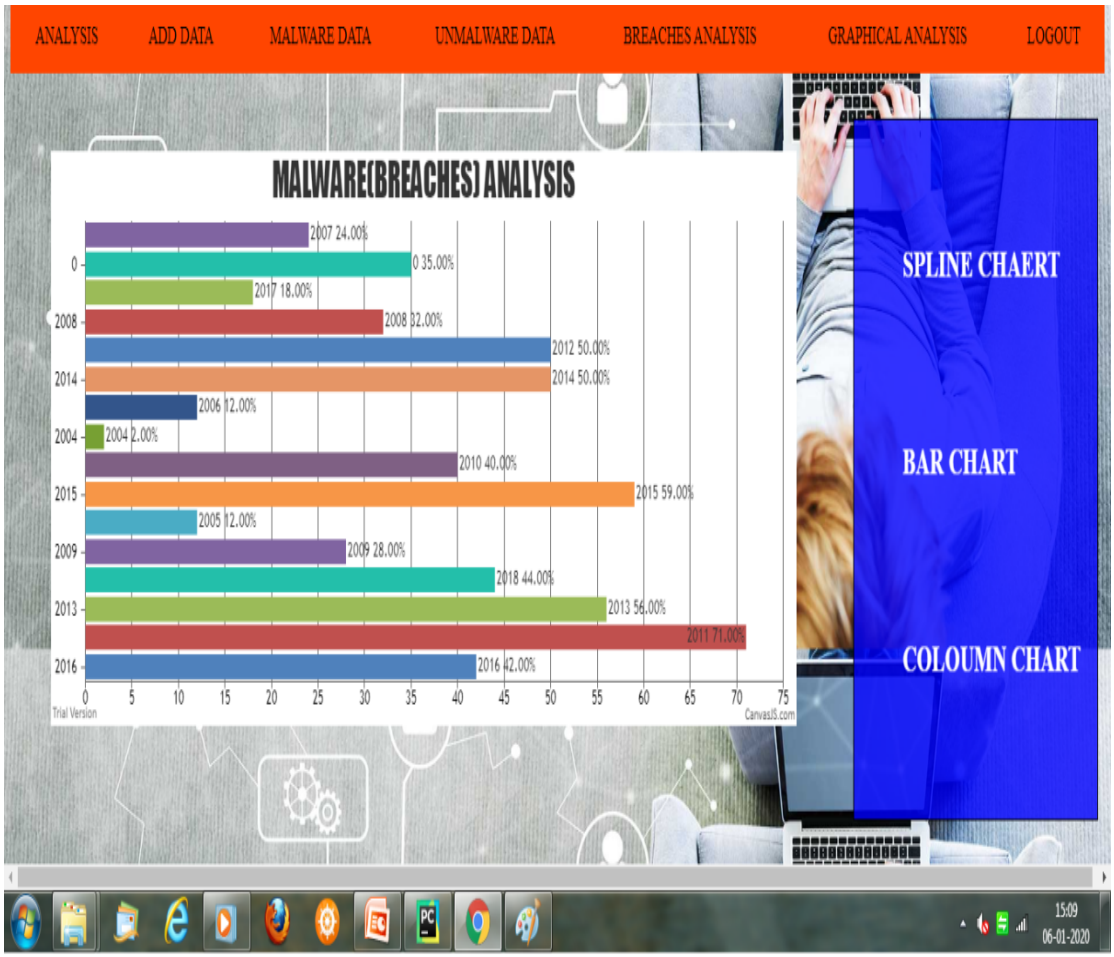


Fig:Bar chart

Fig: Column chart

9. 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 test. 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 actually 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 Centred 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.

Organisation 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, emphasising 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. CONCLUSION

We analysed a hacking breach dataset from the points of view of the incidents inter-arrival time and the breach size, and showed that they both should be modelled by stochastic processes rather than distributions. The statistical models developed in this paper show satisfactory fitting and prediction accuracies.

In particular, we propose using a copula-based approach to predict the joint probability that an incident with a certain magnitude of breach size will occur during a future period of time. Statistical tests show that the methodologies proposed in this paper are better than those which are presented in the literature, because the latter ignored both the temporal correlations and the dependence between the incidents inter-arrival times and the breach sizes.

We conducted qualitative and quantitative analyses to draw further insights. We drew a set of cybersecurity insights, including that the threat of cyber hacking breach incidents is indeed getting worse in terms of their frequency, but not the magnitude of their damage. The methodology presented in this paper can be adopted or adapted to analyse datasets of a similar nature.

11. 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 Identity Theft 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_NetDiligence-2016-Cyber-Claims-Study-ONLINE.pdf.

[6] M. Eling and W. Schnell, "What do we know about 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.

[9] B. Edwards, S. Hofmeyr, and S. Forrest, "Hype and heavy tails: A closer look at data breaches," J. Cybersecur., vol. 2, no. 1, pp. 3–14, 2016.

[10] S. Wheatley, T. Maillart, and D. Sornette, "The extreme risk of personal data breaches and the erosion of privacy," Eur. Phys. J. B, vol. 89, no. 1, p. 7, 2016.

[11] P. Embrechts, C. Klüppelberg, and T. Mikosch, Modelling Extremal Events: For Insurance and Finance, vol. 33. Berlin, Germany: Springer-Verlag, 2013.

[12] R. Böhme and G. Kataria, "Models and measures for correlation in cyber-insurance," in Proc. Workshop Econ. Inf. Secur. (WEIS), 2006, pp. 1–26.

[13] H. Herath and T. Herath, "Copula-based actuarial model for pricing cyber-insurance policies," Insurance Markets Companies: Anal. Actuarial Comput., vol. 2, no. 1, pp. 7–20, 2011.

[14] A. Mukhopadhyay, S. Chatterjee, D. Saha, A. Mahanti, and S. K. Sadhukhan, "Cyber-risk decision models: To insure it or not?" Decision Support Syst., vol. 56, pp. 11–26, Dec. 2013.

[15] M. Xu and L. Hua. (2017). Cybersecurity Insurance: Modelling and Pricing. [Online]. Available: https://www.soa.org/research-reports/ 2017/cybersecurity-insurance.

[16] M. Xu, L. Hua, and S. Xu, "A vine copula model for predicting the effectiveness of cyber defense early-warning," Technometrics, vol. 59, no. 4, pp. 508–520, 2017.

[17] C. Peng, M. Xu, S. Xu, and T. Hu, "Modelling multivariate cybersecurity risks," J. Appl. Stat., pp. 1–23, 2018.

[18] M. Eling and N. Loperfido, "Data breaches: Goodness of fit, pricing, and risk measurement," Insurance, Math. Econ., vol. 75, pp. 126–136, Jul. 2017.

[19] K. K. Bagchi and G. Udo, "An analysis of the growth of computer and Internet security breaches," Commun. Assoc. Inf. Syst., vol. 12, no. 1, p. 46, 2003.

[20] E. Condon, A. He, and M. Cukier, "Analysis of computer security incident data using time series models," in Proc. 19th Int. Symp. Softw. Rel. Eng. (ISSRE), Nov. 2008, pp. 77–86.

[21] Z. Zhan, M. Xu, and S. Xu, "A characterization of cybersecurity posture from network telescope data," in Proc. 6th Int. Conf. Trusted Syst., 2014, pp. 105–126. [Online]. Available: http://www.cs.utsa.edu/~shxu/socs/intrust14.pdf.

[22] Z. Zhan, M. Xu, and S. Xu, "Characterizing honeypot-captured cyber attacks: Statistical framework and case study," IEEETrans. Inf. Forensics Security, vol. 8, no. 11, pp. 1775–1789, Nov. 2013.

[23] Z. Zhan, M. Xu, and S. Xu, "Predicting cyber attack rates with extreme values," IEEE Trans. Inf. Forensics Security, vol. 10, no. 8, pp. 1666–1677, Aug. 2015.

[24] Y.-Z. Chen, Z.-G. Huang, S. Xu, and Y.-C. Lai, "Spatiotemporal patterns and predictability of cyberattacks," PLoS ONE, vol. 10, no. 5, p. e0124472, 2015.

[25] C. Peng, M. Xu, S. Xu, and T. Hu, "Modelling and predicting extreme cyber attack rates via marked point processes," J. Appl. Stat., vol. 44, no. 14, pp. 2534–2563, 2017.

[26] J. Z. Bakdash et al. (2017). "Malware in the future? forecasting analyst detection of cyber events." [Online]. Available: https://arxiv.org/abs/1707.03243.

[27] Y. Liu et al., "Cloudy with a chance of breach: Forecasting cyber security incidents," in Proc. 24th USENIX Secur. Symp., Washington, DC, USA, 2015, pp. 1009–1024.

[28] R. Sen and S. Borle, "Estimating the contextual risk of data breach: An empirical approach," J. Manage. Inf. Syst., vol. 32, no. 2, pp. 314–341, 2015.

[29] F. Bisogni, H. Asghari, and M. Eeten, “Estimating the size of the iceberg from its tip,” in Proc. Workshop Econ. Inf. Secur. (WEIS), La Jolla, CA, USA, 2017.

[30] R. F. Engle and J. R. Russell, “Autoregressive conditional duration: A new model for irregularly spaced transaction data,” Econometrica, vol. 66, no. 5, pp. 1127–1162, 1998.

[31] N. Hautsch, Econometrics of Financial High-Frequency Data. Berlin, Germany: Springer-Verlag, 2011.

[32] P. Embrechts, C. Klüppelberg, and T. Mikosch, Modelling Extremal Events: For Insurance and Finance. Berlin, Germany: Springer, 1997.

[33] T. Bollerslev, J. Russell, and M. Watson, Volatility and Time Series Econometrics: Essays in Honor of Robert Engle. London, U.K.: Oxford Univ. Press, 2010.

XU et al.: MODELING AND PREDICTING CYBER HACKING BREACHES 2871

[34] R. B. Nelsen, An Introduction to Copulas. New York, NY, USA: Springer-Verlag, 2007.

[35] H. Joe, Dependence Modelling With Copulas. Boca Raton, FL, USA: CRC Press, 2014.

[36] J. D. Cryer and K.-S. Chan, Time Series Analysis With Applications in R. New York, NY, USA: Springer, 2008.

[37] B. Peter and D. Richard, Introduction to Time Series and Forecasting. New York, NY, USA: Springer-Verlag, 2002.

[38] P. J. Brockwell and R. A. Davis, Introduction to Time Series and Forecasting. New York, NY, USA: Springer-Verlag, 2016.

[39] D. J. Daley and D. Vere-Jones, An Introduction to the Theory of Point Processes, vol. 1, 2nd ed. New York, NY, USA: Springer-Verlag, 2002.

[40] M. Y. Zhang, J. R. Russell, and R. S. Tsay, "A nonlinear autoregressive conditional duration model with applications to financial transaction data," J. Econ., vol. 104, no. 1, pp. 179–207, 2001.

[41] L. Bauwens and P. Giot, "The logarithmic ACD model: An application to the bid-ask quote process of three NYSE stocks," Ann. Économie Stat., no. 60, pp. 117–149, Oct./Dec. 2000.

[42] L. Bauwens, P. Giot, J. Grammig, and D. Veredas, "A comparison of financial duration models via density forecasts," Int. J. Forecasting, vol. 20, no. 4, pp. 589–609, 2004.

[43] G. W. Corder and D. I. Foreman, Nonparametric Statistics: A Step-byStep Approach. Hoboken, NJ, USA: Wiley, 2014.

[44] P. R. Hansen and A. Lunde, "A forecast comparison of volatility models: Does anything beat a garch(1, 1)?" J. Appl. Econ., vol. 20, no. 7, pp. 873–889, 2005.

[45] S. I. Resnick, Heavy-Tail Phenomena: Probabilistic and Statistical Modelling. New York, NY, USA: Springer-Verlag, 2007.

[46] X. Zhao, C. Scarrott, L. Oxley, and M. Reale, "Extreme value modelling for forecasting market crisis impacts," Appl. Financial Econ., vol. 20, nos. 1–2, pp. 63–72, 2010.

[47] C. Scarrott, "Univariate extreme value mixture modelling," in Extreme Value Modelling and Risk Analysis: Methods and Applications, J. Yan and D. K. Dey, Eds. London, U.K.: Chapman & Hall, 2016, pp. 41–67.

[48] H. Joe, Multivariate Models and Dependence Concepts (Monographs on Statistics and Applied Probability), vol. 73. London, U.K.: Chapman & Hall, 1997.

[49] H. White, "Maximum likelihood estimation of misspecified models," Econometrica, J. Econ. Soc., vol. 50, no. 1, pp. 1–25, 1982.

[50] W. Huang and A. Prokhorov, “A goodness-of-fit test for copulas,” Econ. Rev., vol. 33, no. 7, pp. 751–771, 2014.

[51] W. Wang and M. T. Wells, “Model selection and semiparametric inference for bivariate failure-time data,” J. Amer. Statist. Assoc., vol. 95, no. 449, pp. 62–72, 2000.

[52] C. Genest, J.-F. Quessy, and B. Rémillard, “Goodness-of-fit procedures for copula models based on the probability integral transformation,” Scandin. J. Stat., vol. 33, no. 2, pp. 337–366, 2006.

[53] A. McNeil, R. Frey, and P. Embrechts, Quantitative Risk Management: Concepts, Techniques, and Tools. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[54] P. F. Christoffersen, “Evaluating interval forecasts,” Int. Econ. Rev., vol. 39, no. 4, pp. 841–862, 1998.

[55] R. F. Engle and S. Manganelli, “CAViaR: Conditional autoregressive value at risk by regression quantiles,” J. Bus. Econ. Stat., vol. 22, no. 4, pp. 367–381, 2004.

[56] P. M. Romer, “Increasing returns and long-run growth,” J. Political Econ., vol. 94, no. 5, pp. 1002–1037, 1986.

[57] G. M. Ljung and G. E. P. Box, “On a measure of lack of fit in time series models,” Biometrika, vol. 65, no. 2, pp. 297–303, 1978.

[58] G. R. Shorack and J. A. Wellner, Empirical Processes With Applications to Statistics. Philadelphia, PA, USA: SIAM, 1986.

[59] M. A. Stephens, “Tests based on EDF statistics,” in Goodness-of-Fit Techniques, R. B. d’Agostino and M. A. Stephens, Eds. New York, NY, USA: Marcel Dekker, 1986, pp. 97–193.