National College of Ireland

Configuration Manual

MSc Internship
Cyber Security

Brian Redmond
Student ID: 17137250

School of Computing
National College of Ireland

Supervisor: Mr Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Brian Redmond

Student ID: 17137250

Programme: Cyber Security **Year:** 2018

Module: MSc Internship

Lecturer: Mr Ross Spelman

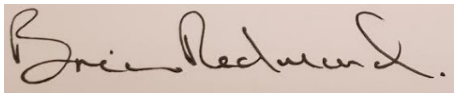
Submission Due Date: 17/8/2020

Project Title: How does enhancing user activity features in Machine Learning algorithms offer a pre-emptive counter measure to detecting corporate insider threats?

Word Count: **687** **Page Count: 56**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:



Date: 14/8/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not	□

sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Brian Redmond
17137250

This research has developed an application to run and perform a series of experiments using the spark Machine Learning library. It uses Java 8 and MySQL as the database. IntelliJ IDEA 2017v3.4. All must be installed to replicate the application setup. The IntelliJ IDE environment can be replaced with any IDE such as Eclipse or Netbeans. Equally the database is agnostic and any relation database could be used.

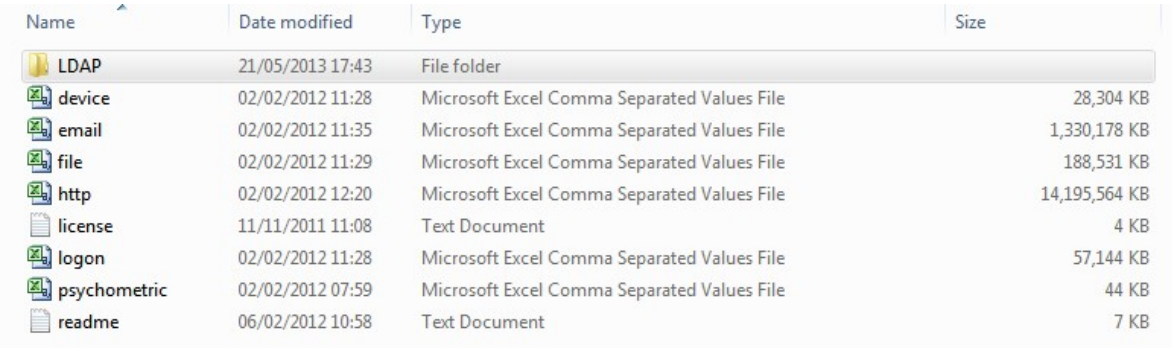
1 Database Setup

1. Install the MySQL database engine v.8.0.21 with default settings.
    a. Link: https://dev.mysql.com/downloads/installer/
    b. MySQL can also be installed though XAMPP, a standard cross platform framework.
        i. Link: https://www.apachefriends.org/index.html

2 Data Load

The data for this application consists of a base set of csv file supplied by CERT Carnegie Mellon University Software Engineering Institute. This dataset is loaded into the database through a series of file loaders.

1. Data files version r4.2 are available from Link: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099 as a zip file.
2. Unzipped files consist of 6 csv file, licensing details and a directory containing LDAP details with a csv file for each month.



Name	Date modified	Type	Size
LDAP	21/05/2013 17:43	File folder	
device	02/02/2012 11:28	Microsoft Excel Comma Separated Values File	28,304 KB
email	02/02/2012 11:35	Microsoft Excel Comma Separated Values File	1,330,178 KB
file	02/02/2012 11:29	Microsoft Excel Comma Separated Values File	188,531 KB
http	02/02/2012 12:20	Microsoft Excel Comma Separated Values File	14,195,564 KB
license	11/11/2011 11:08	Text Document	4 KB
logon	02/02/2012 11:28	Microsoft Excel Comma Separated Values File	57,144 KB
psychometric	02/02/2012 07:59	Microsoft Excel Comma Separated Values File	44 KB
readme	06/02/2012 10:58	Text Document	7 KB

3. Total uncompressed size is 15 gigabytes.
4. Name the data directory to match the loaders files from Appendix 2.

5. Use a third party MySQL administration tool for executing the scripts.
    a. Link: https://www.heidisql.com/
6. This research paper used HeidiSQL version 9.5.
7. First step is to load the database schema for the application.
    a. Database schema can be seen in Appendix 1.
    b. Load the following script file (Threat Database Schema.sql) though the standard query analyser screen.
    Script can be found in \Development5_CERT\Database Schema from the source code archive.

8. Load the following script files though the standard query analyser screen listed in appendix 2.
    a. LoadCertDevice.sql
    b. LoadCertEmail.sql
    c. LoadCertFile.sql
    d. LoadCertHttp.sql
    e. LoadCertLdap.sql
    f. LoadCertLogon.sql

    Script loaders are available from \Development5_CERT\SQLDataLoaders in the source code archive.

    Once complete the following highlighted tables in figure 2.1 should exist.

Host: 127.0.0.1 | Database: cert | Query

Name	Rows	Size	Created	Updated	Engine	Comment	Type
activity	5	16.0 KiB	2020-08-09 14:18:05		InnoDB		Table
averagelogontime	4	16.0 KiB	2019-07-19 08:45:41		InnoDB		Table
certdata	16,964	5.9 MiB	2020-07-05 11:51:52		InnoDB		Table
cfedata	87,286	18.1 MiB	2020-08-09 13:51:25		InnoDB		Table
dailyaverageactivity	4	16.0 KiB	2020-07-05 09:50:44		InnoDB		Table
departments	67	16.0 KiB	2020-08-09 14:18:05		InnoDB		Table
device	388,799	44.1 MiB	2020-03-08 10:54:06		InnoDB		Table
deviceaccess	0	32.0 KiB	2020-07-19 11:25:28		InnoDB		Table
devicethreat	537	96.0 KiB	2020-08-09 14:15:52		InnoDB		Table
email	2,355,265	427.9 MiB	2020-08-05 16:48:08		InnoDB		Table
emailthreat	766	176.0 KiB	2020-08-09 14:15:52		InnoDB		Table
features	59,559	30.0 MiB	2020-08-06 17:26:13		InnoDB		Table
featurescfe	3,631	416.0 KiB	2020-08-06 17:26:32		InnoDB		Table
featuresrecursive	81,395	9.5 MiB	2020-08-05 08:18:29		InnoDB		Table
file	443,409	38.6 MiB	2020-08-05 16:48:56		InnoDB		Table
filethreat	1,734	176.0 KiB	2020-08-09 14:15:52		InnoDB		Table
http	25,042,860	4.3 GiB	2020-07-21 07:29:47		InnoDB		Table
httpthreat	7,777	1.7 MiB	2020-08-09 14:15:52		InnoDB		Table
ldap	16,574	3.5 MiB	2019-07-19 08:51:26		InnoDB		Table
ldapmovement	16,743	1.5 MiB	2019-07-19 08:45:43		InnoDB		Table
ldapthreat	36	16.0 KiB	2020-08-09 14:15:52		InnoDB		Table
log	1,826	336.0 KiB	2020-08-08 12:16:15		InnoDB		Table
logon	851,167	67.6 MiB	2020-08-05 16:47:11		InnoDB		Table
logonthreat	1,150	128.0 KiB	2020-08-09 14:15:52		InnoDB		Table
logresults	1,347	320.0 KiB	2020-06-27 12:11:30		InnoDB		Table
probability	15	16.0 KiB	2020-08-13 12:55:46		InnoDB		Table
probabilitydevice	8	16.0 KiB	2020-08-13 12:55:46		InnoDB		Table
probabilityemail	14	16.0 KiB	2020-08-13 12:55:46		InnoDB		Table
scheduler	12,073	400.0 KiB	2020-07-20 18:39:39		InnoDB		Table
schedulerrun	1	16.0 KiB	2020-08-04 11:41:04		InnoDB		Table
user	13	32.0 KiB	2020-08-09 14:17:36		InnoDB		Table
userdepartments	1,002	176.0 KiB	2020-08-13 12:56:54		InnoDB		Table

Figure 2.1 Database tables, record number and transaction size.

9. The next stage of the data load process is building the aggregate data, features and look tables.

Load the series of SQL scripts to build up the data structure listed in appendix 3.
Each script can be found in the Development5_CERT\SQLDataLoaders directory of the source code archive.
This series of scripts will:

- Aggregate the data into single data tables.
- Create all the synthetic threat users.
- Set all the default risk settings.
- Build up the probability structure.
- Create all data for the look up tables required.
- Set all the normal and threshold values for user records.

Load each script file in sequence.

a. BuildFeatureUser.sql
b. BuildThreat.sql
c. BuildFeature.sql
d. BuildFeatureTables.sql
e. BuildSetThreatValues.sql

10. Data integrity checks.
    With the volume of data involved within the application and the calculation of probability and features involved, it is key to perform a number of data integrity checks after the data is loaded. These checks were automated with the outcome requiring to be manually checked.

    Checks can be found in SQL script DataIntegrityChecks.sql in the Development5_CERT\SQLDataLoaders directory of the source code archive.

    This completes the database setup and loading of data. Data should exist as laid out in appendix 4.

3 Apache Spark

Apache Spark is an open source Machine Learning framework.

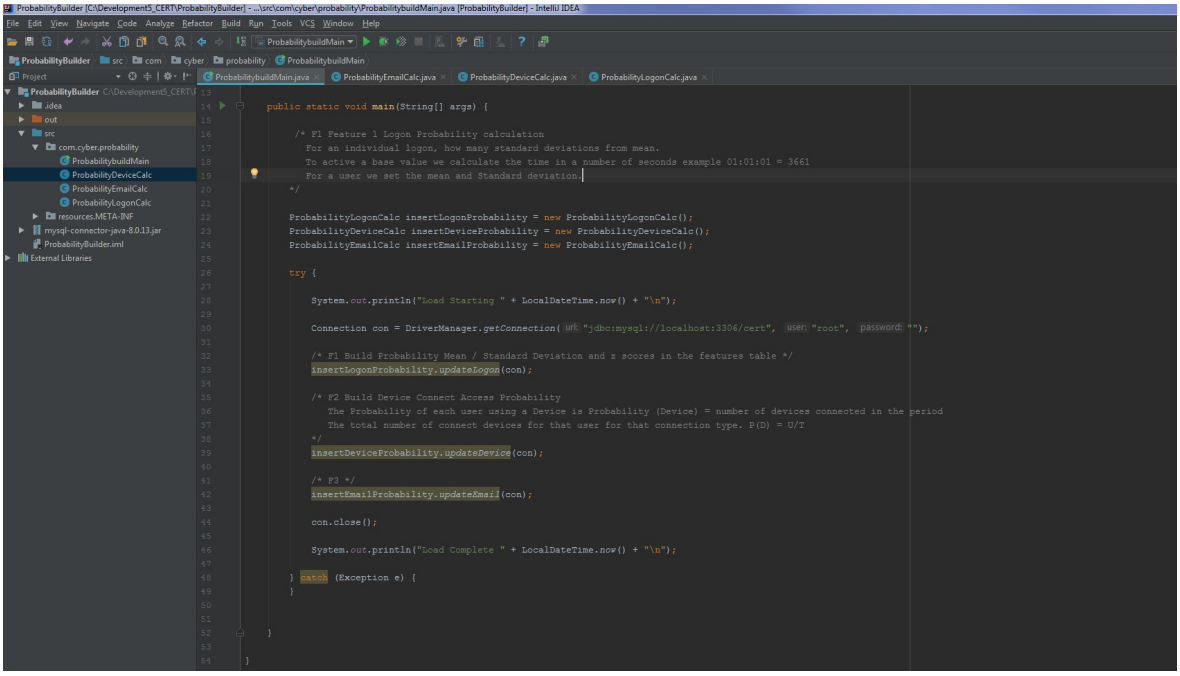
1. Install Java 8 on the PC / Server.
    a. Link:https://www.oracle.com/ie/java/technologies/javase/javase-jdk8-downloads.html
2. Download Apache v 2.3.0and extract the zip file into a \Spark directory
   Setup a SPARK_HOME environment variable

   Once Spark has been installed libraries will be available for integration within our java application.

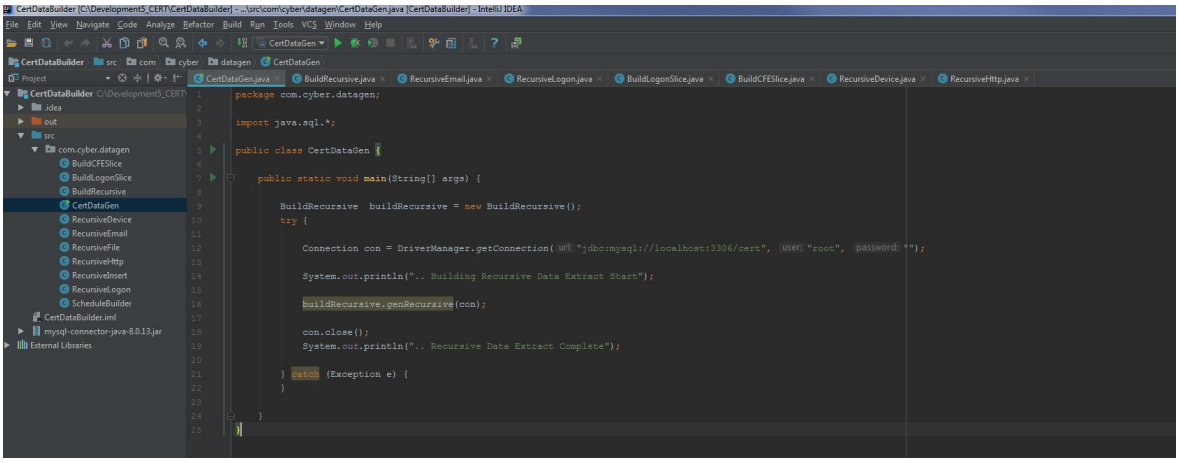
4 Application code

The application consists of 3 core modules split into separate java applications.

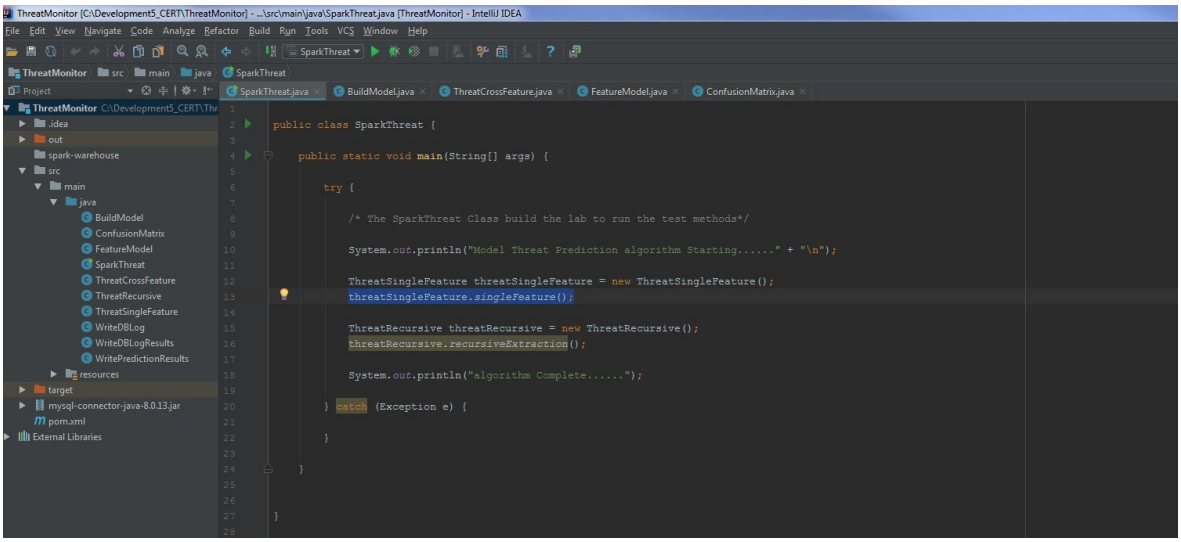
1. **ProbabilityBuilder** calculates the probability Z-Scores for the single feature algorithm for scenarios (Logon, email, device). Source code is available in the C:\Development5_CERT\ProbabilityBuilder listed in appendix 5.
   Include a MySQL driver as a library to the source.



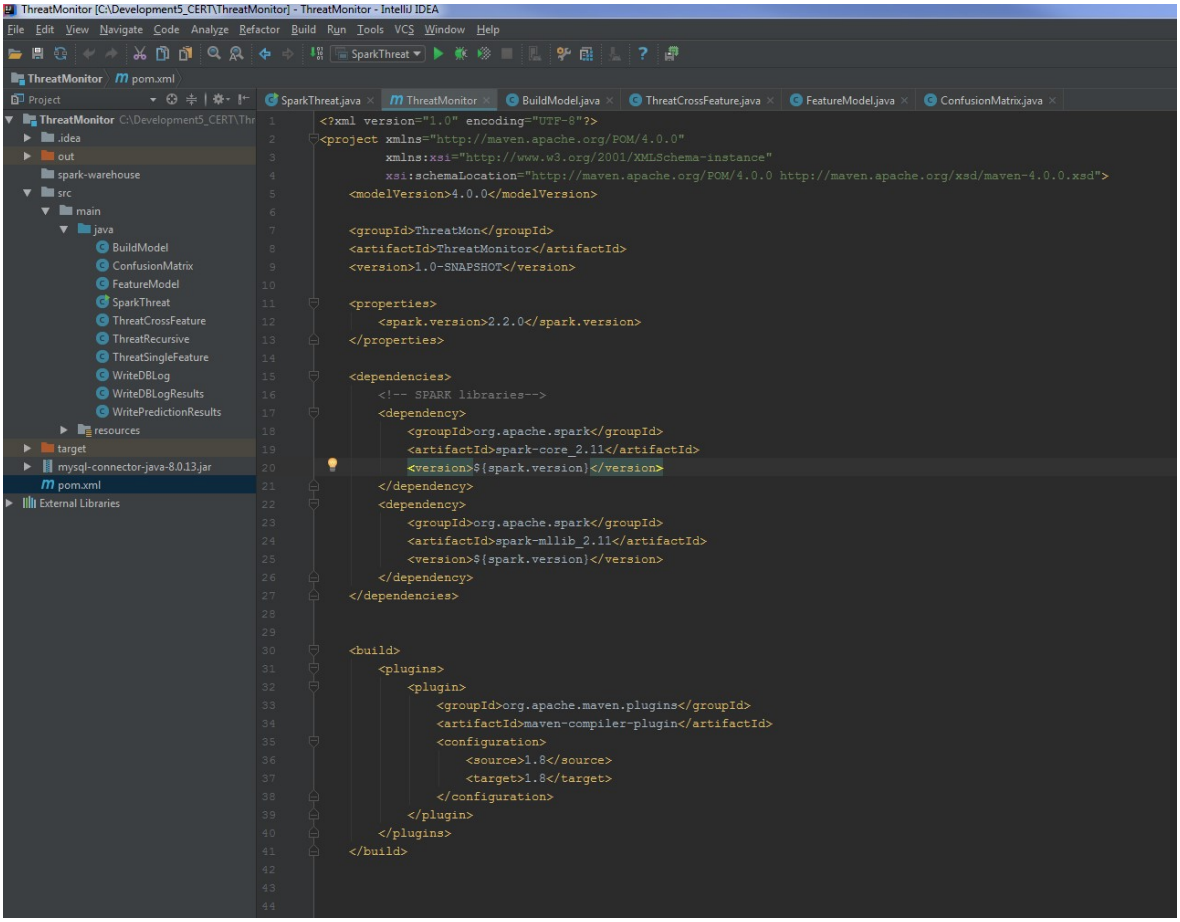
2. **CertBuilder** calculates the Z-Scores for the recursive algorithm. Source code is available in the C:\Development5_CERT\CertDataBuilder listed in appendix 6.



3. **ThreatMonitor** runs the Machine Learning model.
   Source code is available in the Development5_CERT\ThreatMonitor\ listed in appendix 6.



Maven has to be specifically configured to reference the Spark core and mllib libraries.



Appendix 1 Database Schema script Loader file

```
-- --------------------------------------------------------
-- Host:                         127.0.0.1
-- Server version:               10.1.37-MariaDB - mariadb.org binary distribution
-- Server OS:                    Win32
-- HeidiSQL Version:             9.5.0.5196
-- --------------------------------------------------------

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!50503 SET NAMES utf8mb4 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;


-- Dumping database structure for cert
CREATE DATABASE IF NOT EXISTS `cert` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `cert`;

-- Dumping structure for table cert.activity
CREATE TABLE IF NOT EXISTS `activity` (
  `activityId` int(11) NOT NULL AUTO_INCREMENT,
  `activity` varchar(50) NOT NULL DEFAULT '0',
  PRIMARY KEY (`activityId`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.averagelogontime
CREATE TABLE IF NOT EXISTS `averagelogontime` (
  `user` varchar(50) NOT NULL,
  `averageTime` time DEFAULT NULL,
  PRIMARY KEY (`user`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.certdata
CREATE TABLE IF NOT EXISTS `certdata` (
  `certkey` int(11) NOT NULL AUTO_INCREMENT,
  `certId` varchar(50) DEFAULT NULL,
  `certDate` datetime DEFAULT NULL,
  `certTime` time DEFAULT NULL,
  `certUser` varchar(50) DEFAULT NULL,
  `empNo` int(11) NOT NULL DEFAULT '0',
  `certPc` varchar(50) DEFAULT NULL,
  `certActivity` varchar(50) DEFAULT NULL,
  `certFileName` varchar(150) DEFAULT NULL,
  `certDepartment` int(11) DEFAULT NULL,
  `certEmailSize` int(11) DEFAULT '0',
  `certEmailAttachments` int(11) DEFAULT '0',
  `certUrl` varchar(250) DEFAULT NULL,
  `certYear` int(11) DEFAULT '0',
  `certMonth` int(11) DEFAULT '0',
  `featureDayOfWeek` int(11) DEFAULT '0',
  `featureTime` time NOT NULL DEFAULT '00:00:00',
  `featureActivity` int(11) DEFAULT NULL,
  `featurePCAccess` int(11) NOT NULL DEFAULT '0',
```

```
  `featureDailyAverageActivity` double(11,2) NOT NULL DEFAULT '0.00',
  `featureUserGone` int(11) NOT NULL DEFAULT '0',
  `fcalcProbabilityLogon` double NOT NULL DEFAULT '0',
  `fcalcProbabilityDevice` double(15,4) NOT NULL DEFAULT '0.0000',
  `fcalcProbabilityDay` double(15,4) NOT NULL DEFAULT '0.0000',
  `fcalcUnusualHours` int(11) NOT NULL DEFAULT '0',
  `Risk` varchar(50) NOT NULL,
  `updateFrom` char(50) NOT NULL DEFAULT 'S',
  PRIMARY KEY (`certkey`),
  KEY `certUser` (`certUser`)
) ENGINE=InnoDB AUTO_INCREMENT=19916 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.cfedata
CREATE TABLE IF NOT EXISTS `cfedata` (
  `cfeId` int(11) NOT NULL AUTO_INCREMENT,
  `user` varchar(50) DEFAULT NULL,
  `empNo` int(11) DEFAULT NULL,
  `dateActivity` datetime DEFAULT NULL,
  `device` varchar(7) DEFAULT NULL,
  `activity` varchar(25) DEFAULT NULL,
  `activityId` int(11) DEFAULT '0',
  `emailSize` varchar(25) DEFAULT NULL,
  `url` varchar(255) DEFAULT NULL,
  `urlRating` int(11) DEFAULT '0',
  `risk` varchar(15) DEFAULT NULL,
  PRIMARY KEY (`cfeId`),
  KEY `dateActivity` (`dateActivity`),
  KEY `user` (`user`),
  KEY `activity` (`activity`)
) ENGINE=InnoDB AUTO_INCREMENT=100210 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.dailyaverageactivity
CREATE TABLE IF NOT EXISTS `dailyaverageactivity` (
  `user` varchar(50) NOT NULL,
  `averageActivity` double NOT NULL,
  PRIMARY KEY (`user`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.departments
CREATE TABLE IF NOT EXISTS `departments` (
  `departmentId` int(11) NOT NULL AUTO_INCREMENT,
  `department` varchar(150) DEFAULT '0',
  PRIMARY KEY (`departmentId`)
) ENGINE=InnoDB AUTO_INCREMENT=68 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.device
CREATE TABLE IF NOT EXISTS `device` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) DEFAULT '0',
  `pc` varchar(50) DEFAULT '0',
  `activity` varchar(50) DEFAULT '0',
  PRIMARY KEY (`key`),
  KEY `user` (`user`)
) ENGINE=InnoDB AUTO_INCREMENT=405381 DEFAULT CHARSET=latin1;
```

```
-- Data exporting was unselected.
-- Dumping structure for table cert.deviceaccess
CREATE TABLE IF NOT EXISTS `deviceaccess` (
  `user` varchar(50) DEFAULT NULL,
  `device` varchar(10) DEFAULT NULL,
  `deviceAccess` int(11) DEFAULT NULL,
  KEY `user` (`user`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.devicethreat
CREATE TABLE IF NOT EXISTS `devicethreat` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) DEFAULT '0',
  `pc` varchar(50) DEFAULT '0',
  `activity` varchar(50) DEFAULT '0',
  PRIMARY KEY (`key`),
  KEY `user` (`user`)
) ENGINE=InnoDB AUTO_INCREMENT=782 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.email
CREATE TABLE IF NOT EXISTS `email` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) NOT NULL DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) NOT NULL DEFAULT '0',
  `pc` varchar(50) NOT NULL DEFAULT '0',
  `eto` varchar(250) NOT NULL DEFAULT '0',
  `cc` varchar(250) NOT NULL DEFAULT '0',
  `bcc` varchar(250) NOT NULL DEFAULT '0',
  `efrom` varchar(250) NOT NULL DEFAULT '0',
  `size` int(11) NOT NULL DEFAULT '0',
  `attachments` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`key`),
  KEY `user` (`user`)
) ENGINE=InnoDB AUTO_INCREMENT=2629980 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.emailthreat
CREATE TABLE IF NOT EXISTS `emailthreat` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) NOT NULL DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) NOT NULL DEFAULT '0',
  `pc` varchar(50) NOT NULL DEFAULT '0',
  `eto` varchar(250) NOT NULL DEFAULT '0',
  `cc` varchar(250) NOT NULL DEFAULT '0',
  `bcc` varchar(250) NOT NULL DEFAULT '0',
  `efrom` varchar(250) NOT NULL DEFAULT '0',
  `size` int(11) NOT NULL DEFAULT '0',
  `attachments` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`key`)
) ENGINE=InnoDB AUTO_INCREMENT=1027 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.features
```

```
CREATE TABLE IF NOT EXISTS `features` (
  `featureKey` int(11) NOT NULL AUTO_INCREMENT,
  `featureId` varchar(2) DEFAULT NULL,
  `id` varchar(50) DEFAULT NULL,
  `user` varchar(15) DEFAULT NULL,
  `empNo` int(11) DEFAULT NULL,
  `departmentId` int(11) DEFAULT NULL,
  `dateActivity` datetime DEFAULT NULL,
  `pcAccessed` varchar(15) DEFAULT NULL,
  `url` varchar(250) DEFAULT '',
  `activity` varchar(50) DEFAULT NULL,
  `featureActivity` int(11) DEFAULT NULL,
  `featureDayofWeek` int(11) DEFAULT NULL,
  `featurePc` int(11) DEFAULT NULL,
  `featureProbabilityLogon` double DEFAULT '0',
  `featureConnects` varchar(15) DEFAULT '0',
  `featureProbabilityConnect` double DEFAULT '0',
  `featureEmails` int(11) DEFAULT '0',
  `featureEmailSize` int(11) DEFAULT '0',
  `featureProbabilityEmail` double DEFAULT '0',
  `featureProbabilityEmailSize` double DEFAULT '0',
  `featureUrlRating` int(11) DEFAULT '0',
  `risk` varchar(15) DEFAULT NULL,
  PRIMARY KEY (`featureKey`),
  KEY `dateActivity` (`dateActivity`)
) ENGINE=InnoDB AUTO_INCREMENT=96278 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.featurescfe
CREATE TABLE IF NOT EXISTS `featurescfe` (
  `featurekey` int(11) NOT NULL AUTO_INCREMENT,
  `user` varchar(50) NOT NULL,
  `empNo` int(11) NOT NULL DEFAULT '0',
  `dateActivity` datetime NOT NULL,
  `activity` varchar(20) NOT NULL,
  `featureDevice` int(11) DEFAULT '0',
  `featureActivity` int(11) NOT NULL DEFAULT '0',
  `featureDayofWeek` int(11) NOT NULL DEFAULT '0',
  `featureProbabilityLogon` double DEFAULT '0',
  `featureProbabilityDevice` double DEFAULT '0',
  `featureProbabilityEmail` double DEFAULT '0',
  `featureProbabilityEmailSize` double DEFAULT '0',
  `featureUrlRating` int(11) DEFAULT '0',
  `risk` varchar(15) DEFAULT '',
  PRIMARY KEY (`featurekey`)
) ENGINE=InnoDB AUTO_INCREMENT=9274 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.featuresrecursive
CREATE TABLE IF NOT EXISTS `featuresrecursive` (
  `featurekey` int(11) NOT NULL AUTO_INCREMENT,
  `user` varchar(50) NOT NULL,
  `empNo` int(11) NOT NULL DEFAULT '0',
  `dateActivity` datetime NOT NULL,
  `activity` varchar(20) NOT NULL,
  `featureDevice` int(11) DEFAULT '0',
  `featureActivity` int(11) NOT NULL DEFAULT '0',
  `featureDayofWeek` int(11) NOT NULL DEFAULT '0',
  `featureProbabilityLogon` double DEFAULT '0',
  `featureProbabilityDevice` double DEFAULT '0',
```

```
  `featureProbabilityEmail` double DEFAULT '0',
  `featureProbabilityEmailSize` double DEFAULT '0',
  `featureProbabilityFile` double DEFAULT '0',
  `featureUrlRating` int(11) DEFAULT '0',
  `risk` varchar(15) DEFAULT '',
  PRIMARY KEY (`featurekey`)
) ENGINE=InnoDB AUTO_INCREMENT=465644 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.file
CREATE TABLE IF NOT EXISTS `file` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) NOT NULL DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) DEFAULT NULL,
  `pc` varchar(50) DEFAULT NULL,
  `filename` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`key`),
  KEY `user` (`user`)
) ENGINE=InnoDB AUTO_INCREMENT=445582 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.filethreat
CREATE TABLE IF NOT EXISTS `filethreat` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) NOT NULL DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) DEFAULT NULL,
  `pc` varchar(50) DEFAULT NULL,
  `filename` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`key`)
) ENGINE=InnoDB AUTO_INCREMENT=2181 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.http
CREATE TABLE IF NOT EXISTS `http` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) NOT NULL DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) NOT NULL DEFAULT '0',
  `pc` varchar(50) NOT NULL DEFAULT '0',
  `url` varchar(500) NOT NULL DEFAULT '0',
  PRIMARY KEY (`key`),
  KEY `user` (`user`)
) ENGINE=InnoDB AUTO_INCREMENT=28434424 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.httpthreat
CREATE TABLE IF NOT EXISTS `httpthreat` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) NOT NULL DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) NOT NULL DEFAULT '0',
  `pc` varchar(50) NOT NULL DEFAULT '0',
  `url` varchar(500) NOT NULL DEFAULT '0',
  PRIMARY KEY (`key`),
  KEY `user` (`user`)
) ENGINE=InnoDB AUTO_INCREMENT=11653 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
```

```
-- Dumping structure for table cert.ldap
CREATE TABLE IF NOT EXISTS `ldap` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `employee_name` varchar(150) NOT NULL DEFAULT '0',
  `user_id` varchar(50) NOT NULL DEFAULT '0',
  `email` varchar(150) NOT NULL DEFAULT '0',
  `role` varchar(50) NOT NULL DEFAULT '0',
  `business_unit` int(11) NOT NULL DEFAULT '0',
  `functional_unit` varchar(100) NOT NULL DEFAULT '0',
  `department` varchar(100) NOT NULL DEFAULT '0',
  `team` varchar(100) NOT NULL DEFAULT '0',
  `supervisor` varchar(100) NOT NULL DEFAULT '0',
  `ldapYear` int(11) NOT NULL DEFAULT '0',
  `ldapMonth` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`key`)
) ENGINE=InnoDB AUTO_INCREMENT=18237 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.ldapmovement
CREATE TABLE IF NOT EXISTS `ldapmovement` (
  `user` varchar(50) NOT NULL,
  `ldapyear` int(11) NOT NULL,
  `ldapmonth` int(11) NOT NULL,
  `ldapflow` int(11) DEFAULT '0',
  PRIMARY KEY (`user`,`ldapyear`,`ldapmonth`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.ldapthreat
CREATE TABLE IF NOT EXISTS `ldapthreat` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `employee_name` varchar(150) NOT NULL DEFAULT '0',
  `user_id` varchar(50) NOT NULL DEFAULT '0',
  `email` varchar(150) NOT NULL DEFAULT '0',
  `role` varchar(50) NOT NULL DEFAULT '0',
  `business_unit` int(11) NOT NULL DEFAULT '0',
  `functional_unit` varchar(100) NOT NULL DEFAULT '0',
  `department` varchar(100) NOT NULL DEFAULT '0',
  `team` varchar(100) NOT NULL DEFAULT '0',
  `supervisor` varchar(100) NOT NULL DEFAULT '0',
  `ldapYear` int(11) NOT NULL DEFAULT '0',
  `ldapMonth` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`key`)
) ENGINE=InnoDB AUTO_INCREMENT=50 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.log
CREATE TABLE IF NOT EXISTS `log` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `logId` int(11) DEFAULT NULL,
  `feature` varchar(3) DEFAULT NULL,
  `startTime` varchar(20) DEFAULT NULL,
  `finishTime` varchar(20) DEFAULT NULL,
  `sampleSize` int(11) DEFAULT NULL,
  `trained` int(11) DEFAULT NULL,
  `predictionSize` int(11) DEFAULT NULL,
  `testError` decimal(10,8) DEFAULT NULL,
  `accuracy` decimal(10,8) DEFAULT NULL,
  `ROC` decimal(10,8) DEFAULT '0.00000000',
  `sliceTime` varchar(20) DEFAULT NULL,
```

```
  `logFile` varchar(150) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5473 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.logon
CREATE TABLE IF NOT EXISTS `logon` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) NOT NULL DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) NOT NULL DEFAULT '0',
  `pc` varchar(50) NOT NULL DEFAULT '0',
  `activity` varchar(50) NOT NULL DEFAULT '0',
  PRIMARY KEY (`key`),
  KEY `user` (`user`)
) ENGINE=InnoDB AUTO_INCREMENT=854860 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.logonthreat
CREATE TABLE IF NOT EXISTS `logonthreat` (
  `key` int(11) NOT NULL AUTO_INCREMENT,
  `id` varchar(50) NOT NULL DEFAULT '0',
  `date` datetime DEFAULT NULL,
  `user` varchar(50) NOT NULL DEFAULT '0',
  `pc` varchar(50) NOT NULL DEFAULT '0',
  `activity` varchar(50) NOT NULL DEFAULT '0',
  PRIMARY KEY (`key`)
) ENGINE=InnoDB AUTO_INCREMENT=1213 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.logresults
CREATE TABLE IF NOT EXISTS `logresults` (
  `logKey` int(11) NOT NULL AUTO_INCREMENT,
  `logId` int(11) DEFAULT NULL,
  `id` int(11) DEFAULT NULL,
  `label` double DEFAULT NULL,
  `prediction` double DEFAULT NULL,
  `features` varchar(250) DEFAULT NULL,
  `predictions` varchar(250) DEFAULT NULL,
  `probabilities` varchar(250) DEFAULT NULL,
  `runDate` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`logKey`)
) ENGINE=InnoDB AUTO_INCREMENT=5179 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.probability
CREATE TABLE IF NOT EXISTS `probability` (
  `user` varchar(50) NOT NULL,
  `logonMu` double DEFAULT '0',
  `logonSd` double DEFAULT '0',
  PRIMARY KEY (`user`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.probabilitydevice
CREATE TABLE IF NOT EXISTS `probabilitydevice` (
  `user` varchar(50) DEFAULT NULL,
  `deviceAccessed` varchar(10) DEFAULT NULL,
  `deviceMu` double DEFAULT NULL,
  `deviceSd` double DEFAULT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.probabilityemail
CREATE TABLE IF NOT EXISTS `probabilityemail` (
  `user` varchar(50) DEFAULT NULL,
  `emailMu` double DEFAULT '0',
  `emailSd` double DEFAULT '0',
  `emailSizeMu` double DEFAULT '0',
  `emailSizeSd` double DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.scheduler
CREATE TABLE IF NOT EXISTS `scheduler` (
  `sliceTime` datetime DEFAULT NULL,
  KEY `sliceTime` (`sliceTime`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.schedulerrun
CREATE TABLE IF NOT EXISTS `schedulerrun` (
  `startSlice` datetime DEFAULT NULL,
  `endSlice` datetime DEFAULT NULL,
  `intervalDays` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.user
CREATE TABLE IF NOT EXISTS `user` (
  `empNo` int(11) NOT NULL AUTO_INCREMENT,
  `user` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`empNo`),
  KEY `user` (`user`)
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
-- Dumping structure for table cert.userdepartments
CREATE TABLE IF NOT EXISTS `userdepartments` (
  `User` varchar(50) DEFAULT NULL,
  `Department` varchar(150) DEFAULT NULL,
  `departmentID` int(11) DEFAULT NULL,
  KEY `User` (`User`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Data exporting was unselected.
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

Appendix 2 Database script Loader files

```
delete from device;
ALTER TABLE device AUTO_INCREMENT = 1;

LOAD DATA INFILE 'C:/Dataset2/r4.2/device.csv'
   INTO TABLE device
   FIELDS TERMINATED BY ','
```

```
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (id, @var1,user,pc,activity)
   SET date  = STR_TO_DATE(@var1,'%m/%d/%Y %H:%i:%s');

delete from email;
ALTER TABLE email AUTO_INCREMENT = 1;

LOAD DATA INFILE 'C:/Dataset2/r4.2/email.csv'
   INTO TABLE email
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (id, @var1,user,pc,eto,cc,bcc,efrom,size,attachments)
   SET date  = STR_TO_DATE(@var1,'%m/%d/%Y %H:%i:%s');

delete from file;
ALTER TABLE file AUTO_INCREMENT = 1;

LOAD DATA INFILE 'C:/Dataset2/r4.2/file.csv'
   INTO TABLE file
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (id, @var1,user,pc,filename)
   SET date  = STR_TO_DATE(@var1,'%m/%d/%Y %H:%i:%s');

delete from http;
ALTER TABLE http AUTO_INCREMENT = 1;

LOAD DATA INFILE 'C:/Dataset2/r4.2/http.csv'
   INTO TABLE http
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (id, @var1,user,pc,url)
   SET date  = STR_TO_DATE(@var1,'%m/%d/%Y %H:%i:%s');


delete from logon;
ALTER TABLE logon AUTO_INCREMENT = 1;


LOAD DATA INFILE 'C:/Dataset2/r4.2/logon.csv'
   INTO TABLE logon
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (id, @var1,user,pc,activity)
   SET date  = STR_TO_DATE(@var1,'%m/%d/%Y %H:%i:%s');

delete from ldap;
ALTER TABLE ldap AUTO_INCREMENT = 1;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2009-12.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
```

```
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2009, ldapMonth = 12;

LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-01.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2010, ldapMonth = 01;

LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-02.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2010, ldapMonth = 02;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-03.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2010, ldapMonth = 03;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-04.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2010, ldapMonth = 04;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-05.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2010, ldapMonth = 05;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-06.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2010, ldapMonth = 06;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-07.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
```

```
    IGNORE 1 LINES
    (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
    set ldapYear = 2010, ldapMonth = 07;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-08.csv'
    INTO TABLE ldap
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
    set ldapYear = 2010, ldapMonth = 08;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-09.csv'
    INTO TABLE ldap
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
    set ldapYear = 2010, ldapMonth = 09;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-10.csv'
    INTO TABLE ldap
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
    set ldapYear = 2010, ldapMonth = 10;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-11.csv'
    INTO TABLE ldap
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
    set ldapYear = 2010, ldapMonth = 11;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2010-12.csv'
    INTO TABLE ldap
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
    set ldapYear = 2010, ldapMonth = 12;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2011-01.csv'
    INTO TABLE ldap
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
    set ldapYear = 2011, ldapMonth = 01;
```

```
LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2011-02.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2011, ldapMonth = 02;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2011-03.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2011, ldapMonth = 03;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2011-04.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2011, ldapMonth = 04;


LOAD DATA INFILE 'C:/Dataset2/r4.2/LDAP/2011-05.csv'
   INTO TABLE ldap
   FIELDS TERMINATED BY ','
   LINES TERMINATED BY '\r\n'
   IGNORE 1 LINES
   (employee_name,user_id,email,role,business_unit,functional_unit,department,team,supervisor)
   set ldapYear = 2011, ldapMonth = 05;
```

Appendix 3 Database Data Loader files

BuildFeatureUser.sql

```
/* Builds up the aggregated Data tables from all data except LDAP which is seperate
 * A sample for testing and the full data set can be built
 */

/* Clean feature tables, set auto increment key fields back to zero */
delete from features;
ALTER TABLE features AUTO_INCREMENT = 1;

delete from cfedata;
ALTER TABLE cfedata AUTO_INCREMENT = 1;

delete from featurescfe;
ALTER TABLE featurescfe AUTO_INCREMENT = 1;

/* Build the test set of data first */
/* F1 Logon Time - single feature only */
insert into features (featureId,id,user,dateActivity,pcAccessed,activity,featurePc)
select 'F1',id,user,date,pc,activity,substring(pc,4,4) from logon
```

```
where user in ('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580') and
activity = 'Logon';

/*F2 Connect */
insert into features (featureId,user,dateActivity,pcAccessed,activity,featurePc,featureConnects)
select 'F2', user, DATE_FORMAT(date, '%Y-%m-%d') , pc, activity, substring(pc,4,4), count(pc) from device
where user in ('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580')  and
activity = 'Connect'
group by 'F2', user, DATE_FORMAT(date, '%Y-%m-%d') , pc, activity, substring(pc,4,4)
order by user , DATE_FORMAT(date, '%Y-%m-%d'), pc;

/* Email number and Size */
insert into features (featureId,user,dateActivity,activity,featureEmails,featureEmailSize)
select 'F3', user, DATE_FORMAT(date, '%Y-%m-%d') , 'email' as activity, count(user), sum(size) as emailsize
from email
where user in ('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580')
group by 'F3', user, DATE_FORMAT(date, '%Y-%m-%d') , activity
order by user , DATE_FORMAT(date, '%Y-%m-%d');

/* URL Rating */
insert into features (featureId,id,user,dateActivity,pcAccessed,activity,featurePc,url)
select 'F4',id,user,date,pc,'http' as activity,substring(pc,4,4),url from http
where user in ('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580');

/* Set the Default Risk to Normal */
update features set risk = 'Normal';

/* Cross Feature Extraction
   Builds up sample data 10 users from a population of 1000*/
insert into cfedata(user,dateActivity, device, activity, emailSize, url, risk)
select user, date, pc, activity, 0 as emalSize, '' as url, 'Normal' from logon where activity = 'Logon' and user in
('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580')
UNION
select user, date, pc, activity, 0 as emalSize, '' as url, 'Normal' from device where activity = 'Connect' and user in
('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580')
UNION
select user, date, pc, 'email' as activity, size as emalSize, '' as url, 'Normal' from email where user in
('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580')
UNION
select user, date, pc, 'http' as activity, 0 as emalSize, url, 'Normal' from http where user in
('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580')
UNION
select user, date, pc, 'file' as activity, 0 as emalSize, '' as url, 'Normal' from file where user in
('AAM0658','AAL0706','TAP0551','JJB0700','RAR0725','YJV0699','HXL0968','MDH0580');

/* Replicate the loaders for all records
LDAP    email           File access     Device access   logon   Http
16,743  2,629,979       445,581         405,380                 854,859 23,434,423
*** runtime total 5.15 hours

F1 Logon Time
insert into features (featureId,id,user,dateActivity,pcAccessed,activity,featurePc)
select 'F1',id,user,date,pc,activity,substring(pc,4,4) from logon;

F2 Connect
 We are only interested in the connect, we leave out the disconnects
insert into features (featureId,user,dateActivity,pcAccessed,activity,featurePc,featureConnects)
select 'F2', user, DATE_FORMAT(date, '%Y-%m-%d') , pc, activity, substring(pc,4,4), count(pc) from device
where activity = 'Connect'
group by 'F2', user, DATE_FORMAT(date, '%Y-%m-%d') , pc, activity, substring(pc,4,4)
```

```
order by user , DATE_FORMAT(date, '%Y-%m-%d'), pc;

Email number and Size
insert into features (featureId,user,dateActivity,activity,featureEmails,featureEmailSize)
select 'F3', user, DATE_FORMAT(date, '%Y-%m-%d') , 'email' as activity, count(user), sum(size) as emailsize
from email )
group by 'F3', user, DATE_FORMAT(date, '%Y-%m-%d') , activity
order by user , DATE_FORMAT(date, '%Y-%m-%d');

URL Rating
insert into features (featureId,id,user,dateActivity,pcAccessed,activity,featurePc,url)
select 'F4',id,user,date,pc,'http' as activity,substring(pc,4,4),url from http;

Set the Default Risk to Normal
update features set risk = 'Normal';

Cross Feature Extraction
insert into cfedata(user,dateActivity, device, activity, emailSize, url, risk)
select user, date, pc, activity, 0 as emalSize, '' as url, 'Normal' from logon where activity = 'Logon'
UNION
select user, date, pc, activity, 0 as emalSize, '' as url, 'Normal' from device where activity = 'Connect'
UNION
select user, date, pc, 'email' as activity, size as emalSize, '' as url, 'Normal' from email
UNION
select user, date, pc, 'http' as activity, 0 as emalSize, url, 'Normal' from http
UNION
select user, date, pc, 'file' as activity, 0 as emalSize, '' as url, 'Normal' from file;
*/
```

BuildThreat.sql

```
/* Replicate logon device email http threats
   This script sets up all the syntetic threat data
   new user CIA10...  are copied and amended from other users*/
/************** Threat 1 **********************/
delete from logonthreat; ALTER TABLE logonthreat AUTO_INCREMENT = 1;
delete from devicethreat; ALTER TABLE devicethreat AUTO_INCREMENT = 1;
delete from emailthreat; ALTER TABLE emailthreat AUTO_INCREMENT = 1;
delete from httpthreat; ALTER TABLE httpthreat AUTO_INCREMENT = 1;
delete from filethreat; ALTER TABLE filethreat AUTO_INCREMENT = 1;
delete from ldapthreat; ALTER TABLE ldapthreat AUTO_INCREMENT = 1;

insert into logonthreat ( id,date,user,pc,activity) select id,date,'CIA1001',pc,activity from logon where user =
'SBN0971' and activity = 'Logon';
insert into devicethreat ( id,date,user,pc,activity) select id,date,'CIA1001',pc,activity from device where user =
'SBN0971' and activity = 'Connect';
insert into emailthreat ( id,date,user,pc,eto,cc,bcc,efrom,size,attachments) select
id,date,'CIA1001',pc,eto,cc,bcc,efrom,size,attachments from email where user = 'SBN0971';
insert into httpthreat ( id,date,user,pc,url) select id,date,'CIA1001',pc,url from http where user = 'SBN0971';
insert into filethreat ( id,date,user,pc,filename) select id,date,'CIA1001',pc,filename from file where user =
'SBN0971';
insert into ldapthreat ( employee_name, user_id, email, role, business_unit, functional_unit, department, team,
supervisor, ldapYear, ldapMonth )
select 'Tom Threat', 'CIA1001', 'tomthreat@espionage.com', role, business_unit, functional_unit, department,
team, supervisor, ldapYear, ldapMonth
from ldap where user_id = 'SBN0971';

/* Inject the additional Syntetic Threats */
/* Additional late logon CIA1001 */
insert into logonthreat ( date,user,pc,activity) values ('2010-10-23 22:20:00','CIA1001','PC-0151','Logon');
```

```
/* 3 additional email of size */
insert into emailthreat ( date,user,pc,size,attachments) values ('2010-10-22 19:01:00','CIA1001','PC-0151',100000,2);
insert into emailthreat ( date,user,pc,size,attachments) values ('2010-10-22 19:05:00','CIA1001','PC-0151',200000,3);
insert into emailthreat ( date,user,pc,size,attachments) values ('2010-10-22 19:10:00','CIA1001','PC-0151',120000,0);
/* 1 http access*/
insert into httpthreat ( date,user,pc,url) values ('2010-10-22 19:20:00','CIA1001','PC-0151','http://espionage.com192290683.htm');

/********** Threat 2 **********/
insert into logonthreat ( id,date,user,pc,activity) select id,date,'CIA1002',pc,activity from logon where user = 'FEB0306' and activity = 'Logon';
insert into devicethreat ( id,date,user,pc,activity) select id,date,'CIA1002',pc,activity from device where user = 'FEB0306' and activity = 'Connect';
insert into emailthreat ( id,date,user,pc,eto,cc,bcc,efrom,size,attachments) select id,date,'CIA1002',pc,eto,cc,bcc,efrom,size,attachments from email where user = 'FEB0306';
insert into httpthreat ( id,date,user,pc,url) select id,date,'CIA1002',pc,url from http where user = 'FEB0306';
insert into filethreat ( id,date,user,pc,filename) select id,date,'CIA1002',pc,filename from file where user = 'FEB0306';
insert into ldapthreat ( employee_name, user_id, email, role, business_unit, functional_unit, department, team, supervisor, ldapYear, ldapMonth )
select 'Mary Threat', 'CIA1002', 'marythreat@espionage.com', role, business_unit, functional_unit, department, team, supervisor, ldapYear, ldapMonth
from ldap where user_id = 'FEB0306';

/* Insert of a large Email CIA1002*/
insert into emailthreat ( date,user,pc,size,attachments) values ('2010-10-21 20:01:00','CIA1002','PC-7757',1000000,1);

/* 15 connects Previous Day */
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:01:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:02:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:03:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:04:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:05:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:06:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:07:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:08:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:09:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:10:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:11:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:12:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:13:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:14:00','CIA1002','PC-7757','Connect');
insert into devicethreat ( date,user,pc,activity) values ('2010-10-20 19:15:00','CIA1002','PC-7757','Connect');


/* Merge the threat data into the general population */
insert into features (featureId,id,user,dateActivity,pcAccessed,activity,featurePc)
select 'F1',id,user,date,pc,activity,substring(pc,4,4) from logonthreat;

insert into features (featureId,user,dateActivity,pcAccessed,activity,featurePc,featureConnects)
select 'F2', user, DATE_FORMAT(date, '%Y-%m-%d') , pc, activity, substring(pc,4,4), count(pc) from devicethreat
group by 'F2', user, DATE_FORMAT(date, '%Y-%m-%d') , pc, activity, substring(pc,4,4)
order by user , DATE_FORMAT(date, '%Y-%m-%d'), pc;

insert into features (featureId,user,dateActivity,activity,featureEmails,featureEmailSize)
```

```
select 'F3', user, DATE_FORMAT(date, '%Y-%m-%d') , 'email' as activity, count(user), sum(size) as emailsize
from emailthreat
group by 'F3', user, DATE_FORMAT(date, '%Y-%m-%d') , activity
order by user , DATE_FORMAT(date, '%Y-%m-%d');


/* Cross Feature Extraction */
insert into cfedata(user,dateActivity, device, activity, emailSize, url, risk)
select user, date, pc, activity, 0 as emalSize, '' as url, 'Normal' from logonthreat
UNION
select user, date, pc, activity, 0 as emalSize, '' as url, 'Normal' from devicethreat
UNION
select user, date, pc, 'email' as activity, size as emalSize, '' as url, 'Normal' from emailthreat
UNION
select user, date, pc, 'http' as activity, 0 as emalSize, url, 'Normal' from httpthreat
UNION
select user, date, pc, 'file' as activity, 0 as emalSize, '' as url, 'Normal' from httpthreat;

/********** Threat 2 **********/

/* Set the Default Risk to Normal */
update cfedata set risk = 'Normal';
/* Set the Default Risk to Normal */
update features set risk = 'Normal';


/* These are once Offs post initial load
insert into user(user) values ('CIA1002');

update features set empNo = (select empno from user where user.user = features.user);
update cfedata set empNo = (select empno from user where user.user = cfedata.user);

delete from probability;
ALTER TABLE probability AUTO_INCREMENT = 1;
insert into probability(user) select user from user;

delete from probabilitydevice;
ALTER TABLE probabilitydevice AUTO_INCREMENT = 1;
insert into probabilitydevice(user,deviceAccessed) select distinct user,pcaccessed from features where featureId
= 'F2';

delete from probabilityemail;
ALTER TABLE probabilityemail AUTO_INCREMENT = 1;
insert into probabilityemail(user) select distinct user from features where featureId = 'F3';

update features set departmentId = (select departmentId from userdepartments where features.user =
userdepartments.user);

update features set featureActivity = (select activityId from activity where features.activity = activity.activity);
update cfedata set activityId = (select activityId from activity where activity.activity = cfedata.activity);

/* Set the Day of the Week for
update features set featureDayOfWeek = WEEKDAY(dateActivity) + 1;

update cfedata set urlRating = 10, risk = 'Threat' where activity = 'http' and
(url like '%wikileaks.org%') or (url like '%www.actualkeylogger.com%') or (url like
'%www.dailykeylogger.com%') or (url like '%espionage%')
*/
```

Buildfeatures.sql

```
/* Build up the user table and apply unique values to each employee */
/* Clean and set auto increame primary keys to zero */

delete from user;
ALTER TABLE user AUTO_INCREMENT = 1;
insert into user(user) (select distinct user from features);

update features set empNo = (select empno from user where user.user = features.user);
update cfedata set empNo = (select empno from user where user.user = cfedata.user);

delete from probability;
ALTER TABLE probability AUTO_INCREMENT = 1;
insert into probability(user) select user from user;

delete from probabilitydevice;
ALTER TABLE probabilitydevice AUTO_INCREMENT = 1;
insert into probabilitydevice(user,deviceAccessed) select distinct user,pcaccessed from features where featureId
= 'F2';

delete from probabilityemail;
ALTER TABLE probabilityemail AUTO_INCREMENT = 1;
insert into probabilityemail(user) select distinct user from features where featureId = 'F3';

/* Build up department records for each user */
update features set departmentId = (select departmentId from userdepartments where features.user =
userdepartments.user);

update features set featureActivity = (select activityId from activity where features.activity = activity.activity);
update cfedata set activityId = (select activityId from activity where activity.activity = cfedata.activity);

/* Set the Day of the Week for */
update features set featureDayOfWeek = WEEKDAY(dateActivity) + 1;
```

BuildFeatureTables.sql

```
/* Used as a lookup for the data's activities (Logon, HTTP, Device Connect .....) */
/* Run to load new actitivies  when required */

delete from activity;
ALTER TABLE activity AUTO_INCREMENT = 1;

insert into activity(activity)
select distinct(activity) from cfedata order by activity;

/* Build Stats on PC Access per Activity */

delete from departments;
ALTER TABLE departments AUTO_INCREMENT = 1;
insert into departments (department)
select distinct CONCAT(business_unit,functional_unit,department,team) from ldap;

delete from userdepartments;
ALTER TABLE userdepartments AUTO_INCREMENT = 1;
insert into userdepartments (user, department)
select distinct user_id, CONCAT(business_unit,functional_unit,department,team) from ldap;
insert into userdepartments (user, department)
select distinct user_id, CONCAT(business_unit,functional_unit,department,team) from ldapthreat;

/* Update user departments and then CertData for each user */
```

```
update userdepartments set departmentId = (select departmentId from departments where
departments.department = userdepartments.Department);
```

BuildSetThreatValues.sql

```
/* Update the specific label in features anf cfedata to 'Threat' */

update cfedata set urlRating = 10, risk = 'Threat' where activity = 'http' and
(url like '%wikileaks.org%') or (url like '%www.actualkeylogger.com%') or (url like
'%www.dailykeylogger.com%') or (url like '%espionage%');

update features set featureurlRating = 10, risk = 'Threat' where activity = 'http' and
(url like '%wikileaks.org%') or (url like '%www.actualkeylogger.com%') or (url like
'%www.dailykeylogger.com%') or (url like '%espionage%');

/************** Threat 1 / 2 **********************/
update features set risk = 'Threat' where featureurlrating > 0;
update featurescfe set risk = 'Threat' where featureurlrating > 0;
update featuresrecursive set risk = 'Threat' where featureurlrating > 0;

/************** Threat 3 **********************/
update featurescfe set risk = 'Threat' where user = 'RAR0725' and dateactivity = '2010-08-09 07:51:00' and
activity = 'Logon';
update featuresrecursive set risk = 'Threat' where user = 'RAR0725' and dateactivity = '2010-08-09 07:51:00'
and activity = 'Logon';

/************** Threat 4 **********************/
update cfedata set risk = 'Threat' where user = 'CIA1001' and activity = 'Logon' and dateactivity = '2010-10-23
22:20:00';
update features set risk = 'Threat' where user = 'CIA1001' and activity = 'Logon'  and dateActivity = '2010-10-23
22:20:00' ;
update featurescfe set risk = 'Threat' where user = 'CIA1001' and activity = 'Logon'  and dateActivity = '2010-
10-23 22:20:00';
update featuresrecursive set risk = 'Threat' where user = 'CIA1001' and activity = 'Logon'  and dateActivity =
'2010-10-23 22:20:00';


/************** Threat 4 **********************/
update cfedata set risk = 'Threat' where user = 'CIA1002' and activity = 'email' and dateactivity = '2010-10-21
20:01:00';
update features set risk = 'Threat' where user = 'CIA1002' and activity = 'email' and dateactivity = '2010-10-21
00:00:00';
update featuresrecursive set risk = 'Threat' where user = 'CIA1002' and activity = 'email' and dateactivity =
'2010-10-21 20:01:00';

/************** General Logon Threats **********************/
update features set risk = 'Threat' where activity = 'Logon' and time(dateactivity) > '20:00';
update featurescfe set risk = 'Threat' where activity = 'Logon' and time(dateactivity) > '20:00';
update featuresrecursive set risk = 'Threat' where activity = 'Logon' and time(dateactivity) > '20:00';

/* Set Risk levels to threat in featuresrecursive there url rating is 10 */
update featuresrecursive a, cfedata b set a.featureurlrating = 10, b.risk = 'Threat' where
a.user = b.user and a.dateactivity = b.dateactivity and b.urlrating = 10 and b.risk = 'Threat';
```

Appendix 4 Data Tables

logon (6×30)

key	id	date	user	pc	activity
1	{X1D9-S0ES98JV-5357PWMI}	2010-01-02 06:49:00	NGF0157	PC-6056	Logon
2	{G2B3-L6EJ61GT-2222RKSO}	2010-01-02 06:50:00	LRR0148	PC-4275	Logon
3	{U6Q3-U0WE70UA-3770UREL}	2010-01-02 06:53:04	LRR0148	PC-4124	Logon
4	{I0N5-R7NA26TG-6263KNGM}	2010-01-02 07:00:00	IRM0931	PC-7188	Logon
5	{D1S0-N6FH62BT-5398KANK}	2010-01-02 07:00:00	MOH0273	PC-6699	Logon
6	{S6P1-M4MK04BB-0722IITW}	2010-01-02 07:07:00	LAP0338	PC-5758	Logon
7	{M6O6-F9UU11TJ-2166YVFU}	2010-01-02 07:08:00	MHH0180	PC-9822	Logon

device (6×30)

key	id	date	user	pc	activity
1	{J1S3-L9UU75BQ-7790ATPL}	2010-01-02 07:21:06	MOH0273	PC-6699	Connect
2	{N7B5-Y7BB27SI-2946PUJK}	2010-01-02 07:37:41	MOH0273	PC-6699	Disconnect
3	{U1V9-Z7XT67KV-5649MYHI}	2010-01-02 07:59:11	HPH0075	PC-2417	Connect
4	{H0Z7-E6GB57XZ-1603MOXD}	2010-01-02 07:59:49	IIW0249	PC-0843	Connect
5	{L7P2-G4PX02RX-7999GYOY}	2010-01-02 08:04:26	IIW0249	PC-0843	Disconnect
6	{B4V1-F2XA86NJ-0683CLUB}	2010-01-02 08:17:35	HPH0075	PC-2417	Disconnect
7	{N7C0-Y0QQ16KO-4619HDNN}	2010-01-02 08:24:54	HSB0196	PC-8001	Connect
8	{J3A7-Z5JP56DF-7855PJKX}	2010-01-02 08:25:18	RRC0553	PC-6672	Connect
9	{H9X8-R0QB50BG-6009OAKA}	2010-01-02 08:25:19	MOH0273	PC-6699	Connect
10	{J9M2-G1YI63RE-9923EBLY}	2010-01-02 08:29:40	MOH0273	PC-6699	Disconnect
11	{F6Y5-L8KB48CH-8878LYRD}	2010-01-02 08:37:19	HSB0196	PC-8001	Disconnect

email (6×30)

id	date	user	pc	size	attachments
{R3I7-S4TX96FG-8219JWFF}	2010-01-02 07:11:45	LAP0338	PC-5758	25,830	0
{R0R9-E4GL59IK-2907OSWJ}	2010-01-02 07:12:16	MOH0273	PC-6699	29,942	0
{G2B2-A8XY58CP-2847ZJZL}	2010-01-02 07:13:00	LAP0338	PC-5758	28,780	0
{A3A9-F4TH89AA-8318GFGK}	2010-01-02 07:13:17	LAP0338	PC-5758	21,907	0
{E8B7-C8FZ88UF-2946RUQQ}	2010-01-02 07:13:28	MOH0273	PC-6699	17,319	0
{X8T7-A6BT54FP-7241DLBV}	2010-01-02 07:36:03	HVB0037	PC-7979	44,345	0
{H5J6-G2RS59KI-8386FFLL}	2010-01-02 07:52:20	NWK0215	PC-8370	35,328	0
{D9T8-M1HJ89XP-6364INQJ}	2010-01-02 07:54:12	LRR0148	PC-4275	25,255	1
{V3L7-L2RB92RV-9130MFPE}	2010-01-02 07:54:49	LRR0148	PC-4275	33,967	0
{D5K9-P0IJ71WK-6380CQSN}	2010-01-02 07:54:58	LRR0148	PC-4275	19,456	1

http (6×30)

key	id	date	user	pc	url
1	{V1Y4-S2IR20QU-6154HFXJ}	2010-01-02 06:55:16	LRR0148	PC-4275	http://msn.com/The_Human_Centipede_First_Sequence/katsuro/arjf309875127.htm
2	{Q5R1-T3EF87UE-2395RWZS}	2010-01-02 07:00:13	NGF0157	PC-6056	http://urbanspoon.com/Plunketts_Creek_Loyalsock_Creek/loyalsock/ivqrbtnzrferprivivatpbbxvatfnsrgltbfcryzh...
3	{X9O1-O0XW52VO-5806RPHG}	2010-01-02 07:03:46	NGF0157	PC-6056	http://aa.com/Rhodocene/rhodocenium/fhaavatqrfxgbcrkrphgvir1766627142.html
4	{GSS8-U5OG04TE-5299CCTU}	2010-01-02 07:05:26	IRM0931	PC-7188	http://groupon.com/Leonhard_Euler/leonhard/tneqravafrpgfsvfuvatobng292602446.php
5	{L0R4-A9DH29VP-4553AUWM}	2010-01-02 07:05:52	IRM0931	PC-7188	http://flickr.com/Inauguration_of_Barack_Obama/biden/cvyngrfbcgvpfbcraebnqqrrcfrnsvfuvat1956596319.jsp
6	{U7D0-K8FF04MI-4691ZHYP}	2010-01-02 07:05:55	IRM0931	PC-7188	http://skype.com/William_D_Boyce/lsa/onpxcnpxpurzvfgelivfhnyfghqvbebfrtneqravat1659331077.jsp
7	{D4T8-I2QG91QD-5892HWPX}	2010-01-02 07:06:28	LRR0148	PC-4275	http://wikipedia.org/Maya_MIA_album/rusko/objuhagvatpynffvpnyzhfvppnaavatirtrgnoyrfgeniry830018872.jsp
8	{A4Z0-C9LR36OE-7241XUSU}	2010-01-02 07:06:33	IRM0931	PC-7188	http://constantcontact.com/2008_ACC_Championship_Game/herzlich/onxrjnersvfuvatobngfrysurycserfujngre...
9	{K2P9-H6BL68NS-7303TLDJ}	2010-01-02 07:07:16	NGF0157	PC-6056	http://tribalfusion.com/Abbey_Theatre/classon/jbheevphyhzanghergenvyfivqrbtnzrf779057038.html
10	{N2Y8-O6BU87DG-3052HVGF}	2010-01-02 07:07:50	LRR0148	PC-4275	http://pcworld.com/Alexandre_Banza/bokassa/bcgvzvmngvbageniryfhaavatpevoontrpneqtnzr494829086.php
11	{B7T7-C5LJ57MV-5914HNQX}	2010-01-02 07:08:11	IRM0931	PC-7188	http://edmunds.com/RomanPersian_Wars/parthian/qngnonfrpbbxvat1195322042.jsp
12	{I7Q5-Y8LZ15QA-0537HNJP}	2010-01-02 07:08:31	NGF0157	PC-6056	http://gap.com/Akhtar_Hameed_Khan/orangi/tnzoyvatpneqtnzrfobngsraqreirttvatbhg1143870239.php
13	{G9R2-X4TJ00FW-3170ULVH}	2010-01-02 07:11:17	LRR0148	PC-4275	http://files.wordpress.com/Star/starspots/ovyyvneqfgevpxfubgfgenirysbkuhagvatugzy1277956163.aspx
14	{L8S5-A1PH31MO-9651EQOM}	2010-01-02 07:12:32	LRR0148	PC-4275	http://wikipedia.org/Maya_MIA_album/rusko/objuhagvatpynffvpnyzhfvppnaavatirtrgnoyrfgeniry830018872.jsp
15	{T214-M8BH40DX-5584IPWG}	2010-01-02 07:12:35	LAP0338	PC-5758	http://sfgate.com/Sweet_Track/westhav/guebhtuchgrkcrevzranfbcreobiviroffhesvat192290683.htm

file (6×30)

key	id	date	user	pc	filename
1	{L9G8-J9QE34VM-2834VDPB}	2010-01-02 07:23:14	MOH0273	PC-6699	EYPC9Y08.doc
2	{H0W6-L4FG38XG-9897XTEN}	2010-01-02 07:26:19	MOH0273	PC-6699	N3LTSU3O.pdf
3	{M3Z0-O2KK89OX-5716MBIM}	2010-01-02 08:12:03	HPH0075	PC-2417	D3D3WC9W.doc
4	{E1I4-S4QS61TG-3652YHKR}	2010-01-02 08:17:00	HPH0075	PC-2417	QCSW62YS.doc
5	{D4R7-E7JL45UX-0067XALT}	2010-01-02 08:24:57	HSB0196	PC-8001	AU75JV6U.jpg
6	{J6V8-N4VB17MR-1187ZIXI}	2010-01-02 08:26:49	RRC0553	PC-6672	8ICKVGMO.doc
7	{H0Z1-E6PI35PN-0771NJHQ}	2010-01-02 08:27:27	RRC0553	PC-6672	BAO9D5H2.doc
8	{B6V8-V7MT39OY-9781GQAI}	2010-01-02 08:27:37	RRC0553	PC-6672	1PO60RXU.doc
9	{L4I6-F0BS91WB-4633BBVN}	2010-01-02 08:28:08	MOH0273	PC-6699	JS09VZOJ.doc
10	{Y0B6-B2RL48RZ-9855JIMZ}	2010-01-02 08:28:17	RRC0553	PC-6672	MJVNF4DQ.doc
11	{L9N3-K0NQ00BL-3339RWUX}	2010-01-02 08:28:24	MOH0273	PC-6699	LX0I6B1U.pdf
12	{Q5E0-N4RY30SY-8029QMEW}	2010-01-02 08:29:10	RRC0553	PC-6672	WX1IWCEK.txt

ldap (12×30)

key	employee_name	user_id	email	role	business_unit	functional_unit	department	team	supervisor	ldapYear	ldapMonth
1	Calvin Edan Love	CEL0561	Calvin.Edan.Love@dtaa.com	ComputerProgrammer	1	2 - ResearchAndEngineering	2 - SoftwareManagement	3 - Software	Stephanie Briar Harrington	2,009	12
2	Christine Reagan Deleon	CRD0624	Christine.Reagan.Deleon@dtaa.com	Salesman	1	5 - SalesAndMarketing	2 - Sales	3 - RegionalSales	Winter Veda Burks	2,009	12
3	Jade Felicia Caldwell	JFC0557	Jade.Felicia.Caldwell@dtaa.com	SoftwareEngineer	1	2 - ResearchAndEngineering	2 - SoftwareManagement	3 - Software	Stephanie Briar Harrington	2,009	12
4	Aquila Stewart Dejesus	ASD0577	Aquila.Stewart.Dejesus@dtaa.com	ProductionLineWorker	1	3 - Manufacturing	3 - Assembly	3 - AssemblyDept	Whilemina Pandora England	2,009	12
5	Micah Abdul Rojas	MAR0955	Micah.Abdul.Rojas@dtaa.com	ProductionLineWorker	1	3 - Manufacturing	3 - Assembly	6 - AssemblyDept	Sandra Beverly Diaz	2,009	12
6	Gail Rhiannon Mcconnell	GRM0868	Gail.Rhiannon.Mcconnell@dtaa.com	Salesman	1	5 - SalesAndMarketing	2 - Sales	5 - RegionalSales	Hashim Damon Dudley	2,009	12
7	April Alika Levy	AAL0706	April.Alika.Levy@dtaa.com	ProductionLineWorker	1	3 - Manufacturing	3 - Assembly	4 - AssemblyDept	Alan Benjamin Holder	2,009	12
8	Rama Vielka Clayton	RVC0232	Rama.Vielka.Clayton@dtaa.com	Mathematician	1	2 - ResearchAndEngineering	1 - Research	1 - Lab	Jackson Linus Wilkerson	2,009	12
9	Tasha Casey Dalton	TCD0009	Tasha.Casey.Dalton@dtaa.com	AdministrativeAssistant	1	1 - Adminstration			Halla Cathleen Simmons	2,009	12
10	Aurora Sarah Manning	ASM0575	Aurora.Sarah.Manning@dtaa.com	ProductionLineWorker	1	3 - Manufacturing	3 - Assembly	3 - AssemblyDept	Whilemina Pandora England	2,009	12
11	Devin Abdul Rogers	DAR0885	Devin.Abdul.Rogers@dtaa.com	Technician	1	5 - SalesAndMarketing	3 - FieldService	5 - RegionalFieldService	Blythe Veda Cooke	2,009	12
12	Jacqueline Latifah Miles	JLM0364	Jacqueline.Latifah.Miles@dtaa.com	ITAdmin	1	1 - Adminstration	6 - Security	3 - ElectronicSecurity	Francis Brian Armstrong	2,009	12
13	Ferris Omar Barry	FOB0756	Ferris.Omar.Barry@dtaa.com	Technician	1	5 - SalesAndMarketing	3 - FieldService	4 - RegionalFieldService	Blaine Helen Cervantes	2,009	12

Appendix 5 ProbabilityBuilder

ProbabilitybuildMain.java

```
package com.cyber.probability;

/*  The function of this code is to update
    The probabilities of each feature
    Update the label Risk field for threats (Normal/Threats in the main aggregated data set for training and test.
    Then everything coming in at streaming is a Normal record

 */
import java.sql.*;
import java.time.LocalDateTime;

public class ProbabilitybuildMain {

    public static void main(String[] args) {

        /* F1 Feature 1 Logon Probability calculation
          For an individual logon, how many standard deviations from mean.
          To active a base value we calculate the time in a number of seconds example 01:01:01 = 3661
          For a user we set the mean and Standard deviation.
        */

       ProbabilityLogonCalc insertLogonProbability = new ProbabilityLogonCalc();
       ProbabilityDeviceCalc insertDeviceProbability = new ProbabilityDeviceCalc();
       ProbabilityEmailCalc insertEmailProbability = new ProbabilityEmailCalc();

       try {

          System.out.println("Load Starting " + LocalDateTime.now() + "\n");
```

```
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cert", "threatmon",
"threat1!");

        /* F1 Build Probability Mean / Standard Deviation and z scores in the features table */
        insertLogonProbability.updateLogon(con);

        /* F2 Build Device Connect Access Probability
          The Probability of each user using a Device is Probability (Device) = number of devices connected in
the period
          The total number of connect devices for that user for that connection type. P(D) = U/T
        */
        insertDeviceProbability.updateDevice(con);

        /* F3 */
        insertEmailProbability.updateEmail(con);

        con.close();

        System.out.println("Load Complete " + LocalDateTime.now() + "\n");

      } catch (Exception e) {
      }


   }

}
```

ProbabilitydeviceCalc.java

```
package com.cyber.probability;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class ProbabilityDeviceCalc {

   public static String updateDevice(Connection con ) {

      /*



       */
      ResultSet rs; PreparedStatement preparedStmt;
      String deviceProbability = "  select user, DATE_FORMAT(dateActivity, '%Y-%m-%d') as dateActivity,
pcaccessed, AVG(featureConnects) as Mu , STDDEV(featureConnects) as Sd " +
            "from features   where activity = 'Connect' and featureId = 'F2' group by user, pcaccessed";

      String updateDeviceProbability = "update probabilitydevice set deviceMu = ?, deviceSd = ? where user = ?
and deviceaccessed = ?";

      String updateCertdataDeviceProb = "update features, probabilitydevice set featureProbabilityConnect =
IFNULL((featureConnects  - deviceMu ) / deviceSd,0) " +
            "where features.user = probabilitydevice.user and pcAccessed = deviceAccessed and " +
            "activity = 'Connect' and featureId = 'F2'";

      try {
```

```
        Statement stmt = con.createStatement();

        System.out.println("F2. Updating Device Probability......");
        rs = stmt.executeQuery(deviceProbability);
        while (rs.next()) {

           preparedStmt = con.prepareStatement(updateDeviceProbability);
           preparedStmt.setDouble(1, rs.getDouble ("Mu"));
           preparedStmt.setDouble(2, rs.getDouble ("Sd"));
           preparedStmt.setString(3, rs.getString("user"));
           preparedStmt.setString(4, rs.getString("pcaccessed"));
           preparedStmt.execute();

           System.out.println(rs.getString("user") + "," + rs.getString("pcaccessed") + "," + rs.getDouble
("Mu")+ "," +   rs.getDouble ("Sd"));
        }

        preparedStmt = con.prepareStatement(updateCertdataDeviceProb);
        preparedStmt.executeUpdate();

        System.out.println("F2. Updating Device Probability Complete......");

     } catch (Exception e) {
        throw new RuntimeException(e);
     }

     return "";

  }
}
```

ProbabilityEmailCalc.java

```
package com.cyber.probability;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class ProbabilityEmailCalc {

   public static String updateEmail(Connection con ) {


     ResultSet rs; PreparedStatement preparedStmt;
     String emailProbability = "select user, AVG(featureEmails) as emailMu , STDDEV(featureEmails) as
emailSd , AVG(featureEmailSize) as emailSizeMu , STDDEV(featureEmailSize) as emailSizeSd " +
          "from features where activity = 'email' and featureId = 'F3' group by user";

     String updateEmailProbability = "update probabilityemail set emailMu = ?, emailSd = ?, emailSizeMu = ?,
emailSizeSd = ? where user = ?";

     String updateCertdataEmailProb = "update features, probabilityemail set featureProbabilityEmail =
IFNULL((featureEmails  - emailMu ) / emailSd,0) where features.user = probabilityemail.user and activity =
'email' and featureId = 'F3'";
```

```
        String updateCertdataEmailSizeProb = "update features, probabilityemail set featureProbabilityEmailSize
= IFNULL((featureEmailSize  - emailSizeMu ) / emailSizeSd,0) where features.user = probabilityemail.user and
activity = 'email' and featureId = 'F3'";


        try {

            Statement stmt = con.createStatement();

            System.out.println("F3. Updating Email Probability.....");
            rs = stmt.executeQuery(emailProbability);
            while (rs.next()) {

                preparedStmt = con.prepareStatement(updateEmailProbability);
                preparedStmt.setDouble(1, rs.getDouble ("emailMu"));
                preparedStmt.setDouble(2, rs.getDouble ("emailSd"));
                preparedStmt.setDouble(3, rs.getDouble("emailSizeMu"));
                preparedStmt.setDouble(4, rs.getDouble("emailSizeSd"));
                preparedStmt.setString(5, rs.getString("user"));

                preparedStmt.execute();

                System.out.println(rs.getString("user") + "," + rs.getDouble ("emailMu")+ "," +   rs.getDouble
("emailSd") + "," + rs.getDouble ("emailSizeMu")+ "," +   rs.getDouble ("emailSizeSd")     );
            }

            preparedStmt = con.prepareStatement(updateCertdataEmailProb);
            preparedStmt.executeUpdate();

            preparedStmt = con.prepareStatement(updateCertdataEmailSizeProb);
            preparedStmt.executeUpdate();

            System.out.println("F3. Updating Email Probability Complete.....");

        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        return "";

    }

}
```

ProbabilityLogonCalc.java

```
package com.cyber.probability;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class ProbabilityLogonCalc {

    public static String updateLogon(Connection con ) {

        /*
            Calculate the probability using z-score
```

```
        */
       ResultSet rs; PreparedStatement preparedStmt;

       String updateLogonProbability = "update probability set logonMu = ?, logonSd = ? where user = ?";
       String updateCertdataLogon = "update features, Probability set featureProbabilityLogon =
IFNULL((TIME_TO_SEC(`dateActivity`)  - LogonMu ) / LogonSd,0) where features.user = probability.user
and activity = 'Logon';";

       try {
          String standardDeviation = "select user, AVG(TIME_TO_SEC(`dateActivity`)) as Mu ,
STDDEV(TIME_TO_SEC(`dateActivity`)) as Sd " +
                "from features where activity = 'Logon' and featureId = 'F1' " +
                "group by user; ";
          Statement stmt = con.createStatement();

          System.out.println("F1.Updating Logon Probability Load.....");
          rs = stmt.executeQuery(standardDeviation);
          while (rs.next()) {

             preparedStmt = con.prepareStatement(updateLogonProbability);
             preparedStmt.setDouble(1, rs.getDouble ("Mu"));
             preparedStmt.setDouble(2, rs.getDouble ("Sd"));
             preparedStmt.setString(3, rs.getString("user"));
             preparedStmt.execute();

             System.out.println(rs.getString("user") + "," + rs.getDouble ("Mu")+ "," +   rs.getDouble ("Sd"));
          }
          preparedStmt = con.prepareStatement(updateCertdataLogon);
          preparedStmt.executeUpdate();

          System.out.println("F1.Updating Logon Probability Complete.....");

       } catch (Exception e) {
          throw new RuntimeException(e);
       }

       return "";

    }

}
```

Appendix 6 CertDataBuilder

BuildCFESlice.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class BuildCFESlice {

   public static String genSlice(Connection con ) {
```

```
ResultSet rs; ResultSet sd; PreparedStatement preparedStmt; String qSql = "";
Double logonMu =0.0; Double logonSd =0.0; Double deviceMu =0.0; Double deviceSd =0.0; Double deviceCnt =0.0; String query = "";
Double emailMu =0.0; Double emailSd =0.0; Double emailSizeMu =0.0; Double emailSizeSd =0.0; Double emailCnt =0.0; Double emailSize =0.0; Double deviceProd = 0.0;
int urlRating = 0; int interval = 0;

/* select all the activity records for a given time slice to build item features */
String cleanFeatures = "delete from featurescfe";
String selectSlice = "select sliceTime, user, empNo, dateActivity, substring(device,4,4) as device, activity, activityId, emailSize, url , TIME_TO_SEC(`dateActivity`) as TTS, " +
    "CASE WHEN WEEKDAY(dateActivity) + 1 > 5 THEN 1 ELSE 0 END as DOW, urlRating, risk from scheduler, cfedata where dateActivity\n" +
    "between  DATE_SUB(sliceTime, INTERVAL 1 HOUR) and sliceTime  and activity = 'Logon' order by slicetime";
String selectInterval = "select intervalDays from schedulerrun";
Integer counter = 0;

/* ** This function should always run after Probability Builder in order to capture mean and standard deviations of time
1. Calculate the probability of Time
2. Calculate the probability of activity in the last week being out of the normal
 */


try {

   Statement stmt = con.createStatement(); Statement sdmt = con.createStatement();
   preparedStmt = con.prepareStatement(cleanFeatures);
   preparedStmt.executeUpdate();

   BuildLogonSlice buildLogonSlice = new BuildLogonSlice();

   rs = stmt.executeQuery(selectInterval); while (rs.next()) { interval = rs.getInt("intervalDays"); }

   rs = stmt.executeQuery(selectSlice);
   while (rs.next()) {

      System.out.println(counter++ + ".CFE." + rs.getString("user") + " : " + rs.getString("device") + " " + rs.getString("activity"));

      /* Time If the activity item is a user Logon we build an initial feature */
      if (rs.getString("activity").equals("Logon")) {
         /* Return the mean and standard deviation of the access time for that client since inception */
         buildLogonSlice.genLogonFeatures (con, rs.getString("user"), rs.getInt("empNo"),
               rs.getString("dateActivity"), rs.getInt("device"), rs.getInt("activityId"), rs.getInt("DOW"),
               rs.getDouble("TTS") , rs.getString("activity"), rs.getString("risk"), interval);
      }



   }

} catch (Exception e) {
   throw new RuntimeException(e);
```

```
    }

    return "";

  }

  public static String genDay(Connection con ) {

    ResultSet rs; ResultSet sd; PreparedStatement preparedStmt;
    String cleanFeatrues = "update featurescfe set featureDayOfWeek = WEEKDAY(dateActivity) + 1;";

    try {

      Statement stmt = con.createStatement(); Statement sdmt = con.createStatement();
      preparedStmt = con.prepareStatement(cleanFeatrues);
      preparedStmt.executeUpdate();

    } catch (Exception e) {
      throw new RuntimeException(e);
    }

    return "";

}

}
```

BuildLogonSlice.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class BuildLogonSlice {


   public static String genLogonFeatures(Connection con, String user, int empNo, String dateActivity, int
device, int activityId, int DOW, double TTS, String activity, String risk, int interval ) {

      ResultSet rs; ResultSet sd; PreparedStatement preparedStmt; String qSql = "";
      Double logonMu =0.0; Double logonSd =0.0; Double deviceMu =0.0; Double deviceSd =0.0; Double
deviceCnt =0.0; String query = "";
      Double emailMu =0.0; Double emailSd =0.0; Double emailSizeMu =0.0; Double emailSizeSd =0.0;
Double emailCnt =0.0; Double emailSize =0.0;  Double deviceProd = 0.0;
      int urlRating = 0;

      try {

        Statement sdmt = con.createStatement();

        qSql = "select * from probability where user = '" + user + "'";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
          logonMu = sd.getDouble ("logonMu");
          logonSd = sd.getDouble ("logonSd");
```

```
        }
        sd.close();

        /* Device Build up the mean / standard deviation of a device for the prior 7 days.
          Device Find the Statistical probability of the device for the user and device in the prior 7 days against
the average away from the mean per week leading up
          to the event */

        qSql = "select user, device, AVG(deviceCnt) as deviceMu, STDDEV(deviceCnt) as deviceSd from
(select user, WEEK(dateActivity) as wk,  device, " +
               "count(activity) as deviceCnt from cfedata where user = '" + user + "' and substring(device,4,4) = '"
+  device +
               "' and activity = 'Connect' group by user, WEEK(dateActivity) , substring(device,4,4) " +
               "order by user , WEEK(dateActivity) ) sel group by user, device";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
           deviceMu = sd.getDouble ("deviceMu");
           deviceSd = sd.getDouble ("deviceSd");
        }
        sd.close();

        qSql = "select COUNT(ACTIVITY) as deviceCnt from cfedata where dateActivity between '" +
dateActivity + "' - INTERVAL "+interval+" DAY " +
               "and '" + dateActivity + "'" +
               " and user = '" + user + "' and activity = 'Connect' and substring(device,4,4) = '" + device + "'";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
           deviceCnt = sd.getDouble ("deviceCnt");
        }
        sd.close();

        if ( deviceSd == 0.0 ) {deviceProd =0.0;} else {deviceProd = (deviceCnt - deviceMu) / deviceSd;}

        /* Build up email Probability*/

        qSql = "select user, AVG(emailCnt) as emailMu, STDDEV(emailCnt) as emailSd, AVG(emailSize) as
emailSizeMu, STDDEV(emailSize) as emailSizeSd from (select user, WEEK(dateActivity) as wk, " +
               "count(activity) as emailCnt, sum(emailSize) as emailSize from cfedata where user = '" + user + "' "
+
               "and activity = 'email' group by user, WEEK(dateActivity) " +
               "order by user , WEEK(dateActivity) ) sel group by user";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
           emailMu = sd.getDouble ("emailMu");
           emailSd = sd.getDouble ("emailSd");
           emailSizeMu = sd.getDouble ("emailSizeMu");
           emailSizeSd = sd.getDouble ("emailSizeSd");
        }
        sd.close();
        qSql = "select COUNT(ACTIVITY) as emailCnt, sum(emailSize) as emailSize from cfedata where
dateActivity between '" + dateActivity + "' - INTERVAL "+interval+" DAY " +
               "and '" + dateActivity + "'" +
               " and user = '" + user + "' and activity = 'email' ";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
           emailCnt = sd.getDouble ("emailCnt");
           emailSize = sd.getDouble ("emailSize");
        }
        sd.close();
```

```
        /* Set url rating against web sites accessed */
        qSql = "select sum(urlRating) as urlRating from cfedata where dateActivity between '" + dateActivity +
"' - INTERVAL "+interval+" DAY " +
                "and '" + dateActivity + "'" +
                " and user = '" + user + "' and activity = 'http'";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
           urlRating = sd.getInt ("urlRating");
        }
        sd.close();

              /* Write the Features record
               1. Write the probability of Time
               2. Write the probability of unusual activity in the previous 7 days
               3. Write a metric for the day of Access (1-5) 0 and (6-7) 1 , indicating that a threat may be out side
normal hours */

        query = " insert into featurescfe (user, empNo, dateActivity, featureDevice, featureActivity,
featureDayofWeek, featureProbabilityLogon, activity, featureProbabilityDevice," +
              "featureProbabilityEmail, featureProbabilityEmailSize, featureUrlRating, risk)"
              + " values (?,?,?,?,?,?,?,?,?,?,?,?,?)";
        preparedStmt = con.prepareStatement(query);
        preparedStmt.setString(1, user);
        preparedStmt.setInt (2,   empNo);
        preparedStmt.setString (3,  dateActivity);
        preparedStmt.setInt (4,  device);
        preparedStmt.setLong (5,   activityId);
        preparedStmt.setLong (6,   DOW);
        preparedStmt.setDouble (7,   (TTS - logonMu) / logonSd);
        preparedStmt.setString(8, activity);
        preparedStmt.setDouble (9,   deviceSd);
        preparedStmt.setDouble (10,   (emailCnt - emailMu) / emailSd);
        preparedStmt.setDouble (11,   (emailSize - emailSizeMu) / emailSizeSd);
        preparedStmt.setInt (12,  urlRating);
        preparedStmt.setString(13, risk);
        preparedStmt.execute();

     } catch (Exception e) {
        throw new RuntimeException(e);
     }

     return "";

  }
}
```

Buildrecursize.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class BuildRecursive {

   public static String genRecursive(Connection con ) {

      /*  Build recursive class takes each activity (logon, email, device, file access    */
```

```
ResultSet rs; ResultSet sd; PreparedStatement preparedStmt; int interval = 0; Double logonZScore = 0.0;
int counter = 0;
String cleanFeatures = "delete from featuresrecursive";

String selectSlice = "select user, empNo, dateActivity, substring(device,4,4) as device, activity, " +
    "activityId, emailSize, url , TIME_TO_SEC(`dateActivity`) as TTS, " +
    "CASE WHEN WEEKDAY(dateActivity) + 1 > 5 THEN 1 ELSE 0 END as DOW, urlRating, " +
    "risk from cfedata order by dateActivity";

String selectInterval = "select intervalDays from schedulerrun";

RecursiveLogon recursiveLogon = new RecursiveLogon();
RecursiveHttp recursiveHttp = new RecursiveHttp();
RecursiveInsert recursiveInsert = new RecursiveInsert();
RecursiveEmail recursiveEmail = new RecursiveEmail();
RecursiveDevice recursiveDevice = new RecursiveDevice();
RecursiveFile recursiveFile = new RecursiveFile();

try {

  /*  This class reads every transaction from the database and creates a feature value
     for the given timeslice in the scheduler table
     for each activity type all other activities are accumlated in time slices and Z-Scores created
   */
  Statement stmt = con.createStatement();
  preparedStmt = con.prepareStatement(cleanFeatures);
  preparedStmt.executeUpdate();

  /* Get the interval number from the scheduler table */
  rs = stmt.executeQuery(selectInterval); while (rs.next()) { interval = rs.getInt("intervalDays"); }

  rs = stmt.executeQuery(selectSlice);
  while (rs.next()) {

    System.out.println(counter++ + " .Rec. " + rs.getString("user"));
    logonZScore = 0.0; int urlRating = 0; Double zScoredevice =0.0; Double zScorefile =0.0; Double
zScoreEmailSize =0.0; Double zScoreEmailCnt =0.0;

    /* Logon */
    if (rs.getString("activity").equals("Logon")) {
      logonZScore = recursiveLogon.genRecursive(con, rs.getString("user"), rs.getInt("TTS") ,
rs.getString("dateActivity"));
      urlRating = recursiveHttp.genRecursiveHttp(con, rs.getString("user"), rs.getString("dateActivity"),
interval);
      Double emailZscores[] = recursiveEmail.genRecursiveEmail(con, rs.getString("user"),
rs.getString("dateActivity"), interval);
      zScoredevice = recursiveDevice.genRecursiveDevice(con, rs.getString("user"),
rs.getString("dateActivity"), rs.getInt("device"), interval);
      zScorefile = recursiveFile.genRecursiveFile(con,  rs.getString("user"), rs.getString("dateActivity"),
rs.getInt("device"), interval);

      /* Insert Record */
      recursiveInsert.insertRecursive (con, rs.getString("user"), rs.getInt("empNo"),
          rs.getString("dateActivity"), rs.getInt("device"), rs.getInt("activityId"), rs.getInt("DOW"),
          logonZScore , rs.getString("activity"), rs.getString("risk"), interval, urlRating,
emailZscores[0], emailZscores[1], zScoredevice, zScorefile );
    }
    /* email */
    if (rs.getString("activity").equals("email")) {
      logonZScore = 0.0;
```

```
                urlRating = recursiveHttp.genRecursiveHttp(con, rs.getString("user"), rs.getString("dateActivity"),
interval);
                zScoreEmailSize = recursiveEmail.genRecursiveEmailSize(con, rs.getString("user"),
rs.getString("dateActivity"), rs.getDouble("emailSize"));
                Double emailZscores[] = recursiveEmail.genRecursiveEmail(con, rs.getString("user"),
rs.getString("dateActivity"), interval);
                zScoredevice = recursiveDevice.genRecursiveDevice(con, rs.getString("user"),
rs.getString("dateActivity"), rs.getInt("device"), interval);
                zScorefile = recursiveFile.genRecursiveFile(con,  rs.getString("user"), rs.getString("dateActivity"),
rs.getInt("device"), interval);

                /* Insert Record */
                recursiveInsert.insertRecursive (con, rs.getString("user"), rs.getInt("empNo"),
                        rs.getString("dateActivity"), rs.getInt("device"), rs.getInt("activityId"), rs.getInt("DOW"),
                        logonZScore , rs.getString("activity"), rs.getString("risk"), interval, urlRating,
emailZscores[0], zScoreEmailSize, zScoredevice, zScorefile );
            }

            if (rs.getString("activity").equals("Connect") || rs.getString("activity").equals("http") ||
rs.getString("activity").equals("file")) {
                logonZScore = 0.0;
                urlRating = recursiveHttp.genRecursiveHttp(con, rs.getString("user"), rs.getString("dateActivity"),
interval);
                Double emailZscores[] = recursiveEmail.genRecursiveEmail(con, rs.getString("user"),
rs.getString("dateActivity"), interval);
                zScoredevice = recursiveDevice.genRecursiveDevice(con, rs.getString("user"),
rs.getString("dateActivity"), rs.getInt("device"), interval);
                zScorefile = recursiveFile.genRecursiveFile(con,  rs.getString("user"), rs.getString("dateActivity"),
rs.getInt("device"), interval);

                /* Insert Record */
                recursiveInsert.insertRecursive (con, rs.getString("user"), rs.getInt("empNo"),
                        rs.getString("dateActivity"), rs.getInt("device"), rs.getInt("activityId"), rs.getInt("DOW"),
                        logonZScore , rs.getString("activity"), rs.getString("risk"), interval, urlRating,
emailZscores[0], emailZscores[1], zScoredevice, zScorefile );
            }

          }

      } catch (Exception e) {

      }

      return "";
   }
}
```

CertDataGen.java

```
package com.cyber.datagen;

import java.sql.*;

public class CertDataGen {

   public static void main(String[] args) {

      BuildRecursive  buildRecursive = new BuildRecursive();
      try {
```

```
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cert", " threatmon ",
"threat1!");

        System.out.println(".. Building Recursive Data Extract Start");

        buildRecursive.genRecursive(con);

        con.close();
        System.out.println(".. Recursive Data Extract Complete");

      } catch (Exception e) {
      }

   }
}
```

RecursiveDevice.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

   /* Device Build up the mean / standard deviation of a device for the prior 7 days.
    * Device Find the Statistical probability of the device for the user and device in the prior 7 days against the
average away from the mean per week leading up
    * to the event */


public class RecursiveDevice {

   public static Double genRecursiveDevice(Connection con, String user, String dateActivity, int device, int
interval ) {

      String qSql = ""; ResultSet sd;
      Double deviceMu =0.0; Double deviceSd =0.0; Double deviceCnt =0.0; Double zScoredevice =0.0;

      try {

         Statement sdmt = con.createStatement();
         qSql = "select user, device, AVG(deviceCnt) as deviceMu, STDDEV(deviceCnt) as deviceSd from
(select user, WEEK(dateActivity) as wk,  device, " +
               "count(activity) as deviceCnt from cfedata where user = '" + user + "' and substring(device,4,4) = '"
+  device +
               "' and activity = 'Connect' and dateActivity <= '" + dateActivity + "' group by user,
WEEK(dateActivity) , substring(device,4,4) " +
               "order by user , WEEK(dateActivity) ) sel group by user, device";
         sd = sdmt.executeQuery(qSql);
         while (sd.next()) {
            deviceMu = sd.getDouble ("deviceMu");
            deviceSd = sd.getDouble ("deviceSd");
         }
         sd.close();

         qSql = "select COUNT(ACTIVITY) as deviceCnt from cfedata where dateActivity between '" +
dateActivity + "' - INTERVAL "+interval+" DAY " +
               "and '" + dateActivity + "'" +
               " and user = '" + user + "' and activity = 'Connect' and substring(device,4,4) = '" + device + "'";
```

```
            sd = sdmt.executeQuery(qSql);
            while (sd.next()) {
               deviceCnt = sd.getDouble ("deviceCnt");
            }
            sd.close();

            if ( deviceSd == 0.0 ) {zScoredevice =0.0;} else {zScoredevice = (deviceCnt - deviceMu) / deviceSd;}

        } catch (Exception e) {

        }

        return zScoredevice;

    }


}
```

RecursiveEmail.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class RecursiveEmail {

    public static Double[] genRecursiveEmail(Connection con, String user, String dateActivity, int interval) {

        String qSql = ""; ResultSet sd;
        Double emailMu =0.0; Double emailSd =0.0; Double emailSizeMu =0.0; Double emailSizeSd =0.0;
Double emailCnt =0.0; Double emailSize =0.0;
        Double emailCntZScore = 0.0; Double emailSizeScore = 0.0;

        try {

            Statement sdmt = con.createStatement();
            qSql = "select user, AVG(emailCnt) as emailMu, STDDEV(emailCnt) as emailSd, AVG(emailSize) as
emailSizeMu, STDDEV(emailSize) " +
                    "as emailSizeSd from (select user, WEEK(dateActivity) as wk, " +
                    "count(activity) as emailCnt, sum(emailSize) as emailSize from cfedata where user = '" + user + "' "
+
                    "and activity = 'email' and dateActivity < '" + dateActivity + "' group by user, WEEK(dateActivity)
" +
                    "order by user , WEEK(dateActivity) ) sel group by user";
            sd = sdmt.executeQuery(qSql);
            while (sd.next()) {
               emailMu = sd.getDouble ("emailMu");
               emailSd = sd.getDouble ("emailSd");
               emailSizeMu = sd.getDouble ("emailSizeMu");
               emailSizeSd = sd.getDouble ("emailSizeSd");
            }

            sd.close();
            qSql = "select COUNT(ACTIVITY) as emailCnt, sum(emailSize) as emailSize from cfedata where
dateActivity between '" + dateActivity + "' - INTERVAL "+interval+" DAY " +
                    "and '" + dateActivity + "'" +
```

```
            " and user = '" + user + "' and activity = 'email' ";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
           emailCnt = sd.getDouble ("emailCnt");
           emailSize = sd.getDouble ("emailSize");
        }
        sd.close();

        if ( emailSd == 0.0 ) {emailCntZScore =0.0;} else {emailCntZScore = ((emailCnt - emailMu) /
emailSd);}
        if ( emailSizeSd == 0.0 ) {emailSizeScore =0.0;} else {emailSizeScore =  ((emailSize - emailSizeMu) /
emailSizeSd);}



    } catch (Exception e) {

    }

    return new Double[] {emailCntZScore,emailSizeScore};

  }


  public static Double genRecursiveEmailSize(Connection con, String user, String dateActivity, Double
emailSize) {

      String qSql = ""; ResultSet sd;
      Double emailMu =0.0; Double emailSd =0.0; Double emailCnt =0.0;
      Double emailSizeScore = 0.0;

      try {

        Statement sdmt = con.createStatement();
        qSql = "select user, AVG(emailSize) as emailMu, STDDEV(emailSize) as emailSd from cfedata where
user = '" + user + "' " +
              "and activity = 'email' and dateActivity < '" + dateActivity + "'";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
           emailMu = sd.getDouble ("emailMu");
           emailSd = sd.getDouble ("emailSd");
        }

        sd.close();

        if ( emailSd == 0.0 ) {emailSizeScore =0.0;} else {emailSizeScore = ((emailSize - emailMu) /
emailSd);}

      } catch (Exception e) {

      }

      return emailSizeScore;

  }


}
```

RecursiveFile.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class RecursiveFile {

   public static Double genRecursiveFile(Connection con, String user, String dateActivity, int device, int
interval ) {

      String qSql = ""; ResultSet sd;
      Double fileMu =0.0; Double fileSd =0.0; Double fileCnt =0.0; Double zScorefile =0.0;

      try {

         Statement sdmt = con.createStatement();
         qSql = "select user, device, AVG(fileCnt) as fileMu, STDDEV(fileCnt) as fileSd from (select user,
WEEK(dateActivity) as wk,  device, " +
               "count(activity) as fileCnt from cfedata where user = '" + user + "' and substring(device,4,4) = '" +
device +
               "' and activity = 'file' and dateActivity <= '" + dateActivity + "' group by user, WEEK(dateActivity)
, substring(device,4,4) " +
               "order by user , WEEK(dateActivity) ) sel group by user, device";
         sd = sdmt.executeQuery(qSql);
         while (sd.next()) {
            fileMu = sd.getDouble ("fileMu");
            fileSd = sd.getDouble ("fileSd");
         }
         sd.close();

         qSql = "select COUNT(ACTIVITY) as fileCnt from cfedata where dateActivity between '" +
dateActivity + "' - INTERVAL "+interval+" DAY " +
               "and '" + dateActivity + "'" +
               " and user = '" + user + "' and activity = 'file' and substring(device,4,4) = '" + device + "'";
         sd = sdmt.executeQuery(qSql);
         while (sd.next()) {
            fileCnt = sd.getDouble ("fileCnt");
         }
         sd.close();

         if ( fileSd == 0.0 ) {zScorefile =0.0;} else {zScorefile = (fileCnt - fileMu) / fileSd;}


      } catch (Exception e) {

      }

      return zScorefile;
   }
}
```

RecursiveHttp.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.Statement;

public class RecursiveHttp {

   public static Integer genRecursiveHttp(Connection con, String user, String dateActivity, int interval) {

      String qSql = ""; ResultSet sd; int urlWeight = 0;

      try {

         Statement sdmt = con.createStatement();
         qSql = "select sum(urlRating) as urlRating from cfedata where dateActivity between '" + dateActivity +
"' - INTERVAL "+interval+" DAY " +
               "and '" + dateActivity + "'" +
               " and user = '" + user + "' and activity = 'http'";
         sd = sdmt.executeQuery(qSql);
         while (sd.next()) {
            urlWeight = sd.getInt ("urlRating");
         }
         sd.close();


      } catch (Exception e) {

      }

      return urlWeight;

   }

}
```

RecursiveLogon.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class RecursiveLogon {

   public static Double genRecursive(Connection con, String user, double TTS, String dateActivity) {

      String qSql = ""; ResultSet sd; Double logonMu =0.0; Double logonSd =0.0; Double logonZScore = 0.0;

      try {

      Statement sdmt = con.createStatement();
      qSql = "select user, AVG(TIME_TO_SEC(`dateActivity`)) as logonMu ,
STDDEV(TIME_TO_SEC(`dateActivity`)) as logonSd from cfedata where user = '"
            + user + "'  and activity = 'Logon' and  dateActivity < '" + dateActivity + "'";
      sd = sdmt.executeQuery(qSql);
      while (sd.next()) {
         logonMu = sd.getDouble ("logonMu");
         logonSd = sd.getDouble ("logonSd");
      }
      sd.close();
```

```
    if ( logonSd == 0.0 ) {logonZScore =0.0;} else {logonZScore = ((TTS - logonMu) / logonSd);}


  } catch (Exception e) {

  }

    return logonZScore;

  }

}
```

RecursiveInsert.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class RecursiveInsert {

  public static String insertRecursive (Connection con, String user, int empNo, String dateActivity, int device,
int activityId, int DOW,
                                double logonZScore, String activity, String risk, int interval, int urlRating, Double
emailCntZScore, Double emailSizeScore,
                                Double zScoredevice, Double zScorefile) {

    String query = ""; PreparedStatement preparedStmt;

    try {


        query = " insert into featuresrecursive (user, empNo, dateActivity, featureDevice, featureActivity,
featureDayofWeek, featureProbabilityLogon, activity, featureProbabilityDevice," +
              "featureProbabilityEmail, featureProbabilityEmailSize, featureProbabilityFile, featureUrlRating,
risk)"
              + " values (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";

        preparedStmt = con.prepareStatement(query);
        preparedStmt.setString(1, user);
        preparedStmt.setInt (2,   empNo);
        preparedStmt.setString (3,  dateActivity);
        preparedStmt.setInt (4,  device);
        preparedStmt.setLong (5,   activityId);
        preparedStmt.setLong (6,   DOW);
        preparedStmt.setDouble (7,  logonZScore);
        preparedStmt.setString(8, activity);
        preparedStmt.setDouble (9,   zScoredevice);
        preparedStmt.setDouble (10,   emailCntZScore);
        preparedStmt.setDouble (11,   emailSizeScore);
        preparedStmt.setDouble (12,   zScorefile);
        preparedStmt.setInt (13,  urlRating);
        preparedStmt.setString(14, risk);
        preparedStmt.execute();
```

```
        } catch (Exception e) {

        }

        return "";



    }
}
```

ScheduleBuilder.java

```
package com.cyber.datagen;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class ScheduleBuilder {

    public static String genScheduler(Connection con ) {

        ResultSet rs; PreparedStatement preparedStmt;

        String buildSchedule = "insert into scheduler select DATE_ADD(max(sliceTime), INTERVAL 1 HOUR)
from scheduler";

        try {

            preparedStmt = con.prepareStatement(buildSchedule);
            for (int i = 0; i < 5976; i++) {
                preparedStmt.executeUpdate();
            }


        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        return "";

    }

}
```

Appendix 7 ThreatMonitor

BuildModel.java

```
import org.apache.commons.io.FileUtils;
import org.apache.spark.ml.classification.RandomForestClassificationModel;
import org.apache.spark.ml.classification.RandomForestClassifier;
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator;
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator;
import org.apache.spark.ml.feature.VectorAssembler;
import org.apache.spark.sql.Dataset;
```

```
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.ml.feature.*;
import org.apache.spark.mllib.linalg.Matrix;
import org.apache.spark.mllib.evaluation.MulticlassMetrics;
import org.apache.spark.ml.classification.NaiveBayes;
import org.apache.spark.ml.classification.NaiveBayesModel;
import static org.apache.spark.sql.functions.col;
import static org.apache.spark.sql.functions.when;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

import java.io.File;
import java.util.List;
import java.util.Random;


public class BuildModel {

   public static String executModel(String feature, String FeatureDesc, String feat, String[] inputColumns, String
sliceTime ) {

      String OutputModel="C:\\Dataset\\spark.model";
      String startRunTime; String finishRunTime; String timeLine = "";
      timeLine = "Model Started " + LocalDateTime.now(); LocalDateTime stime = LocalDateTime.now();
      Boolean noConsole = false; Boolean noFileLog= true; double accuracy =0;

      // initialise Spark session
      SparkSession sparkSession = SparkSession.builder().appName("SparkThreat").config("spark.master",
"local").getOrCreate();
      sparkSession.sparkContext().setLogLevel("OFF");

      try {

         /* Initialise all the logging classes */
         WriteDBLog writeLog = new WriteDBLog();
         WriteDBLogResults writeDBLogResults = new WriteDBLogResults();
         WritePredictionResults writePredictionResults = new WritePredictionResults();
         ConfusionMatrix confusionMatrix = new ConfusionMatrix();

         /* Record Algorithm Start Time */
         DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm");
         startRunTime = stime.format(format);

         /* Load data from the MySQL database into the rawdata dataset */
         /* use the variable feature to deine the interval date and the test type, simple or recursive */
         Dataset<Row> rawData = sparkSession.read()
               .format("jdbc")
               .option("url", "jdbc:mysql://localhost:3306/cert")
               .option("dbtable", feature)
               .option("user", "threatmon")
               .option("password", "threat1!")
               .load();

         Dataset<Row> transformedDataSet = null;
```

```
        /* build the transformation dataset with column headers depending on the algorithm  type  */
        switch(feat) {
           case "F1":
              transformedDataSet = rawData
                    .withColumn("empno", rawData.col("empno").cast("integer"))
                    .withColumn("featureActivity", rawData.col("featureActivity").cast("integer"))
                    .withColumn("featureDayOfWeek", rawData.col("featureDayOfWeek").cast("integer"))
                    .withColumn("featureDevice", rawData.col("featuredevice").cast("integer"))
                    .withColumn("featureProbabilityLogon",
rawData.col("featureProbabilityLogon").cast("double"))
                    .withColumn("label", when(col("Risk").equalTo("Normal"), 0)
                         .when(col("Risk").equalTo("Threat"), 1).otherwise(3));

              break;
           case "F2":
              transformedDataSet = rawData
                    .withColumn("empno", rawData.col("empno").cast("integer"))
                    .withColumn("featureActivity", rawData.col("featureActivity").cast("integer"))
                    .withColumn("featurePc", rawData.col("featurePc").cast("integer"))
                    .withColumn("featureProbabilityConnect",
rawData.col("featureProbabilityConnect").cast("double"))
                    .withColumn("label", when(col("Risk").equalTo("Normal"), 0)
                         .when(col("Risk").equalTo("Threat"), 1).otherwise(3));
              break;
           case "F4":
              transformedDataSet = rawData
                    .withColumn("empno", rawData.col("empno").cast("integer"))
                    .withColumn("featureActivity", rawData.col("featureActivity").cast("integer"))
                    .withColumn("featurePc", rawData.col("featurePc").cast("integer"))
                    .withColumn("featureurlrating", rawData.col("featureurlrating").cast("integer"))
                    .withColumn("label", when(col("Risk").equalTo("Normal"), 0)
                         .when(col("Risk").equalTo("Threat"), 1).otherwise(3));

              break;
           case "F3":
              transformedDataSet = rawData
                    .withColumn("empno", rawData.col("empno").cast("integer"))
                    .withColumn("featureActivity", rawData.col("featureActivity").cast("integer"))
                    .withColumn("featureProbabilityEmail", rawData.col("featureProbabilityEmail").cast("double"))
                    .withColumn("featureProbabilityEmailSize",
rawData.col("featureProbabilityEmailSize").cast("double"))
                    .withColumn("label", when(col("Risk").equalTo("Normal"), 0)
                         .when(col("Risk").equalTo("Threat"), 1).otherwise(3));
              break;
           case "CFE":
              transformedDataSet = rawData
                    .withColumn("empno", rawData.col("empno").cast("integer"))
                    .withColumn("featureDevice", rawData.col("featureDevice").cast("integer"))
                    .withColumn("featureActivity", rawData.col("featureActivity").cast("integer"))
                    .withColumn("featureDayOfWeek", rawData.col("featureDayOfWeek").cast("integer"))
                    .withColumn("featureProbabilityLogon",
rawData.col("featureProbabilityLogon").cast("double"))
                    .withColumn("featureProbabilityDevice",
rawData.col("featureProbabilityDevice").cast("double"))
                    .withColumn("featureProbabilityEmail", rawData.col("featureProbabilityEmail").cast("double"))
                    .withColumn("featureProbabilityEmailSize",
rawData.col("featureProbabilityEmailSize").cast("double"))
```

```
                .withColumn("featureUrlRating", rawData.col("featureUrlRating").cast("integer"))
                .withColumn("label", when(col("Risk").equalTo("Normal"), 0)
                    .when(col("Risk").equalTo("Threat"), 1).otherwise(3));
            break;
          case "REC":
            transformedDataSet = rawData
                .withColumn("empno", rawData.col("empno").cast("integer"))
                .withColumn("featureDevice", rawData.col("featureDevice").cast("integer"))
                .withColumn("featureActivity", rawData.col("featureActivity").cast("integer"))
                .withColumn("featureDayOfWeek", rawData.col("featureDayOfWeek").cast("integer"))
                .withColumn("featureProbabilityLogon",
rawData.col("featureProbabilityLogon").cast("double"))
                .withColumn("featureProbabilityDevice",
rawData.col("featureProbabilityDevice").cast("double"))
                .withColumn("featureProbabilityEmail", rawData.col("featureProbabilityEmail").cast("double"))
                .withColumn("featureProbabilityEmailSize",
rawData.col("featureProbabilityEmailSize").cast("double"))
                .withColumn("featureProbabilityFile", rawData.col("featureProbabilityFile").cast("double"))
                .withColumn("featureUrlRating", rawData.col("featureUrlRating").cast("integer"))
                .withColumn("label", when(col("Risk").equalTo("Normal"), 0)
                    .when(col("Risk").equalTo("Threat"), 1).otherwise(3));
            break;
        }

        System.out.println("     " + FeatureDesc + " Sample Size: " + transformedDataSet.count() + " Items and "
+ transformedDataSet.columns().length + " Features.");

        if (noConsole) {
          transformedDataSet.show(1);
        }

        VectorAssembler assembler = new
VectorAssembler().setInputCols(inputColumns).setOutputCol("features");
        Dataset<Row> featureSet = assembler.transform(transformedDataSet);

        /* Random Forest Parameters */
        String impurity = "gini";         /* information gain calculation */
        Integer numTrees = 100;
        Integer maxDepth = 20;            /* Maximun Depth of a Tree */
        Integer maxBins = 100;            /* Bins for splitting features */
        long seed = 1234;                 /* Seed set to maintain consistancy in data */
        String subsetStrategy = "sqrt";

        /* Split up the dataset into training and testing */
        Dataset<Row>[] threatTrainTestSet = featureSet.randomSplit(new double[]{0.7, 0.3});
        Dataset<Row> trainingDataSet = threatTrainTestSet[0];
        Dataset<Row> testDataSet = threatTrainTestSet[1];

        System.out.println("     " + FeatureDesc + " Training Set: " + trainingDataSet.count() + " Items and " +
trainingDataSet.columns().length + " Features.");
        System.out.println("     " + FeatureDesc + " Test Set: " + testDataSet.count() + " Items and " +
testDataSet.columns().length + " Features.");

        if (noConsole) {
           System.out.println(FeatureDesc + " Training Set: " + trainingDataSet.count() + " Items and " +
trainingDataSet.columns().length + " Features.");
           trainingDataSet.show(1);
```

```
            System.out.println(FeatureDesc + " Test Set: " + testDataSet.count() + " Items and " +
testDataSet.columns().length + " Features.");
            testDataSet.show(1);
        }

        /* Run the random forest algorthm */
        RandomForestClassifier randomForestClassifier = new RandomForestClassifier()
              .setSeed(seed)
              .setImpurity(impurity)
              .setMaxDepth(maxDepth)
              .setFeatureSubsetStrategy(subsetStrategy)
              .setNumTrees(numTrees)
              .setMaxBins(maxBins);
        RandomForestClassificationModel model = randomForestClassifier.fit(trainingDataSet);
        Dataset<Row> predictions = model.transform(testDataSet);

        /* Save the model for further use */
        FileUtils.deleteDirectory(new File(OutputModel));
        model.save(OutputModel);

        /* All Metrics Accuracy |  Confusion Matrix*/

        MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator()
              .setLabelCol("label")
              .setPredictionCol("prediction")
              .setMetricName("accuracy");
        accuracy = evaluator.evaluate(predictions);
        List<Row> predictionAsRows =
              predictions.select("id", "label", "features", "rawPrediction", "probability",
"prediction").collectAsList();

        System.out.println("Random Forest    Accuracy = " + accuracy + " Test Error " + (1.0 - accuracy) + "
Predictions Count: " + predictions.count() + "\n");

        /*confusionMatrix.buildConfusionMatrix(predictionAsRows);*/


          if (noConsole) {
            predictions.show(10);

System.out.println("_____
___");
        }

        /* Write the Timeline of the Model runtime to the Prediction output file. */
        /* Build up Model Logging Details */

        timeLine = timeLine + " " + "" + "Completed " + LocalDateTime.now() + "\n"; LocalDateTime fTime =
LocalDateTime.now(); finishRunTime = fTime.format(format);
        Random r = new Random();  int low = 1; int high = 10000000; int result = r.nextInt(high-low) + low;

        writeLog.writeDBLog(result, startRunTime, finishRunTime, transformedDataSet.count(),
trainingDataSet.count(), predictions.count(), (1.0 - accuracy), accuracy, feat, sliceTime, 0);
        writeDBLogResults.writeDBLogResults(result, predictionAsRows);

        if (noFileLog) {
            writePredictionResults.writePredictionResults(result, timeLine, accuracy, predictionAsRows);
```

```
        }

        sparkSession.stop();

      } catch (Exception e) {
        sparkSession.stop();
        throw new RuntimeException(e);

      }

      return "";
    }


}
```

ConfusionMatrix.java

```
import org.apache.spark.sql.Row;

import java.io.File;
import java.io.PrintWriter;
import java.util.List;

public class ConfusionMatrix {

   public static String buildConfusionMatrix( List<Row> predictions ) {

      final int [] cm ={0};
      try {

         predictions.forEach(row -> {

            Double label =  Double.parseDouble(row.get(1).toString());
            Double prediction =  Double.parseDouble(row.get(5).toString());

            if(label == 1 && prediction == 1) { cm[0]++; } //TP
            if(label == 0 && prediction == 0) { cm[1]++; } //TN
            if(label == 1 && prediction == 0) { cm[2]++; } //FP
            if(label == 0 && prediction == 1) { cm[3]++; } //FN

         });

         String outputLinex = "TP: " + cm[0] + " , " + "TN: " + cm[1]  + " , " + "FP: " + cm[2]  + " , " + "FN: " + cm[3] ;
         System.out.println(":" + outputLinex);*/

      } catch (Exception e) {
         throw new RuntimeException(e);
      }

      return  "";

   }


}
```

FeatureModel.java

```
public class FeatureModel {

   public static String featureF1Test( ) {

      String feature = "(select featureKey as id,empno,featureActivity, featureDayOfWeek, featureDevice,
featureProbabilityLogon, risk from featuresrecursive where  activity = 'Logon' ) features";
      return feature;

   }

   public static String featureF2Test( ) {

      String feature = "(select featureKey as id,empno,featureActivity, featurePc, featureProbabilityConnect, risk
from features where featureId = 'F2' ) features";
      return feature;

   }

   public static String featureF3Test( ) {

      String feature = "(select featureKey as id,empno,featureActivity, featureProbabilityEmail,
featureProbabilityEmailSize, risk from features where featureId = 'F3'  ) features";
      return feature;

   }

   public static String featureF4Test( ) {

      String feature = "(select featureKey as id, empno, featureactivity, featurepc, featureurlrating, risk from
features where featureid = 'F4'  ) features";
      return feature;

   }

   public static String featureCFETest( String sliceTime) {

      String feature = "( select featureKey as id, empNO, featureDevice, featureActivity, featureDayOfWeek,
featureProbabilityLogon, featureProbabilityDevice, " +
            "featureProbabilityEmail, featureProbabilityEmailSize, featureUrlRating, risk from featurescfe where
dateActivity <= '" + sliceTime + "') features";
      return feature;

   }

   public static String featureRECTest( String sliceTime) {

      String feature = "( select featureKey as id, empNO, featureDevice, featureActivity, featureDayOfWeek,
featureProbabilityLogon, featureProbabilityDevice, " +
            "featureProbabilityEmail, featureProbabilityEmailSize, featureProbabilityFile, featureUrlRating, risk
from featuresrecursive where dateActivity <= '" + sliceTime + "') features";
      return feature;

   }
}
```

SparkThreat.java

```
public class SparkThreat {

   public static void main(String[] args) {

      try {

         /* The SparkThreat Class build the lab to run the test methods*/

         System.out.println("Model Threat Prediction algorithm Starting....." + "\n");

         ThreatSingleFeature threatSingleFeature = new ThreatSingleFeature();
         threatSingleFeature.singleFeature();

         ThreatRecursive threatRecursive = new ThreatRecursive();
         threatRecursive.recursiveExtraction();

         System.out.println("algorithm Complete.....");

      } catch (Exception e) {

      }

   }


}
```

ThreatCrossFeature.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ThreatCrossFeature {

   public static String crossFeatureExtraction( ) {

      try {

         Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cert", " threatmon ",
"threat1!");
         Statement stmt = con.createStatement(); ResultSet rs; String sliceTime =""; long counter=0;

         BuildModel buildModel = new BuildModel();
         FeatureModel featureModel = new FeatureModel();

         /* Schedule a run for each hour from the first date to the last date from inception */
         /* Example run from 2011-05-07 00:00:00 to 2011-05-07 23:00:00  23 hours */

         System.out.println("Cross Feature Extraction Starting ...");
```

```
        String selectSlice = "select * from scheduler where sliceTime between (select startSlice from
schedulerrun) and ( select endSlice from schedulerrun) ";
        rs = stmt.executeQuery(selectSlice);

        while (rs.next()) {

           counter ++; System.out.println(counter + ". " + rs.getString("sliceTime") + " Build Cross Features ");
           String[] inputColumnsCFE = {"empno", "featureDevice", "featureActivity", "featureDayOfWeek",
"featureProbabilityLogon", "featureProbabilityDevice", "featureProbabilityEmail",
"featureProbabilityEmailSize", "featureUrlRating"};
           buildModel.executModel(featureModel.featureCFETest(rs.getString("sliceTime")), "Cross Feature
Extraction", "CFE", inputColumnsCFE, rs.getString("sliceTime"));

        }

     } catch (Exception e) {

     }

     System.out.println("Cross Feature Extraction Complete ...");

     return "";

     }
}
```

ThreatRecursive.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ThreatRecursive {

   public static String recursiveExtraction( ) {

     try {


        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cert", " threatmon ",
"threat1!");
        Statement stmt = con.createStatement(); ResultSet rs; String sliceTime =""; long counter=0;

        BuildModel buildModel = new BuildModel();
        FeatureModel featureModel = new FeatureModel();

        System.out.println("Cross Feature Extraction Starting ...");

        String selectSlice = "select * from scheduler where sliceTime between (select startSlice from
schedulerrun) and ( select endSlice from schedulerrun) ";
        rs = stmt.executeQuery(selectSlice);

        while (rs.next()) {

           counter ++; System.out.println(counter + ". " + rs.getString("sliceTime") + " Build Recursive Features
");
```

```
            String[] inputColumnsCFE = {"empno", "featureDevice", "featureActivity", "featureDayOfWeek",
"featureProbabilityLogon", "featureProbabilityDevice", "featureProbabilityEmail",
"featureProbabilityEmailSize", "featureProbabilityFile", "featureUrlRating"};
            buildModel.executModel(featureModel.featureRECTest(rs.getString("sliceTime")), "Recursive
Feedback Extraction", "REC", inputColumnsCFE, rs.getString("sliceTime"));

          }


      } catch (Exception e) {

      }

      return "";


      }
}
```

ThreatSingleFeature.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ThreatSingleFeature {


   public static String singleFeature( ) {

      try {

         Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/cert", " threatmon ",
"threat1!");
         Statement stmt = con.createStatement(); ResultSet rs;  String sliceTime ="";

         BuildModel buildModel = new BuildModel();
         FeatureModel featureModel = new FeatureModel();

         System.out.println("Single Time Feature Starting ...");

         String[] inputColumnsF1 = {"empno", "featureActivity", "featureDayOfWeek", "featureDevice",
"featureProbabilityLogon"};
         buildModel.executModel(featureModel.featureF1Test(), "Logon Time Feature", "F1", inputColumnsF1,
sliceTime);

         System.out.println("Single Time Feature Complete....." + "\n");



      } catch (Exception e) {

      }

      return "";
```

```
  }


}
```

WriteDBLog.java

```
import java.util.Random;
import java.sql.*;

public class WriteDBLog {

      public static String writeDBLog(int rValue, String startRunTime, String finishRunTime, long
transformedDataSet, long trainingSet,  long predictions, double testError, double accuracy, String feat, String
sliceTime, double ROC) {

         String query = "";
         PreparedStatement preparedStmt;
         String OutputDir="C:\\Dataset\\Spark_Predictions_" + rValue + ".csv";

         try {

            Connection con = DriverManager.getConnection(
                  "jdbc:mysql://localhost:3306/cert", " threatmon ", "threat1!");
            Statement stmt = con.createStatement();
            query = " insert into log (logId, startTime, finishTime, sampleSize, trained, predictionSize, testError,
accuracy, feature, sliceTime, logfile, ROC)"
                  + " values (?,?,?,?,?,?,?,?,?,?,?,?)";

            preparedStmt = con.prepareStatement(query);
            preparedStmt.setInt(1, rValue);
            preparedStmt.setString (2,   startRunTime.toString());
            preparedStmt.setString (3,   finishRunTime.toString());
            preparedStmt.setLong (4,   transformedDataSet);
            preparedStmt.setLong (5,   trainingSet);
            preparedStmt.setLong (6,   predictions);
            preparedStmt.setDouble (7, testError);
            preparedStmt.setDouble(8, accuracy);
            preparedStmt.setString(9, feat);
            preparedStmt.setString(10, sliceTime);
            preparedStmt.setString(11, OutputDir);
            preparedStmt.setDouble(12, ROC);
            preparedStmt.execute();

            con.close();

         } catch (Exception e) {
            throw new RuntimeException(e);
         }

         return "";
      }


}
```

WriteDBLogResults.java

```
import org.apache.spark.sql.Row;

import java.util.List;
import java.util.Random;
import java.sql.*;

public class WriteDBLogResults {

   public static String writeDBLogResults(int rValue, List<Row> predictions) {

      String query = "";
      PreparedStatement preparedStmt;

      try {

         Connection con = DriverManager.getConnection(
               "jdbc:mysql://localhost:3306/cert", " threatmon ", "threat1!");
         Statement stmt = con.createStatement();

         query = " insert into logresults (logId, id, label, prediction, features, predictions, probabilities )"
               + " values (?,?,?,?,?,?,?)";

         for (int i = 0; i < predictions.size(); i++) {
            if( Double.parseDouble(predictions.get(i).get(1).toString()) !=
Double.parseDouble(predictions.get(i).get(5).toString()) ) {
               preparedStmt = con.prepareStatement(query);
               preparedStmt.setInt(1, rValue);
               preparedStmt.setInt(2, Integer.parseInt(predictions.get(i).get(0).toString()));
               preparedStmt.setDouble(3, Double.parseDouble(predictions.get(i).get(1).toString()));
               preparedStmt.setDouble(4, Double.parseDouble(predictions.get(i).get(5).toString()));
               preparedStmt.setString(5, predictions.get(i).get(2).toString() );
               preparedStmt.setString(6, predictions.get(i).get(3).toString() );
               preparedStmt.setString(7, predictions.get(i).get(4).toString() );
               preparedStmt.execute();
            }
         }


      } catch (Exception e) {
         throw new RuntimeException(e);
      }

      return "";
   }
}
```

WritePredictionResults.java

```
import org.apache.spark.sql.Row;

import java.io.File;
import java.io.PrintWriter;
import java.util.List;
```

```
public class WritePredictionResults {

   public static String writePredictionResults( int rValue, String timeLine, double accuracy, List<Row>
predictions ) {

      try {


         String OutputDir="C:\\Dataset\\Spark_Predictions_" + rValue + ".csv";

         PrintWriter pw = new PrintWriter(new File(OutputDir));
         StringBuilder fw = new StringBuilder();

         fw.append(timeLine);
         fw.append('\n'); fw.append('\n'); pw.write(fw.toString()); fw.setLength(0);

         fw.append("Test Error = " + (1.0 - accuracy) + "    Accuracy = " + accuracy);
         fw.append('\n'); fw.append('\n'); pw.write(fw.toString()); fw.setLength(0);

         fw.append("Failed Predictions");
         fw.append('\n'); pw.write(fw.toString()); fw.setLength(0);

         /* Write the failed Predictions First*/
         predictions.forEach(row -> {

            if( Double.parseDouble(row.get(1).toString()) !=  Double.parseDouble(row.get(5).toString()) ) {
               String outputLine = "id: " + row.get(0) + " , " + "label: " + row.get(1) + " , "
                     + "prediction: " + row.get(5) + " , "
                     + "features: " + row.get(2) + " , " +
                     "predictions: " + row.get(3) + " , " + "probabilities: " + row.get(4) + "\n";

               fw.append(outputLine);
               pw.write(fw.toString());
               fw.setLength(0);
            }
         });

         fw.append("\n\nAll Predictions");
         fw.append('\n'); pw.write(fw.toString()); fw.setLength(0);

         /* Write all the Prediction Records*/
         predictions.forEach(row -> {

            String outputLine = "id: " + row.get(0) + " , " + "label: " + row.get(1) + " , "
                  + "prediction: " + row.get(5) + " , "
                  + "features: " + row.get(2) + " , " +
                  "predictions: " + row.get(3) + " , " + "probabilities: " + row.get(4);

            fw.append(outputLine);
            fw.append('\n');
            pw.write(fw.toString());
            fw.setLength(0);

         });

         pw.close();
```

```
        } catch (Exception e) {
            throw new RuntimeException(e);
        }


        return  "";

    }
}
```