**University of Moratuwa**

**Department of Computer Science and Engineering**


# CS 4202 - Research and Development Project

## Final Year Project Report

# Web Information Extraction System to Sense Information Leakage

## Project Group - SWIS

L.P.D.S. Chandraweera (130079C)

H.A.I.C Dayarathna (130102T)

W.W.A De Silva (130113D)

K.A.U.Sewwandi (130561X)

**Internal Supervisor**

Dr. H.M.N. Dilum Bandara

**External Supervisor**

Mr. Nalinda Herath

THIS REPORT IS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE OF ENGINEERING AT UNIVERSITY OF MORATUWA, SRI LANKA.

13th of November, 2017

# Declaration

We, the project group SWIS (L.P.D.S. Chandraweera, H.A.I.C Dayarathna, W.W.A De Silva and K.A.U.Sewwandi under the supervision of Dr. H.M.N. Dilum Bandara and Mr. Nalinda Herath) hereby declare that except where specified reference is made to the work of others, the project "Web Information Extraction System to Sense Information Leakage" is our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgement.

Signatures of the candidates:

1. …… L.P.D.S. Chandraweera (130079C)
2. …… H.A.I.C Dayarathna (130102T)
3. …… W.W.A De Silva (130113D)
4. …… K.A.U.Sewwandi (130561X)

| Supervisor: | External Supervisor: |
|---|---|
| …….. | …….. |
| (Signature and Date) | (Signature and Date) |
| Dr. H.M.N. Dilum Bandara | Mr. Nalinda Herath |

# Abstract

With the exponential growth of sensitive data stored in computer systems, data breaches are becoming inevitable. Exposure of such data breaches has also become a major problem where information on a data breach is published on the Internet with the motive of damaging the reputation of data owners. Such data breaches are mostly exposed on online text sharing sites like Pastebin.com and social media sites like Twitter and Facebook. Therefore, early detection of data leakages and evidence of hacking attacks is of prime importance to mitigate potential damages. We address the problem of automated identification of data leakages, as well as classifying and ranking such incidents while maximizing recall and minimizing false positives. We developed an automated, scalable monitoring platform for early detection of data leakages and evidence of hacking attacks. The platform preprocess, filter, classify, and rank the suspected message feeds collected from various sources using machine learning and text classification techniques. Utility of the proposed platform is demonstrated by connecting live Pastebin and Twitter posts, while focusing on data breaches related to Sri Lankan financial and government institutes. The proposed platform is scalable where it can process three pastes per second and 2,500 tweets per second with modest hardware and has an average precision of 90.44% and recall of 97.62%.



**Keywords**: data breaches; machine learning; real-time; social media; text classification

# Acknowledgement

First and foremost we would like to express our sincere gratitude to our project supervisor, Dr. H.M.N. Dilum Bandara for the valuable guidance and dedicated involvement at every step throughout the process.

We would also like to thank our external supervisor Mr. Nalinda Herath for the valuable advices and the direction given to us regarding the project.

In addition, we would like to thank Prof. Gihan Dias and Dr. Kutila Gunasekera for the support, encouragement and insightful comments.

Last but not least, we would like to express our greatest gratitude to the Department of Computer Science and Engineering, University of Moratuwa for providing the support for us to successfully finish the project.

## Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| APT | Advanced Persistent Threat |
| ARFF | Attribute-Relation File Format |
| AUP | Acceptable Use Policy |
| BIN | Bank Identification Number |
| CC | Credit Card |
| CEP | Complex Event Processing |
| CERTs | Computer Emergency Response Teams |
| CID | Card Identification Number |
| CF | Configuration Files |
| CSIRTs | Computer Security Incident Response Team |
| CVC | Card Verification Code |
| CVV | Card Verification Value |
| DA | DNS Attack |
| DB | Database Dump |
| DLP | Data Leakage Prevention |
| DNS | Domain Name System |
| EO | E-mail Only |
| EC | E-mail Conversation |
| IRC | Internet Relay Chat |
| IUO | Internal Use Only |
| JSON | JSON JavaScript Object Notation |
| NCSC | National Cyber Security Centre |
| NER | Named Entity Recognition |
| NLP | Natural Language Processing |
| NLTK | Natural Language Toolkit |
| PAN | Primary Account Numbers |
| PHI | Personal Health Information |
| PII | Personally Identifiable Information |
| PIN | Personal Identification Number |
| PK | Private Key |
| POC | Proof of Concept |
| RSS | Rich Site Summary |
| SIEM | Security Information and Event Management |
| SQL | Structured Query Language |
| UC | User Credentials |

# 1 Introduction

## 1.1 Background

With the digitalization, organizations are keeping all their data in digital form as sensitive data like military secrets, trade secrets enabling easy access, management, and simplified storage. Securing these sensitive data while allowing convenient access to authorized users is a crucial task for any organization. However, various design, implementation, and human errors/omissions enable unauthorized parties to access and expose sensitive data. Such an exposure of data is considered as a *data breach*. Unsecured data will lead to data breaches and can be harmful to the data owner in many ways. When valuable information is taken by wrong people the damages are unpredictable which will be both short and long term.

Data breaches are inevitable today world due to several aspects like data breaches on government information, targeting business parties in avocation of competitive advantage in business, targeting personals through phishing attacks to leverage benefits in case of personal contention. Data Leakage Prevention (DLP) systems are used to prevent unwanted accidental or malicious leakage of sensitive data into hands of unauthorized parties which would help organizations to self-defend their sensitive information.

Many harmful consequences occur when a data breach is exposed, as it can reach to anyone who is treacherous more or less. Such data breach exposures may contain dumps with login credentials, database dumps, configuration files, Personally Identifiable Information (PII), etc. Apart from data leakages hackers leave evidence of hacking attacks. The motives behind full or partial data exposure includes damaging the reputation of the data owner, improving the reputation of the hacker among hacker communities, and as a way of proving that the attackers has access to data to potential buyers.

## 1.2 Motivation

Commercial Bank of Sri Lanka was hacked and its data were published online on May 12, 2016 by the Bozkurtlar hacking group. Same group has also posted seven other data dumps from banks in the Middle East and Asia since April 26 [1]. Twitter was the first to reveal the news and only through that Commercial bank got to know about their own data exposure. By

that time the exposed data has been reached to everyone around the Globe. This caused a major damage to the reputation of Commercial Bank [1].

These data breach exposures mostly happen via text sharing websites like Pastebin applications and social media sites such as Facebook, Twitter, Google+, and LinkedIn. Among Pastebin applications Pastebin.com [2] is the widely used text sharing site by hacker communities to expose data breaches. The targeted entities are clueless about the data breaches and exposure until stolen data or the evidence of hacking attack is exposed in some media like Twitter. Data leaks and evidence of attacks of more than hundred Sri Lankan organizations including financial companies and educational institutes were publicized via Pastebin and Twitter in the recent past [3], [4], [5].

In effective data breach incident response, early detection of data leakages is of higher priority. An automated, effective, and scalable monitoring platform for early detection of data leakages and evidence of hacking attacks in Pastebin applications and social media sites could speed up the aforementioned incident response. That platform should effectively automate identification of data leakages and evidence of hacking attacks, as well as classification of retrieved data. This platform should be customizable to identify data leakages and evidence of hacking attacks related to a given domain, from an individual to a mass scale data breach in real time. Real-time identification is stressed here to minimize further damages happened to the organization through the data breach exposure. Classification of retrieved data is of higher priority because a data leakage or an evidence of hacking attack need to be identified precisely.

Following are some scenarios that depicts the significance of an early detection platform for data leakages and evidence of hacking attacks.

***Scenario one:*** *A dump file of user emails and passwords posted as a tweet*

*In such a case the platform should identify the affected parties and should notify them promptly so that they can take actions to change user credentials and reduce the further damages. This is very significant as putting user credentials in some other person's hand would cause a severe damage. Once this happened to Commercial bank of Sri Lanka [5]. If such a system was in place, news would not go far and could have reduce the damage caused to the reputation of the bank.*

***Scenario two:*** *A credit card dump posted in Pastebin with CVV2 and other sensitive data*

Credit card fraud is a major source of financial losses in today's world which is an unrecoverable loss for the individuals. Hackers used to publish the credit card dumps on internet mostly through Pastebin. The Bank Identification Number (BIN) of the credit card numbers are matched to identify the issuing banks of the breached accounts. Also the bank can remove the related content with credit card numbers from Pastebin by reporting the incident to the website administrators.

Early detection of data leakages and evidence of hacking attacks, and immediate response by the data owner reduces escalation of damages. Most of the organizations currently have various manual methodologies to detect data breach exposures through Pastebin sources and social media which causes to spend significant amount of time and effort. Therefore, it is imperative to have an automated early detection platform for the above addressed problem.

LeakHawk 1.0 is such a solution proposed by Nalinda Herath [6]. A working model of the proposed platform was implemented as a proof of concept. The PoC monitors www.pastebin.com, the mostly used Pastebin application, for sensitive information leakages and evidence of hacking attacks related to Sri Lanka. LeakHawk 1.0 is more focused on the depth of the problem rather than the breadth. It considers all the Pastebin feeds as textual content and use text-engineering techniques to categorize the content. To cover the breath, it is required to integrate social media feeds. Apart from that the sub-modules in LeakHawk 1.0 needs to be improved to gain higher recall value and minimize false positives, while being scalable enough to handle large number of messages.

## 1.3 Problem Statement

The problem addressed by this project as be formulated as follows:

*In the event of a data leakage how to classify/rank such incidents while maximizing recall and minimizing false positives?*

In a situation where sensitive information belongs to a particular party is exposed through Internet, there should be a monitoring platform or some mechanism to identify them promptly. The system should not eliminate any sensitive data leakage or evidence of hacking attack as false negatives and should minimize the number of false positives to reduce the management overhead. The accurately identified content should be analyzed and classified/ranked based on the severity of the data breach. The severity ranking should be based on classifying the

content of the post as critical, high, or low. In a scenarios where the leaked content is not available, but an evidence of a data breach or a hacking incident is available, the system should able to identify them as well to improve the accuracy.

The development of a solution for data leakage incident detection can be depicted as a text classification research problem of non-structured and semi-structured data, as contents of the posts in Pastebin and Twitter are textual inputs. Therefore, for the text classification problem could be addressed using machine learning techniques and rule-based methods. The textual input needs to be preprocessed only to seek the significant textual content by removing unwanted text like stop words in a language. The semantics in preprocessed textual input is extracted and the severity of the content is ranked and predicted from the extracted semantics.

To improve the accuracy and reduce false alarms all the posts from a given source should be taken into account. This requires scalable stream processing techniques. Although stream handling is done properly some posts can still be missed out due to variable delays in the system and network. To handle this incorporation of message queuing tools and techniques are required. Because textual inputs from different data sources are in different formats and lengths, filtering and classification may need to be customized for each data source. Some of the posts may contain one or more URL(s) which may be an evidence of hacking attack which needs to be explored further to find whether the content of that post contains a data leakage or an evidence of hacking attack. This leads to improve accuracy and reduce false alarms as most of the aspects that a textual input in post can be checked are considered. Therefore, it is important to process both a single post and a chain of linked posts.

## 1.4 Research Contributions

We make the following research contributions:

- Designed and developed a real-time, automated, scalable platform for early detection of data leakages and evidence of hacking attacks related to a user defined domain. The proposed platform is named LeakHawk 2.0.
- The proposed platform is scalable where it can process 3 pastes per second and 2,500 tweets per second with modest hardware and has an average precision of 90.44% and recall of 97.62%.

- The proposed platform is customizable where several data sources like Facebook and Google+ could be integrated and direct for further processing.
- An open source implementation for pastebin.com and Twitter for real-time identification of data leakages and evidence of hacking attacks.
- We further developed a text corpus of Pastebin and twitter posts that can be used for further researches in the information security domain.

## 1.5 Outline

Rest of the report is organized as follows. Chapter 2 discusses existing literature relevant to the project. Design of the system and its architecture are presented in Chapter 3. Chapter 4 presents the implementation details of the project including the tools and technologies, system components and dashboard. Performance analysis of LeakHawk 2.0 and comparison with earlier system is presented in Chapter 5. Chapter 6 concludes the report with problems encountered, challenges, and future work.

# 2 Literature Review

In this chapter we formulates the background information and existing literature related to the research problem. We first present an introduction to data breaches and evidence of attacks in Section 2.1. Furthermore, it presents the criticality of exposing the evidences of hacking attacks and sensitive data leakages on the Internet. Section 2.2 provides a brief history of security incident exposures related to Sri Lanka and other countries. Section 2.3 analyze Pastebin and Twitter in terms of architecture, features and limitations with respect to security incident monitoring. A discussion on the existing monitoring systems and their capabilities are presented in Section 2.4. Section 2.5 discusses real-time stream handling techniques that can be applied to handle large data feeds. Section 2.6 analyzes the text classification methodologies and how they can be incorporated to the design of LeakHawk.

## 2.1 Data Breaches and Evidence of Hacking Attacks

A *data breach* can be defined as an incident that involves the unauthorized or illegal viewing, access or retrieval of data by an individual, organization or a country. It is a type of security breach specifically designed to steal and/or publish data to an unsecured or illegal location. A data breach occurs when an unauthorized person accesses a secure database or a repository. A data breach may result in data loss, including financial, personal and health information. A hacker also may use stolen data to impersonate himself to gain access to a more secure location. For example, a hacker's data breach of a network administrator's login credentials can result in access of an entire network [7].

A data breach can be carried out unintentionally or intentionally. An unintentional data breach occurs when a legitimate custodian of information such as an employee loses or negligently uses corporate tools. An employee who accesses unsecured websites, downloads a compromised software program on a work laptop, connects to an unsecured Wi-Fi network, loses a laptop or smartphone in a public location, etc. runs the risk of having his company's data breached. In 2015, Nutmeg, an online investment management firm, had its data compromised when a flawed code in the system resulted in emailing the personally identifiable information (PII) of 32 accounts to the wrong recipients. The information that was sent out included names, addresses, and investment details and put the account holders at risk of identity theft [8].

An intentional data breach occurs when a cyber attacker hacks into an individual's or company's system for the purpose of accessing proprietary and personal information. Cyber hackers use a variety of ways to get into a system. Some embed malicious software in websites or email attachments that make the computer system vulnerable to easily enter and access data by hackers. Some hackers use botnets, which are infected computers, to access other computers' files [8].

A data breach can be harmful in many ways. Once the sensitive information is put in wrong hands, the consequences are unpredictable. For example, it could put an entire nation at risk of a terrorist attack or an organization may have to pay a huge penalty or lose its reputation, damaging its competitive advantages. An individual who is subjected to a credit card breach may lose a significant amount of money via unauthorized transactions [9].

Owners and users of a breached system or network don't always know immediately when the breach occurred. In 2016, Yahoo announced what could be the biggest cybersecurity breach yet when it claimed that an estimated 500 million accounts were breached. Further investigation revealed that the data breach had actually occurred two years prior in 2014.

While some cyber criminals use stolen information to harass or extort money from companies and individuals, others sell the breached information in underground web marketplaces that trade in illegal assets. Examples of information that are bought and sold in these dark webs include stolen credit card information, business intellectual property, SSN, and trade secrets [8].

### 2.1.1 Evidence of attacks

Apart from sensitive information, hackers publish evidence of attacks via social media and text sharing sites. In most cases, results of politically motivated attacks and Hacktivist movements are posted online to embarrass the targeted entities [6].


*Figure 2-1: Data Breach exposure via Twitter*

Notably, attacks such as web site defacements, DDoS attacks, SQL Injection attacks, and DNS related attacks (zone transfers and cache poisoning) are exposed. Some examples of evidence of hacking attacks are illustrated in Figure 2.1, Figure 2.2 and Figure 2.3.


*Figure 2-2: Commercial Bank attack exposed via Twitter*

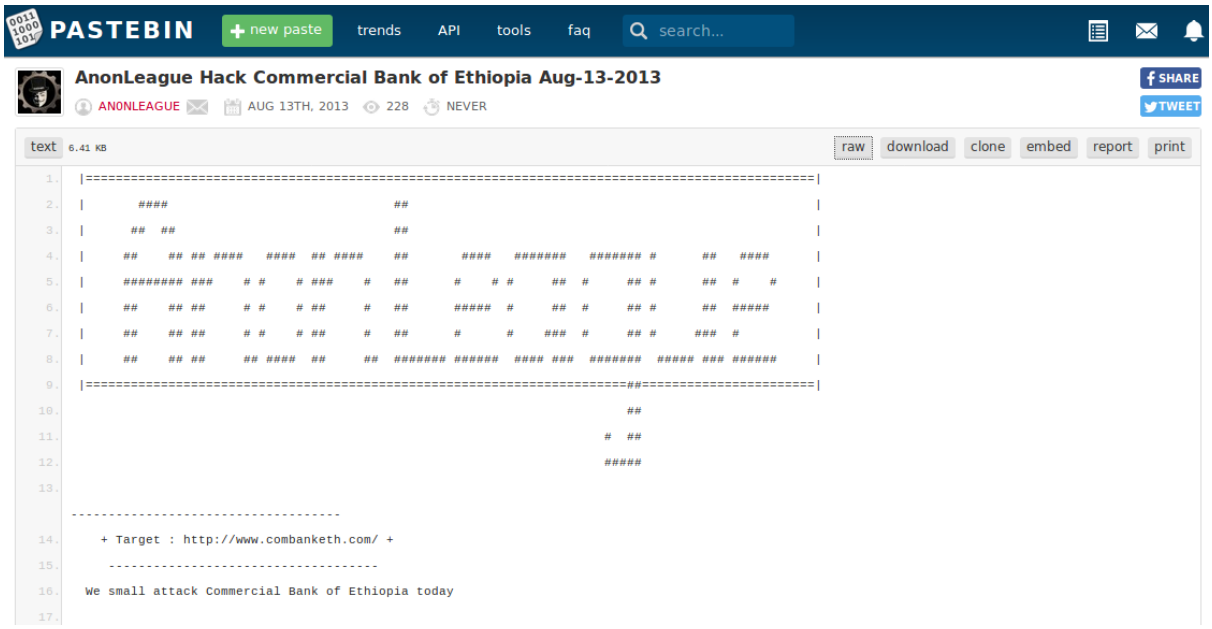


*Figure 2-3: Commercial Bank attack exposed via Pastebin*

### 2.1.2 Making Data Breaches and Hacking Incidents Public

The majority of successful companies of today are well aware of common data security issues and they put a great deal of trust into their own efforts towards preventing a data security breach. However, as demonstrated by recent security breaches of several large, tech-savvy companies no set of security measures is completely infallible to a breach.

For organizations that own critical information assets such as customer data, intellectual property and proprietary corporate data, the risk of a data breach is much higher. When it comes to government and military-related entities, risk of a data breach becomes more and more critical. Even the organizations that do not have very sensitive information under their repositories, but maintains a good online presence, will be under great dissatisfaction with respect to their reputation. For example, the primary website of a renounced non-profit organization can be defaced by a Hacktivist group which eventually poses a severe damage to their reputation [6].

Hackers expose the stolen data with several intentions. Sometimes cyber criminals are paid by some parties to compromise the infrastructure and data owned by their opponents. While the breached data is being used for various misconducts, once the utilization of those data is done or the value of the data is expired, attackers tend to publish the content on the Internet to carry out further damage to the reputation of the data owner or the organization. For instance in the recent data breach of one of the major private banks in Sri Lanka, the published content did not affect a direct financial loss, but greatly impaired the reputation of the bank [1]. Alternatively, a successful penetration of security parameters of a renowned organization could significantly improve the status of a hacker who conducted the attack. Revealing the stolen content will prove the hacking attack and the attackers will be endorsed among the hacking communities.

Some attackers target the vulnerabilities of popular websites to let them know about the lack of controls and the security holes of their external security system. The motive behind such attacks is not malicious, but exposing the vulnerabilities into public channels will violate the white-hat security principles. Some hacking incidents are done and made public by the attackers just for their own pleasure.

## 2.2 Data Leaks Related to Sri Lanka

Organizations that store and process sensitive and valuable trade and market information, client information and transaction history data, continues to be at the top of potential targets for cyber criminals who probe, scan and penetrate the IT infrastructure of these organizations to carry out massive cyber-attacks.

For years, cyber warfare has been used to conduct destruction against governments, officials, public and private corporations. Cyber warfare has targeted missile guidance systems, power grids, nuclear reactors and more [10]. Although not being an iconic character in the cyber warfare, Sri Lanka has suffered numerous hacking incidents which have been exposed via online channels. In 2011, a series of attacks were carried out by a hacker group called AnonymousSriLanka targeting a set of government institutes, educational institutes, and Internet Service Providers [3]. These attacks were politically motivated and identified as an outcome of anger towards Sri Lanka after the eradication of LTTE terrorists. In 2013, another set of attacks were conducted against a set of online targets belong to Sri Lankan organizations [4]. Apart from these major incidents, some ad-hoc sensitive information dumps and evidence of hacking incidents have been posted in online channels time to time. In general, around 80% of the reported events are exposed via Pastebin applications. Most out of the remaining incidents are exposed via social media feeds. Recent incident targeting one of the major commercial banks in Sri Lanka was exposed via Twitter Feed [1] and Pastebin. The dump contains 158,276 files in 22,901 folders and is about 6.97 GB uncompressed. The compromised data contains annual reports, application forms, bank financial statements, .PHP files, web development backups and other files needed for the functioning of the bank's corporate front-end web infrastructure. The attackers appear to have compromised the bank's systems using a SQL injection attack and uploading a web shell - a script that enables remote administration - onto the bank's PHP server.

## 2.3 Pastebin and Twitter

A paste is defined as a textual content posted onto a website where it receives its unique URL so that it can then be shared to access the paste. The contents of a paste could be a game chat, a programming code chunk, configuration file, a recipe, a dump of leaked information. Text sharing sites allow users to post pastes and allow for public viewing. These simple websites provide the users an easy interface for creating, managing and sharing textual content via

multiple channels. These web applications were originated to assist Internet Relay Chat (IRC) to share a large amount of texts between users using the unique URL provided by the website [6].

### 2.3.1 Pastebin

Pastebin is a popular website for storing and sharing text. Though it's mostly used for distributing legitimate data, it seems to be frequently used as a public repository of stolen information, such as network configuration details and authentication records. Various hacker groups and individuals also use Pastebin to distribute their loot the highest—a trend perhaps initially set in motion by LulzSec [11].

www.pastebin.com was the first Pastebin application which was developed in 2002 [2]. It is the most popular Pastebin among the programmers as well as hacking communities. First security information breach on Pastebin was reported in 2009 when roughly 20,000 compromised Hotmail accounts were disclosed in a public post. Being simple, reliable and easy-to-use, text sharing websites such as Pastebin allows their users to even anonymously publish documents online and keep them valid for a longer time span. These are ideal conditions required by hacker groups to publish sensitive information on the Internet. The properties of Pastebin that causes it to become popular among hacker community are ease of use, non-authentication to post anything and allows sharing long text messages without limiting content [6].

#### 2.3.1.1 Pastebin Structure

This section describes the attributes and functionalities of www.pastebin.com regarding the importance of monitoring for sensitive information leakages and evidence of hacking attacks. Figure 2-4 illustrates the homepage of Pastebin and Table 1 describes the each attribute of the main interface in detail. Trending Pastes page allows the users to view the pastes with most hits [12]. It can be customized to display popular pastes at different times such as right now, last seven days, last 30 days; last 365 days and all time. Figure 2-5 shows the trending pastes in the last month. As seen in the figure, almost all the pastes are apparently related to a data leak or hacking incident. Public Archive or the Paste Archive page lists all the newly added pastes on a single page [13] as shown in Figure 2-6. If anyone is interested in scrapping Pastebin for data leaks or hacking notifications, he/she will need to monitor this page. However, the application does not allow the users to make too many requests. Such IP will

be blacklisted for few hours. Most of the Pastebin applications follow the same behavior and that is one of the hurdles in building Pastebin monitoring tools [6].

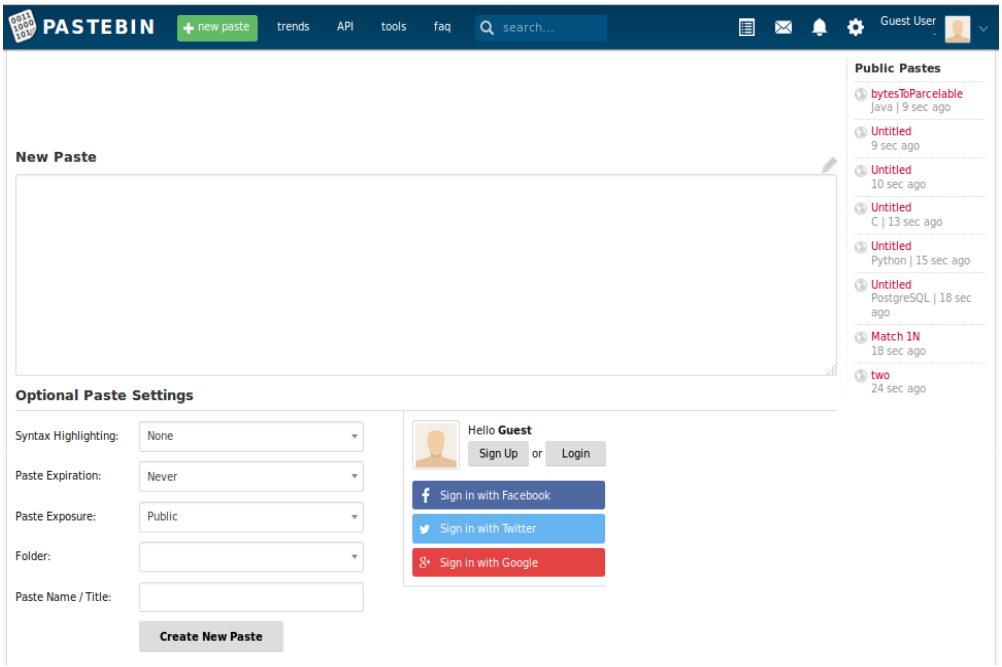


*Figure 2-4: Pastebin Main Interface*

*Table 1: Attributes of Pastebin main interface*

| **Section** | **Description/Importance** |
|---|---|
| Trending pastes | Trending pastes lists the most frequently accessed pastes by all the users. Mostly this section lists leaked data from popular targets as such data will attract a lot of attention. |
| Pastebin API | Pastebin provides an API for the users to publish their posts conveniently. It also provides a scraping API (paid service) for searching and downloading pastes. |
| Pastebin alerts | Pastebin allows the users to provide a set of keywords and be notified via e-mail when a post is made containing any of those keywords. |
| Text insert area | This area will contain the text dumps. Normal users can post data up to a maximum size of 512 kilobytes; PRO users can paste up to 10MB. A single paste can accommodate considerably a larger text dump which is one of the reasons paste sites are used by hacking communities to dump their data. |
| Pastes by the user | Lists the pastes made by the logged-in user. |
| Public pastes | This section is called the Pastebin Archive. It is frequently being updated with all the public pastes made by all users. If someone is interested in monitoring the Pastebin real time for leaked data, he will be required to focus on the content of this page. |

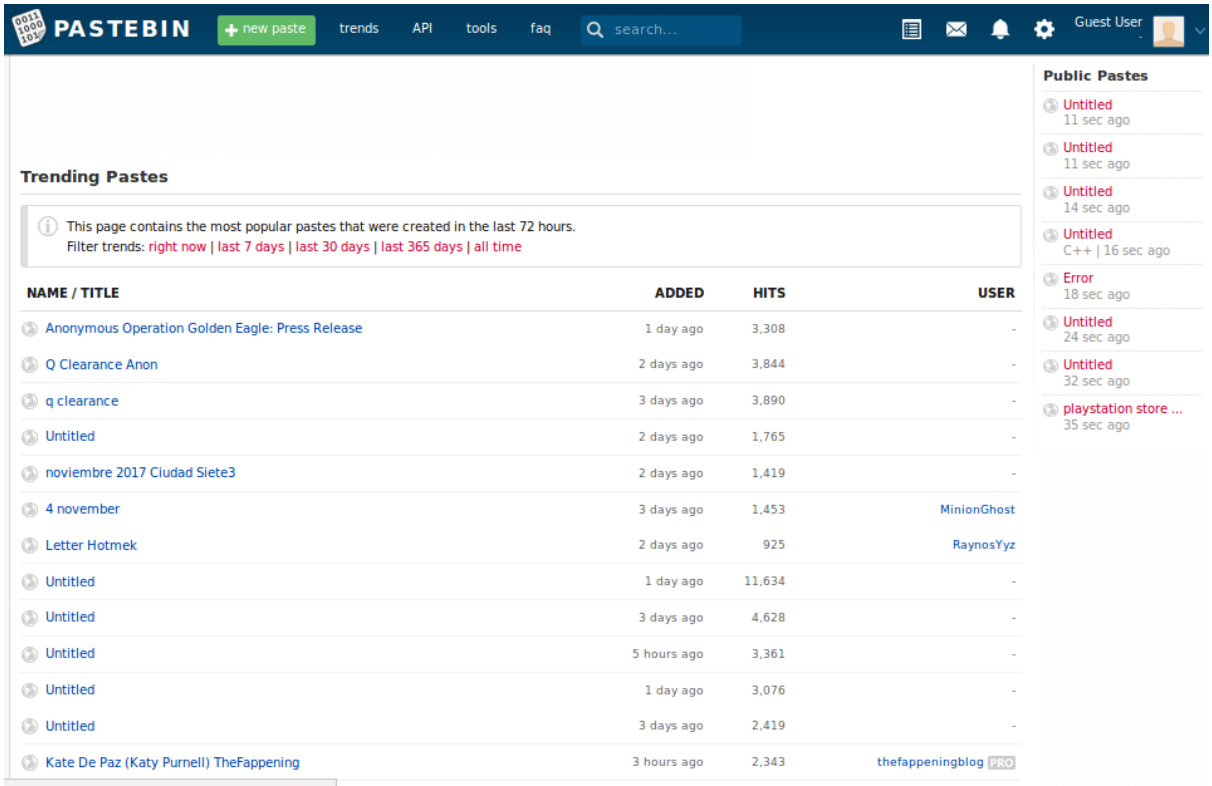


*Figure 2-5: Trending posts page - Pastebin [Snapshot was taken on 8 Nov 2017]*

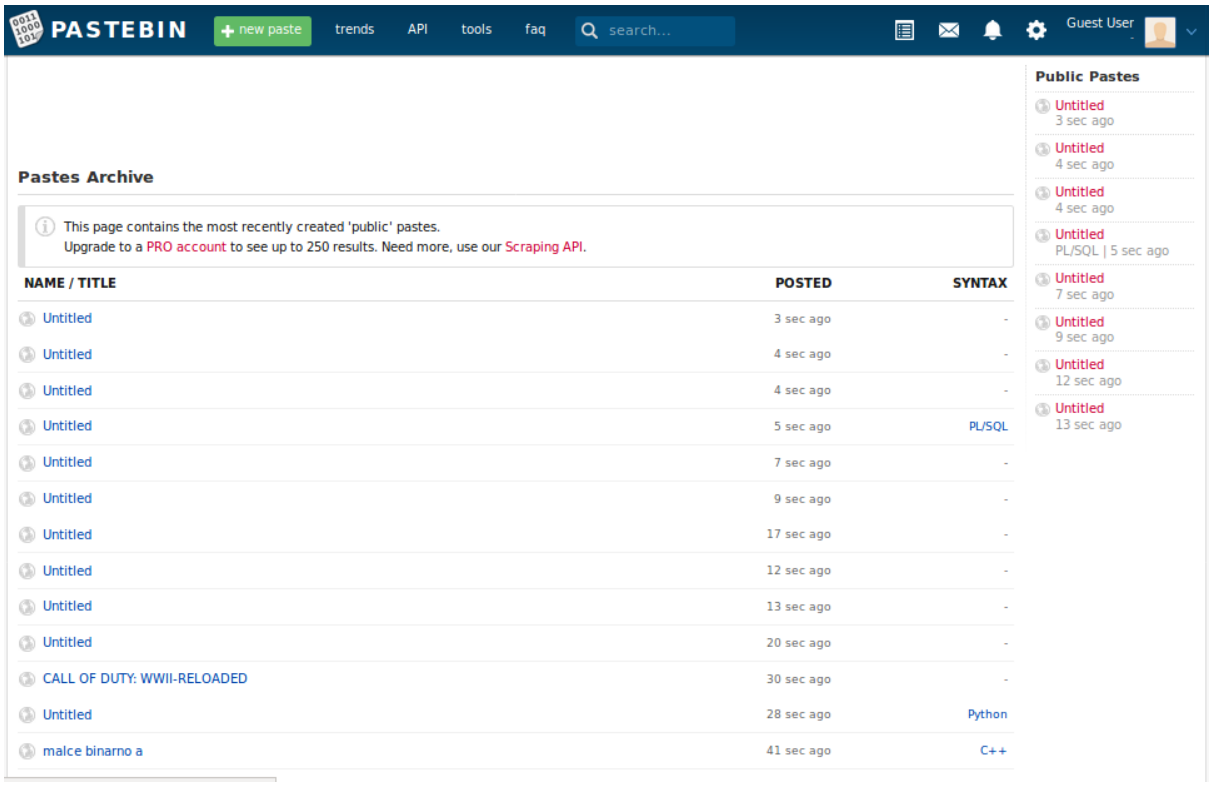


*Figure 2-6: Site structure of Pastebin public archive page [pastebin.com]*

Figure 2-7 is a paste/post that was published on Pastebin by user ANONYMOUSSRILANKA regarding a data breach of University of Moratuwa Sri Lanka.

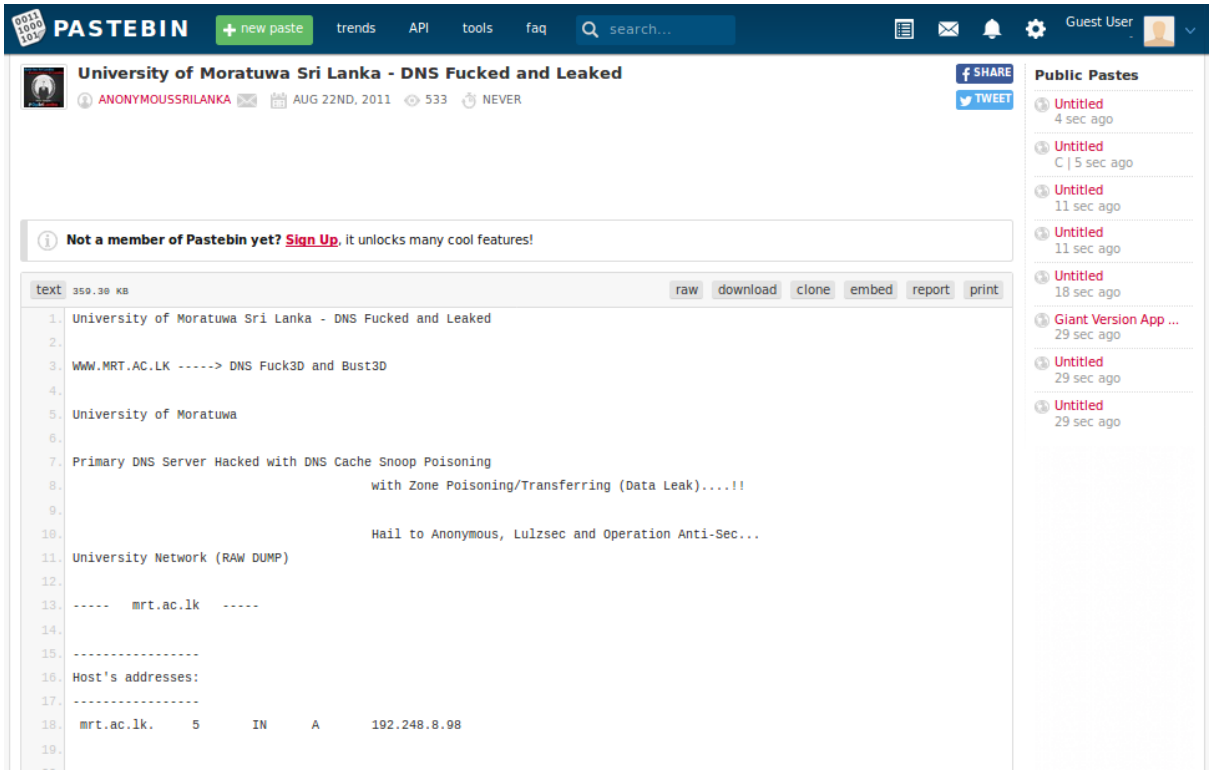


*Figure 2-7: Data dump posted on Pastebin*

Although the Pastebin is frequently being misused for posting breached data, hacking notifications, login credentials, pornographic content, website does maintain an Acceptable Use Policy. Pastebin makes it clear that posting personal data, email lists, login credentials are against the AUP and will result in its removal. However, with the amount of posts being made per day, the site administrators depend on the abuse reports submitted by the users for content removal, rather evaluating each paste. However, the other Pastebin applications may be less accommodating, which require commercial or legal motivation for content removal and to retrieve origin information to support forensic investigations [6].

### 2.3.2 Twitter

A tweet is a short text message posted by users on Twitter which is limited by 140 characters and allows user's followers to view the tweet. If a user likes to have other's posts on their timeline, he is called a follower. Twitter has been used as a medium for real-time information dissemination and it has been used in various brand campaigns, elections, and as a news media. Since 2006 when its launch, Twitter has an increasing popularity and as of August 2013 about 500 million tweets are being generated every day and 200 billion tweets annually

[3]. When a new topic becomes popular on Twitter, it is listed as a trending topic which is a short phrase or a hash tag. The following are some identified reasons why hackers prefer social media for data leakage:

- Easy login to social media sites – Anyone can create an account and use the credentials to login to the site and no validation on provided information.
- Access to social media available on limitless devices - Apps have been created for easy login which makes the environment for hackers to leak data.
- Large number of users - Since there's a large number of users all around the world it is easy to publish a message so that a large social group can view it when the post is made public.
- Upload photos/videos and files - Unrestricted photo albums and videos allow everyone to view the photos and videos that are potentially sensitive to organizations which makes a preferable chance for hackers.
- No limit on number of posts - Any user can post anything anytime in any number of times a day. No mediator to validate a post unless it is reported by a user. Hackers can not only leak data but can express bad things about an organization with the motive of harming the reputation of the organization. Although there's a character limit in Twitter, there's no limit in Facebook where any long post can be posted.

These facts create ideal conditions for hackers to leak data through social media like Twitter.

#### 2.3.2.1 Twitter Structure

This section describes the attributes and functionalities of www.twitter.com regarding the importance of monitoring for sensitive information leakages and evidence of hacking attacks. Figure 2-8 illustrates the homepage of twitter and Table 2 describes each attribute in detail.

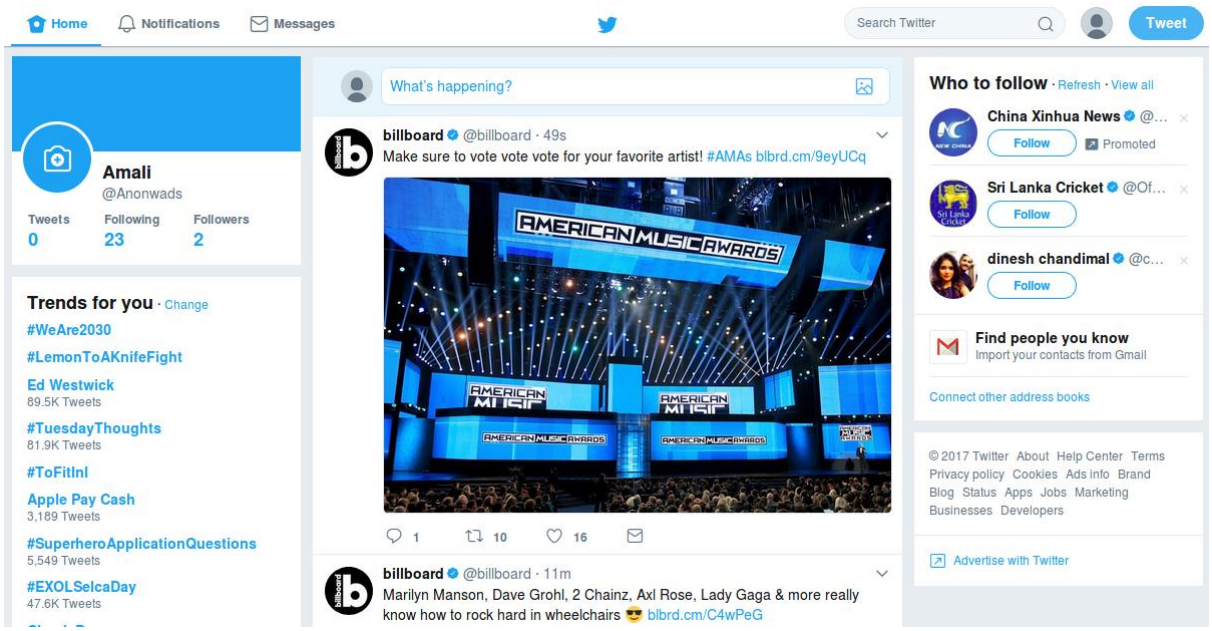


*Figure 2-8: Twitter Homepage*

*Table 2: Features of a Tweet*

| **Section** | **Description/ Importance** |
|---|---|
| Trends | Trends section contain the top ten current trending topics of tweets. Hackers could use one of this trending topics to tweet some hacked data and it'll reach an enormous number of users. |
| Tweet | This button can be used to post a new tweet in Twitter which allows to add photos and videos to the tweet. |
| Twitter API | Twitter REST API and Streaming API can be used to public tweets that are posted with all the metadata related to the tweet. |
| Twitter news feed | The news feed contain the tweets of followers of the logged in user. |
| Suggestions | The suggestions shown in top right corner can be used to follow any interested parties and get their tweets to our timelines. |
| Notifications | Notifications are shown when some user started following the logged user. |
| Messages | This tab can be used to send direct message to some authenticated user. |
| Following | This shows how many users that we are following |
| Followers | This shows how many users follow us |

Figure 2-9 illustrates tweets that were on Twitter by regarding hacking attacks on a private bank and NIBM Sri Lanka.

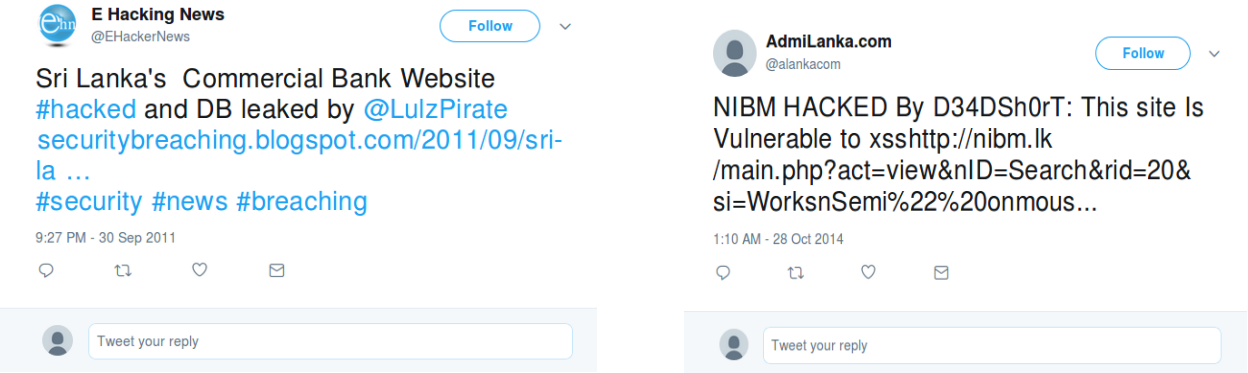


*Figure 2-9: Tweets regarding hacking attack in Sri Lanka*

## 2.4 Existing Monitoring Systems

### 2.4.1 Facebook Monitors Pastebin for Leaked Credentials

Facebook has started monitoring Pastebin and other text sharing sites after the incident of leakage of 700,000 Dropbox credentials with emails and passwords [14]. This process was initiated to monitor leakage of credentials of Facebook users [15].

This monitoring is not only on Facebook credentials of those users, since the same password is used across several websites this system monitors all of them. When an email password pair dump is found on a text sharing site this system automatically check them with the user database of Facebook. Since this only monitors email password credential leaks this is not extendable or customizable to monitor other sensitive content. The underlying architecture is not made open source here.

### 2.4.2 Haveibeenpwned.com [HIBP]

Haveibeenpwned is a monitoring platform that allows users to check whether their personal data has been exposed. HIBP also allows users to sign up and get notified if their personal data is compromised in future through data breaches. This system keep track of data breaches happened in Internet and stores them in the database so that users can query later and check whether their data has been compromised. This too only allows users to check against credentials no other things like credit card dumps, configuration file dumps etc. which are frequently been pasted in Internet. HIBP highly depends on DumpMon, a twitter bot that

monitors Pastebin for possible data leakages [16]. A scalable architecture is not found here in HIBP.

### 2.4.3 Pastefind

Pastefind monitors Pastebin for new pastes which is a python script and source code is available in [17] which is not currently maintained or managed by the developer. Due some recent changes in Pastebin pastebinfind.py is not functioning as expected. Pastefind allows users to set a time parameter for the time period between two requests made to Pastebin since Pastebin black lists IPs which make frequent requests through the APIs.

### 2.4.4 Google Alerts and Google Custom Search

Google alerts can be used to monitor pastebin.com [18] which is not that efficient because it depends on the indexing of Google search engine. Google search in addition can be used to monitor Pastebin by using the accurate queries.

### 2.4.5 PasteMon

PasteMon [19] was initially developed in python which was later rewritten in Perl. Pastemon.pl runs as a daemon in the background and monitors Pastebin for a sensitive content. PasteMon utilizes keyword based rules and regular expressions to identify possible data leakages in Pastebin sites. PasteMon itself has a decent recall and it introduces a large number of false positives as the output from the system.

### 2.4.6 LeakedIn

LeakedIn [20] monitors Pastebin based on PasteMon script which was initially developed to give a look and feel to users on data breaches. This wholly covers a better breadth by considering a larger scope of data breaches while introducing a considerable number of false positives to the system. LeakedIn utilizes regular expressions for the processing.

### 2.4.7 DumpMon

DumpMon is a Twitter bot that monitors Pastebin sites [16] to identify sensitive information leakage. DumpMon uses regular expressions to process the textual input where it monitors sensitive content related to account/database dumps, Google API Keys, SSH private keys, Cisco Configuration Files, and Honeypot Log Dumps.

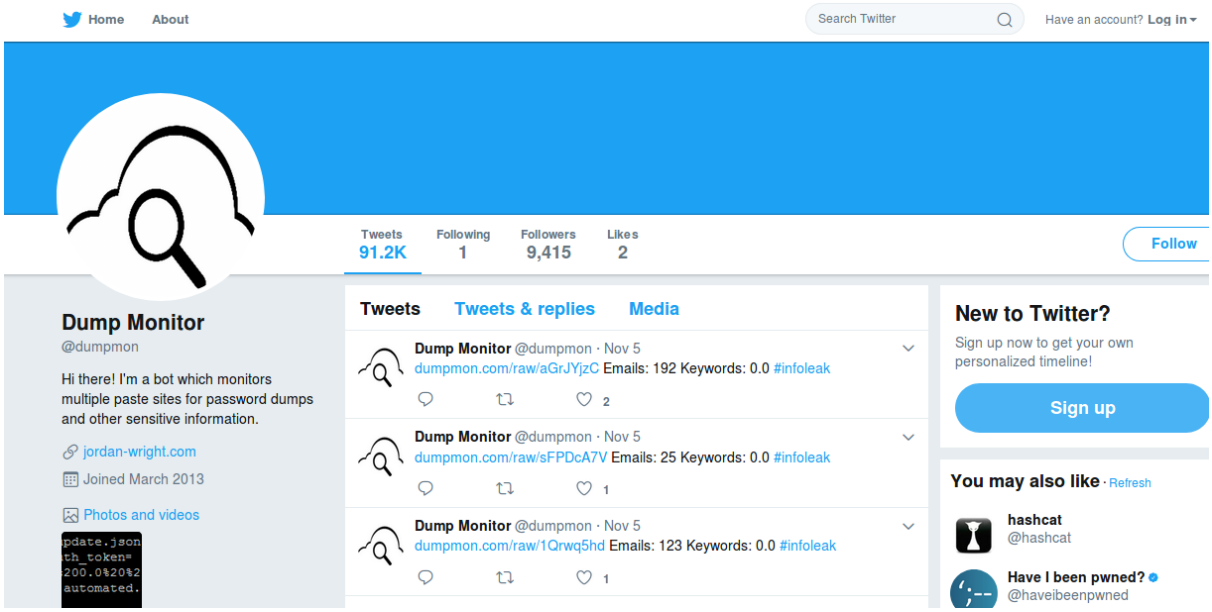


*Figure 2-10: DumpMon Twitter account*

DumpMon monitors Pastebin sites for sensitive data and maintain a multithreaded environment by enforcing a thread for each site to monitor new pastes. Once a possible data leakage is found it posts a tweet in Twitter about the possible data breach.

DumpMon introduces a large number of false alarms to the system which brings the need of a multilayered architecture with several filtering layers to improve accuracy, precision and recall.

A set of tweets on possible information leak is illustrated in Figure 2-11.

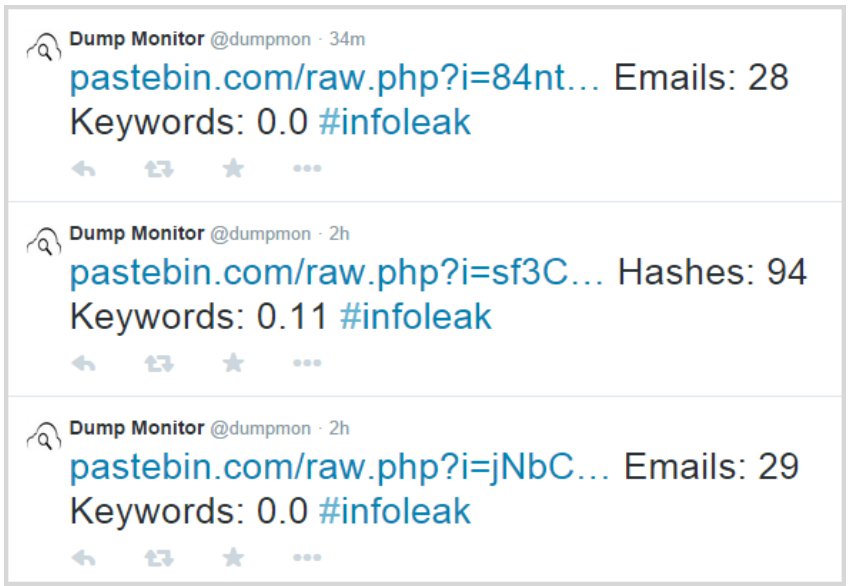


*Figure 2-11: DumpMon tweets on possible information leaks*

The DumpMon architecture can be illustrated as in Figure 2-12 which is a multithreaded environment.

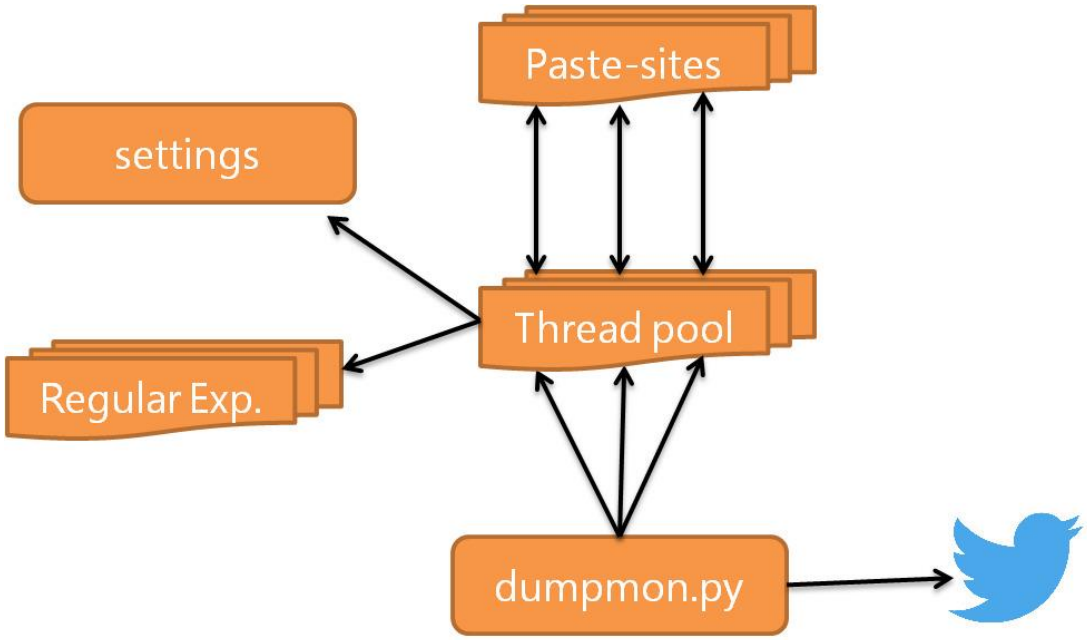


*Figure 2-12: DumpMon architecture*

### 2.4.8 LeakHawk 1.0

LeakHawk 1.0, the first version LeakHawk 2.0 is a Proof of Concept that leverages pattern based and machine learning based methodologies to detect data leakages and evidence of hacking attacks by monitoring Pastebin [2]. It has addressed the same as the problem that is addressed in this research which is "In the event of a data leakage, how to identify and classify/rank such incidents while maximizing recall and minimizing false positives". LeakHawk 1.0 follows a layered architecture as shown in Figure 2-13.

Connectors are used to monitor and access new pastes made in Pastebin and feed them to aggregation layer. At the aggregation layer the entered data are preprocessed and aligned to feed to classification layer of the system. Classification layer is the core of the platform and all text processing and analysis is done here. Database layer stores the retrieved data along with the metadata which is fed into the classification layer for processing. It stores domain information which LeakHawk is configured to monitor. Also it maintains administrative contacts of data owners to notify about identified data leakages and evidence of hacking attacks. When a security incident is predicted, Notifier alerts the respective data owners via the configured methods (e.g. email and SMS).

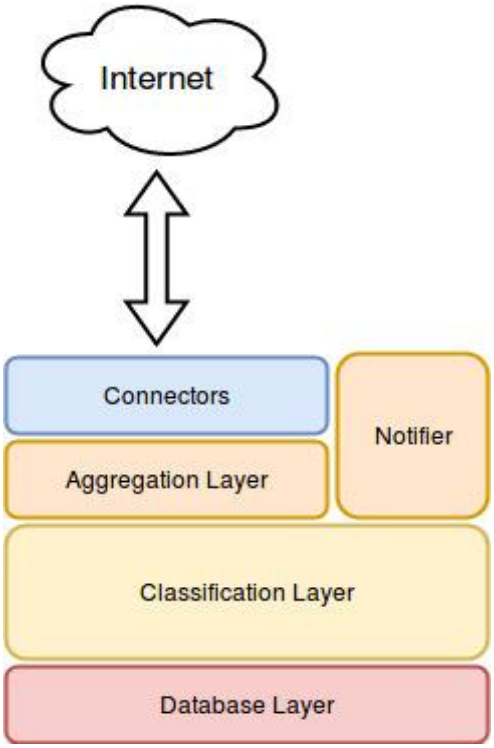


*Figure 2-13: Layered architecture of LeakHawk 1.0*

The high level architecture design of LeakHawk 1.0 is shown in Figure 2-14.

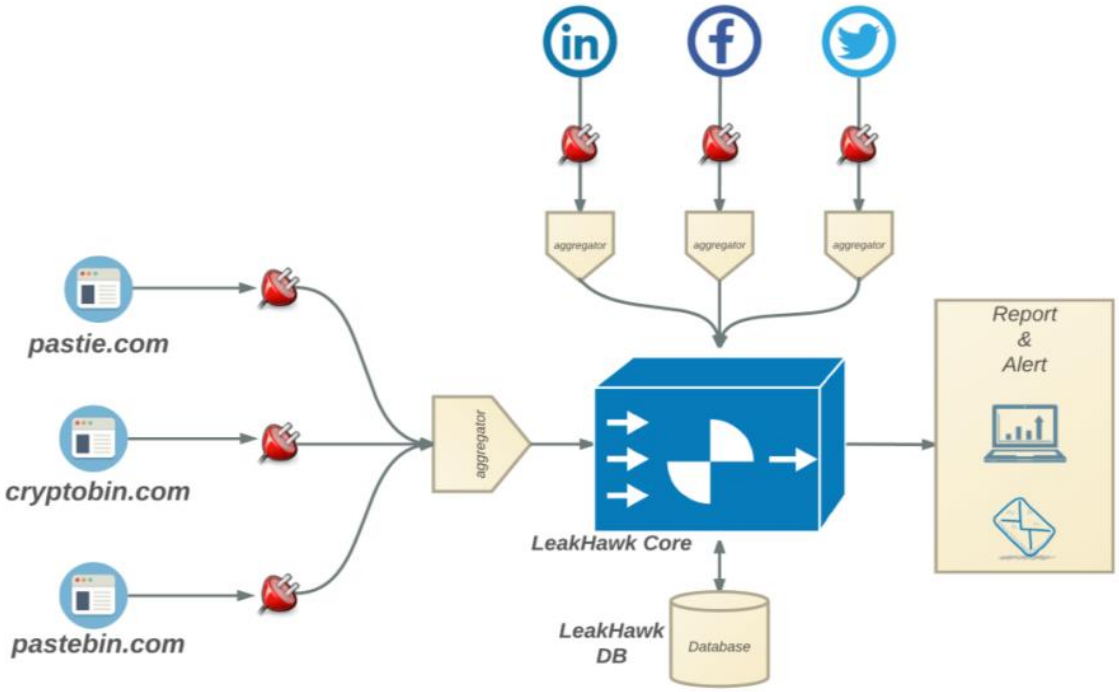


*Figure 2-14: High level architecture design of LeakHawk 1.0*

Connectors incorporates multiple feeds from different data sources (e.g., Pastebin applications, Twitter feeds, etc.) by keeping an uninterrupted connection with the particular data source and aggregator aggregates them into the primary classification engine, the LeakHawk Core. There are different connectors for different data sources while one aggregator is used to aggregate all inputs from different data sources. The combination of a connector and an aggregator is known as a sensor. So the relevant sensor will be notified when a post is made in the particular data source and download and feed it into the internal core.

The sensor used to retrieve posts from Pastebin is the Pastebin sensor which is a java based application that is used to retrieve all the new posts promptly from pastebin.com site. New pastes are downloaded and stored in LeakHawk database along with metadata of the post. The implementation of Pastebin sensor satisfies non-functional requirements like timeliness, comprehensiveness, non-violation of Acceptable Use Policy (AUP) of Pastebin. In order to query new posts from Pastebin scraping Application Programming Interface (API) can be used with java. A normal user cannot access all the posts with a given efficiency since Pastebin doesn't allow that and blacklist the user. So to have a particular efficiency in retrieval need to be a PRO member of Pastebin.

The generic classification engine is made to work independently to ensure loose coupling between the modules of the platform which increases reusability and modularization of the system. There are sub-modules within the LeakHawk Core as shown in below figure. They will classify each textual input into one or more predefined classes and classify them according to a rule-based mechanism designed for each class.

The component architecture of LeakHawk 1.0 is shown in Figure 2-15

LeakHawk Core is the primary processing engine of the monitoring platform. The submodules of LeakHawk core are Pre Filter, Context Filter, Evidence Classifier, Content Classifier and Synthesis. After aggregators aggregating the textual input into classification engine it is fed into the Pre filter. The primary objective of Pre filter is to filter-out non-sensitive data inputs like code snippets, game chat sessions, pornographic content, torrent information, non-textual pastes and trial and empty pastes. Here Pre filter, Context filter and Evidence classifier uses keyword based and regular expression based approach while Content filter uses both pattern based and machine learning based approach. Pre filter screens out non-related input posts which reduces processing overhead in next filters and classifiers. The post types that needs to be screen out were identified by analyzing the training corpus retrieved from Pastebin. Some

preprocessing techniques were used in the Pre Filter which reduces further overhead of processing. The usage of Pre Filter is made optional since it brings a lot of false positives is of less usage in this context.

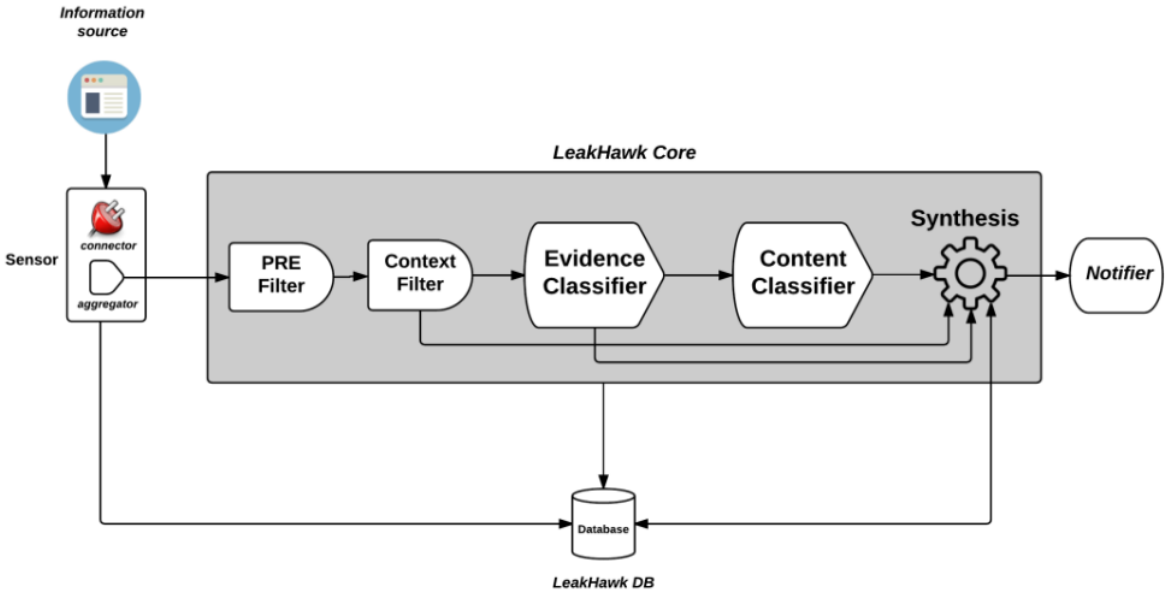


*Figure 2-15: Component architecture of LeakHawk 1.0*

The context filter is used to screen out non context related information and extracts only the input documents related to the context the system is focused on. The context defines the information regarding a particular organization, nation or an individual that is unique for each entity. If LeakHawk is utilized by an individual, he/she can configure a template for the context containing his/her unique information domain. The filtered data from Pre filter is sent to Context Filter to check whether it contains data related to defined domain or context. The information domain is defined by a user or an administrator with their preferred domain information which defines which needs to be filtered into the system for further processing.

The Evidence Classifier is used to identify whether the input document indicates an evidence of an attack or a sensitive information leakage. In Evidence classifier several heuristic checkpoints were considered which can be used later to define feature vector when applying machine learning techniques. From this the feature vector can be inferred and apply machine learning techniques to evidence classifier.

Pre filter, Context Filter and Evidence Classifier uses regular expression based matching to extract relevant input documents that contains evidence of attack. Content Classifier classifies each textual input into a set of predefined classes. Each input document is classified into one

or more of the nine defined classes. For each classifier from nine classifiers in Content classifier, a set of heuristics can be defined which later helps to infer the feature vector that is used in machine learning. Here unigrams, bigrams and trigrams were considered to minimize false positives resulting from the classifier.

After the class assignment done in the content classifier, LeakHawk core performs a set of rule-based checks to identify the sensitive content with respect to the each class. For instance, the Content Classifier labels a particular input document as a Credit Card Information Dump based on the content and metadata of the document. The system labels the sensitivity of a document as CRITICAL, HIGH, or LOW according to semantics and magnitudinal facts found in the post.

In the performance analysis done on each component Pastebin sensor was find to work 100% well. Author has submitted 40 posts to Pastebin within a period of 1 minute and verify whether LeakHawk can fetch all the posts and the result was LeakHawk downloaded all the 40 posts altogether 58 (18 usual posts by others) posts pasted within a one-minute cycle. This was done 10 times in 8 days in 2 weeks' time period to get the results and no false negatives found. The limitation in Pre filter is it adds a large number of false negatives to the system. So the utility of this filter was made optional in LeakHawk 1.0.

The performance of the Context Filter exclusively depends on the values of the information template. A corpus of 2300 data with 220 positive data samples and 2080 negative data samples were used to evaluate the accuracy of Context Filter. For instance in LeakHawk Sri Lankan domain was used in Context Filter the words like "Sri Lanka", "Lanka" contributed in a larger portion for the accuracy of the Context Filter. The word "LK" has led to a higher number of false positives. But without "LK" it led to a very high number of false positives increasing the false positive rate by 50%. Pattern matching mechanisms gives the same result by adding more and more false positives although it gave certain results.

The extracted content related to Sri Lankan domain are fed into the Evidence Classifier and Content Classifier for further text classification process. 940 positive samples were fed into Evidence Classifier and is fed with 10 different samples of test data with the number of entries per seed ranging from 100 to 1,000. 1193 positive samples were fed into Content Classifier and is fed with 20 different sample test sets with the number of entries per seed ranging from 30 to 850.

## 2.5 Real-Time Character Based Stream Handling

In context of the information, real-time processing means transforming the latest available information, handling the numerous data as it is generated. It can also take as when talking about real-time processing, it means processing the data with very low latency.

### 2.5.1 Stream Processing - Apache Storm

Stream processing [21] enables us to analyze the stream to extract mathematical or statistical information analytics on the runtime within the stream. Stream processing solutions are designed to handle Big Data in real time with a highly scalable, available, and fault tolerant architecture.

Apache Storm is a real-time fault-tolerant computation system for processing large volumes of high-velocity data. Storm is currently being used to run various critical computations in various places in real-time and it is a free and open source distributed real-time computation system.

Characteristics of Storm are,

- Fast – Benchmarked as processing one million 100 byte messages per second per node
- Scalable – With parallel calculations that run across a cluster of machines. And it is designed to add or remove nodes from the cluster without disturbing existing data flows through storm topology.
- Fault-tolerant (Resilient) – When workers die, Storm will automatically restart them. If a node dies, the worker will be restarted on another node. Since storm is normally deployed in a large cluster, the storm topology can continue processing existing topology with minimum performance impact when one machine is failed due to any reason.
- Reliable – Storm guarantees that each unit of data (tuple) will be processed at least once or exactly once. Messages are only replayed when there are failures
- Easy to operate – Standard configurations are suitable for production on day one. Once deployed, Storm is easy to operate.

### 2.5.2 Overview of Storm

Storm runs on a distributed cluster. Clients submit topologies to a master node, which is called the Nimbus. Nimbus is responsible for distributing and coordinating the execution of the topology. The actual work is done on worker nodes. Each worker node runs one or more worker processes. At any point in time a single machine may have more than one worker processes, but each worker process is mapped to a single topology. Note more than one worker process on the same machine may be executing different part of the same topology.

Nimbus node is the master node of the apache storm and is the touchpoint between the user and the storm. To submit a job to the Storm cluster, the user describes the topology as a Thrift object and sends that object to Nimbus. Thereafter nimbus coordinates all the computations of submitted job in the cluster by distributing codes and launching workers across the cluster. And also nimbus monitors computation and reallocates workers as needed.

Initially user submitted code is stored in the local disk of the nimbus. And then nimbus uses a combination of the local disk(s) and Zookeeper to store state about the topology. All coordination between Nimbus and the Supervisors is done using Zookeeper. Since Nimbus and the Supervisor daemons are fail-fast and stateless, all their state is kept in Zookeeper. Each worker node runs a Supervisor that communicates with Nimbus. Supervisor nodes communicates with Nimbus through Zookeeper, starts and stops workers according to signals from Nimbus. It also monitors the health of the workers and respawns them if necessary.

Each worker process runs a JVM, in which it runs one or more executors. Executors are made of one or more tasks. The actual work for a bolt or a spout is done in the task. Tasks provide intra-bolt/intra-spout parallelism, and the executors provide intra-topology parallelism.

Basic storm data processing architecture consists of tuples, streams, spouts and bolts. Logical collection of all of these is called a topology and it is a directed graph. Vertices in this graph represents the bolts/spouts and edges represents the flow of data. Each components is described below:

- Tuples – An ordered list of elements. Tuples can contain any kind of data.
- Streams – An unbounded sequence of tuples is processed and created in parallel.
- Spouts – Sources of streams in a computation. Generally spouts will read tuples from an external source and emit them into the topology (e.g., from a Kafka consumer).

- Bolts – Process input streams and produce output streams. They can run functions, filter and aggregate, join data or talk to databases. Both spout and bolt can emit more than one stream.
- Topologies – The logic for a real-time application is packaged into a Storm topology. Logic needs to be represented using network of spouts and bolts.

The Storm system relies upon the notion of stream grouping to specify how tuples are sent between processing components. In other words, it defines how that stream should be partitioned among the bolt's tasks. In particular, Storm supports different types of stream groupings such as:

- Shuffle grouping – Tuples are randomly distributed across the bolt's tasks in a way such that each bolt is guaranteed to get an equal number of tuples.
- Fields grouping – The stream is partitioned by the fields specified in the grouping (hashes on a subset of the tuple attributes/fields).
- All grouping – Replicates the entire stream to all the consumer tasks.
- Global grouping – Sends the entire stream to a single bolt.
- Local grouping – Sends tuples to the consumer bolts in the same executor. If the target bolt has one or more tasks in the same worker process, tuples will be shuffled to just those in-process tasks. Otherwise, this acts like a normal shuffle grouping.

Apart from Apache Storm, there are other open source big data analysis tools. Some of them are Apache HBase, Hadoop, Apache Spark and Yahoo S4. Iqbal and Soomro compared among those four tools and Apache Storm's perspective for big data analysis as follows:

**Apache HBase** – Apache HBase [22] is a Java based, open-source software, which enables to store Big Data. It is highly non-relational in nature and provides Google's Bigtable like functionality to store sparse data. HBase is widely used when random and real-time access to Big Data is required and is operates on the top of HDFS.

**Hadoop** – Hadoop [23] is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment. The key features of Apache Hadoop are its reliability, scalability and its processing model. It allows processing the large sets of data across clusters of machines using distributed programming paradigm. It operates the information in small batches and uses

MapReduce framework to process the data and is called batch processing software. MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. MapReduce serves two essential functions: It parcels out work to various nodes within the cluster or map, and it organizes and reduces the results from each node into a cohesive answer to a query.

**Apache Spark** – Apache Spark [24] project is open source based for processing fast and large-scale data, which relies on cluster computing system. Like Apache Hadoop it is also designed to operate on batches, but the batch window size is very small. Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.

**Yahoo S4** – Yahoo S4 [25] empowers developer to easily design applications, which can process real-time streams of data in a distributed cluster system with scalability and fault-tolerant. It is inspired by MapReduce model and process the data in distributed fashion.

Table 3 compares Storm with other big data analysis tools.

*Table 3: Comparison of Storm with other big data analysis tools*

| **Other Tool** | **Developer** | **Type** | **Differences** |
|---|---|---|---|
| HBase | Apache | Batch Processing | Storm provides real time data processing, while HBase (over HDFS) does not process rather offers low-latency reads of processed data for querying later. |
| Hadoop | Apache | Batch Processing | The main difference is that Storm can do real-time processing of streams of Tuple's (incoming data) while Hadoop do batch processing with MapReduce jobs. |
| Spark | UC Berkeley AMPLab | Batch Processing | A batch processing framework that is capable of doing micro-batching also called Spark Streaming, while Apache Storm is real-time stream processing frameworks that also perform micro-batching also called Storm-Trident. So architecturally they are very different, but have some similarity on the functional side. With micro-batching, one can achieve higher throughput at the cost of increased latency. With Spark, this is unavoidable and with Storm, one can use the core API (spouts and bolts) to do one-at-a- time processing to avoid the inherent latency overhead imposed by micro-batching.<br>Many enterprises use Storm as a mature tool while Spark Streaming is still new. |
| S4 | Yahoo | Streaming Processing | The main difference is that, storm gives guaranteed processing with high performance and thread programming support. |

There are five key attributes of Apache Storm which make it the first choice for real-time unbounded data processing. Those attributes are Easy to use, Fast, Fault-tolerance, Reliability and Scalability. Those attributes were described earlier in this section. Following criteria can be used to decide whether to use Apache storm or not for our application:

- Fault tolerance: High fault tolerance
- Latency: Sub Seconds
- Processing Model: Real-time stream processing model
- Programming language dependency: Any programming language
- Reliable: Each tuple of data should be processed at least once.
- Scalability: High scalability.

Later they have done three different experiments using twitter big data taking from twitter streaming API [12]. The experiments will execute three different scenarios with live data and will collect statics. The three experiments are, top ten words collected during a particular period of time, top ten languages collected during a particular period of time and number of times a particular "word" being used in twits, twitted in a particular period of time.

All the three experiments performed successfully. So it proves that Apache Storm can process real-time data with very low latency. Modelling the programming logic using the bolts and spouts is also easy. Because required parallelism can be configured for each bolt, we can easily configure it per each bolt according to workload in each bolt/spout.

When deciding which tool to be used for our job, choosing one over another should be done carefully. We have to consider about processing guarantees, programming models, and APIs. Also, results of a research done by Codova shows that Storm was around 40% faster than Spark, processing tuples of small size. However, as the tuple's size increased, Spark had better performance maintaining the processing times. Generally tweets are also works as small size tuples. Therefore, it can be concluded that Apache Storm to process twitter real time data would be the best choice for our purpose.

## 2.6 Text Analysis for Sensitive Document Classification

### 2.6.1 Text classification of social media and crowdsourced data

Sparks et al [26] identified and located facility types like restaurants, airports and stadiums and identify methods to determine in which time periods they become popular among people using social media like Twitter and Facebook. Locating these facility types helps in places where land use data is needed. Population distribution, biodiversity monitoring, urban dynamics and energy consumption are some tasks where land use data is significant. Identifying when these facility types mentioned above become popular helps determining in which time of the day, in which days of the week, in which festive occasions etc. the facility types are popular and helps in population dynamics. These data are beneficial for urban planners and general geographic research. In current context social media is a widely using source of information around the world. The posts have the ability to tag spatial and temporal data along with it by making social media a near real time source to get information for land use classification. They have showed travel records and trip surveys, call detail records (CDR) are not near real time and available sources of information to get land use data. The benefit of social media other than traditional methods is, it not only allow to tag spatial and temporal data but to describe why they are there and what they are doing as textual descriptions. This research has basically focused on the textual description of the post not the check-in data (tagged data) associated with social media.

Authors mainly used Twitter for their study and used a data set of 1 year consisting of English tweets. Preprocessing of textual input is done by removing emoticons, non-ASCII characters, hashtags and URLs from the content.

[A]. Now that's what I call a #beer. #FamilyDayOut2 @ Cargo Restaurant Bar

[B]. Now that's what I call a beer. FamilyDayOut2 at Cargo Restaurant Bar

Here [A] shows the text before preprocessing and [B] shows the preprocessed text as mentioned above. After preprocessing the data set, they have created the training data set using NLP tools like Stanford university's CoreNLP. The URL and hashtag removed text [A] is sent to perform NLP processing. The result is a set of words as in [B] which is easily can be used to apply TF-IDF.

[A] Just waiting on my food (at Tracks End Restaurant in Chicago, IL)

[B] (Just, wait, food, at, tracks, end, restaurant, chicago, il)

They have created TF-IDF vectors using preprocessed text in [B], which are to be feed into ML classifiers. Naive Bayes and Support Vector Machine classification algorithms has been used for the classifying whether a person is at the location of interest or not when the tweet was sent. They had trained the classifiers using 10 fold cross validation (90% of data for training and 10% for testing) and had done it 100 times. They have taken accuracy and precision as their evaluation metrics. Accuracy was defined as depicted below.

The results were obtained for all three facility types mentioned above and airports have shown unique results for both NB and SVM classifiers in accuracy and precision whereas restaurants and stadiums had a bias for SVM in both accuracy and precision. For restaurants it has shown a maximum accuracy of 0.932 and a precision of 0.987.

In our research TF-IDF weighting is used to represent the text as vectors that are used to feed into ML classifiers since it has become successful in this research.

### 2.6.2 Text Categorization with Support Vector Machines

Joachim et al [27] considered the results obtained in to show that SVMs are appropriate for text categorization process. According to the study the properties of SVM which make it appropriate for text categorization are,

- high dimensional input space - SVM handles overfitting protection, so no consideration of number of features used
- few irrelevant features in text categorization
- sparse document vectors - For each document vector it contains very few features where feature values are not zero
- most text categorization problems are linearly separable - SVM finds linear separators in its classification

They have used two training datasets for training the classifiers. To get an unbiased result they have used different number of selected best feature sets (500 best, 1000 best, etc., all features) to train all the classifiers they have considered. Here SVM has used to learn a polynomial classifier and a Radial Basic Function (RBF). The results of SVM classifier was compared with four other classifiers namely Naive Bayes, Rocchio, C4.5 and K-NN. From the results it can be concluded that SVM performs better independent of parameters where in

polynomial classifier it has shown an average of 86.0 and in RBF classifier it has shown an average of 86.4.

By considering the results of the comparison of SVM classifier with other classifiers on text categorization we concluded that SVM classifier can be used in classification process of our research since mainly we are also doing text based classification process throughout all classifiers.

### 2.6.3 Twitter trending topic classification

Lee et al [28] mentioned that the trending topics generated by twitter are hard to understand and identify, so there's a need to have a more meaningful trending topic classification of tweets. The researchers have identified 18 general categories such as sports, politics, technology etc. that can be used as trending topics. Mainly they had taken two approaches for the topic classification as Bag-of-words concept for text classification and network based classification. A variation of NB classifier which Naive Bayes Multi-nominal classifier which consider word frequency is used for text classification. The procedure that they have carried out during preprocessing of text in text classification is removed tokens that contain hyperlinks, tokenized the document which removes delimited characters and stop words and then converted the tokens into tf-idf vectors. For each category top 500 and 1000 frequent words were used. As the next step network based data modelling was done in order to find similar topics for a given category. That was done using Page-Rank Algorithm and Twitter social network information such as tweet time, number of tweets made on a topic and friend-follower relationship. This model assumes that if the users sending tweets on two topics have a similarity, then the two topics should have a similarity.

Text based classification was done using NB, SVM-L, NBM classifiers and results have shown that NB showed lower accuracy than NBM and SVM-L also had a slightly lower accuracy than NBM. In network based classification five classifiers were trained using manually labeled similar topic data set. The classifiers were C5.0, K-NN, SVM, Logistic Regression and ZeroR where C5.0 classifier has shown the maximum accuracy of 70.96%.

By considering the approach taken in this research it has shown that NBM has the highest accuracy in text classification which has a slightly higher accuracy than SVM. By considering the above two researches we concluded that for our research we would use SVM, NBM mainly for comparison since they had shown the best results in text classification.

### 2.6.4 Ontology-based Supervised Text Classification

Risch, Petit, and Rousseaux [29] proposed a text classification method linking the three domains; natural language processing, machine learning and big data. A method of supervised classification of documents based on a domain ontology developed in a real time and big data environment is presented here.

Automatic text classification or categorization (ATC) is a multidisciplinary research field composed of machine learning, natural language processing (NLP), Big Data, real time analysis and so on. There are three approaches of ATC: supervised, unsupervised and semi-supervised classification of documents. They have used supervised approach since their goal was to create a statistical classification model from a corpus of previously annotated documents. A supervised classification method contains two main parts: learning the model with the labeled data and predicting labels on new data. In this method they have created the learning model by giving each concept of the ontology a probability of class belonging.

The documents are received in real time and then each document is pre-processed to extract a list of lexical units. Relying on a pre-built ontology, they propose a classification method based on the similarities between the ontology, the analyzed document and the associated probabilities.

Authors have chosen the Tika API maintained by the Apache Foundation for extracting raw text. This API allows the extraction of raw text from over 1,000 file types. To detect the language of a text, they have chosen the LangDetect API that can recognize a multitude of languages including European, Japanese, Russian, etc. and Yandex Translator to translate the text to English. Once the text is translated into English or analyzed as written in English, pre-processing is done using mainly Stanford API. They have used standard natural language processing operations like changing the text data to lowercase, sentences detection, tokenization, part of speech tagging, lemmatization and parsing. At the end of the feature extraction task, each text document is described by a set of noun phrases from which the classification model is built.

In training the model class probabilities are assigned for each concept of the ontology. They have used a technique similar to overlapping techniques as a concept can belong to several classes. The probabilities are determined by computing the frequency of each concept in the documents.

Authors used two methods in the prediction phase as direct prediction and extended prediction. In order to avoid having documents classified in irrelevant classes, they have used a threshold value between 0 and 1. In the extended prediction they find new features that were not mentioned by using noun phrases and selected neighbor concepts.

In the real-time analysis phase to address the speed and volume issues they have used two APIs: Apache Kafka and Spark Streaming. They have chosen Apache Kafka because it is a distributed message broker which can handle a big amount of messages per second. Its objective is to manage the flow of messages between producers and consumers. In this case a producer is a source of documents. It can be an RSS feed, a social media feed, etc. A consumer is an application that will receive and process the message (document). In order to handle the messages in real time they have used Spark Streaming which possesses a connector with Kafka.

They have also implemented graph analysis using Apache Spark GraphX and have used Apache Cassandra API for database management because it is easy to use, especially with Spark because of its connector.

In our project to handle the feed we use Kafka and for text classification we are using supervised classification method. In this research they have translated the content in other languages to English. But we decided to keep that for future enhancements to be handled after completing the main tasks. We hope to use Stanford API for pre-processing; the changing the text data to lowercase, sentences detection, tokenization, part of speech tagging, lemmatization and parsing.

### 2.6.5 Early Detection of Spam Mobile Apps

Seneviratne et al [30] have done a research about automatically identifying the Spam mobile applications. Authors proposed an adaptive boost classifier for early identification of spam apps at the time of app submission. This app classifier utilizes only those features that can be derived from an app's metadata available during the publication approval process. It does not require any human intervention such as manual inspection of the Meta data or manual app testing. They have validated this app classifier, by applying it to a large dataset of apps collected between December 2013 and May 2014, by crawling and identifying apps that were removed from Google Play Store. This research shows that it is possible to automate the process of detecting spam apps solely based on apps' metadata available at the time of

publication and achieve both high precision and recall. Their classifier achieves an accuracy over 95% with precision varying between 85%-95% and recall varying between 38%-98%.

In their classifier they used heuristics checkpoints to identify whether app is spam or not spam. They mainly focus on nine heuristics checkpoints and app will be classified as spam or not according to the results from those checkpoints. First checkpoint is “Does the app description describe the app function clearly and concisely?” this checkpoint is measured by using “bi-grams” and “tri-grams” in the description of the application. Second one is “Does the app description contain too much details, incoherent text, or unrelated text?” this checkpoint is measured using another sixteen feature list and a decision tree classifier with maximum depth 10. Third one is “Does the app description contain a noticeable repetition of words or keywords?” this checkpoint is measured by considering the number of unique words in the description relative to the all the word count in the description. Fourth one is “Does the app description contain unrelated keywords or references?” this checkpoint is measured using if-idf weights. Fifth one is “Does the developer have multiple apps with approximately the same description?” in this checkpoint, they have checked three points. The total number of other apps the developer has, The total number of apps with an English language description which can be used to measure descriptions similarity and the number of other apps from the same developer having a description cosine similarity(s), of over 60%, 70%, 80% and 90%. Next checkpoint is “Does the app identifier (appid) make sense and have some relevance to the functionality of the application or does it appear to be auto generated?” in this checkpoint they are considering about the app id’s and considered 13 points to check the appid is suspicious or not. Also other than these checkpoints they have considered metadata like the category of the application.

# 3 Design

This chapter describes the design of the project which include activity diagram, class diagrams, topology and a brief description of each component of the system. High-level architecture of LeakHawk 2.0 is shown in Figure 3-1 compared to LeakHawk 1.0 architecture illustrated in Figure 2-14. LeakHawk 2.0 uses a modularized architecture, as it is customizable and scalable where new features can be added to the system.

## 3.1 High-Level Design

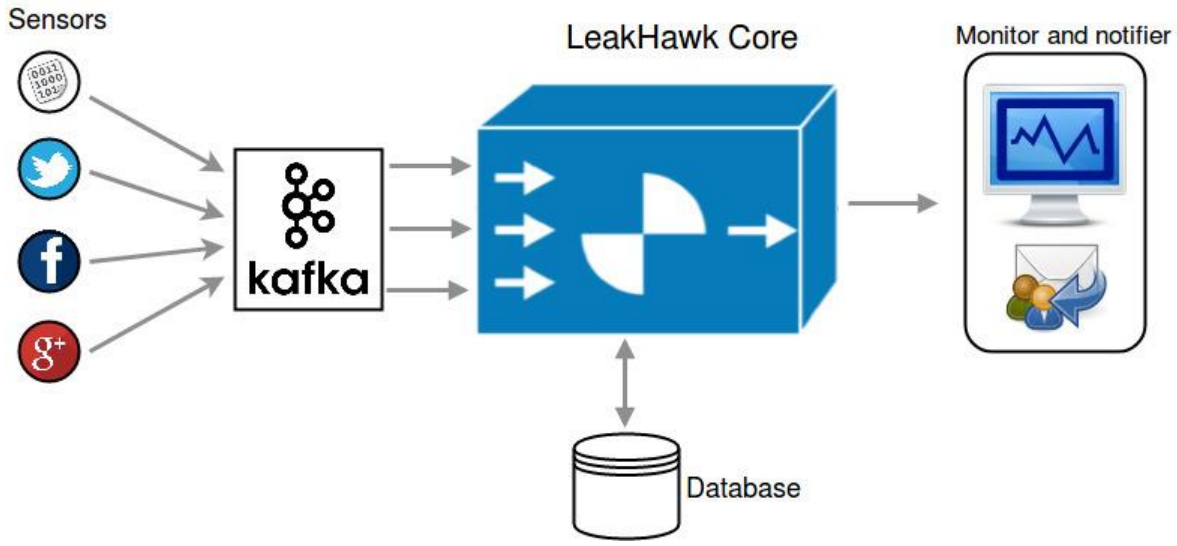


*Figure 3-1: High level architecture of LeakHawk 2.0*

Figure 3.1 High-Level architecture of the LeakHawk.

The system has the ability to incorporate data feeds from several data sources like Twitter, Facebook, and Google+. Those message producers use Apache Kafka [31] as a message broker to hold the incoming textual input in the respective queue and provide to the message consumers in LeakHawk core. Because posts from all data sources come as plain text, the LeakHawk core is a generalized module that can be used for text classification process independent of the data origin. Sub-modules inside the LeakHawk engine will check whether the incoming text belongs to the configured information domain. If so, then the text is classified to find whether it has an evidence of a data breach exposure. If the post contains an evidence and contains URLs, content pointed by those URLs are also pulled for further classification. Regardless of whether the post contains an evidence or not, the post is categorized into one of predefined classes and LeakHawk predicts the sensitivity of the input based on results of class categorization. If there is evidence of a data breach exposure related to the entity who wants to check whether their data is exposed, the system notifies through the dashboard. In summary, LeakHawk engine monitors online channels like pastebin.com and Twitter and classifies the input text and predicts a sensitivity label of that text, and notifies data owners via a notification module.

## 3.2 Classification of Posts

Figure 3-2 outlines the flow of a post classification process, where it follows the following steps.

1. When a new paste or a tweet is published, it is retrieved and aggregated into the LeakHawk Core for further processing.
2. Check whether the post is empty or belongs to any of the predefined categories such as gaming chats, pornographic content etc. If the post identified as irrelevant it is discarded, else it is sent to the next level.
3. Next, check whether the post belongs to the defined domain (e.g., financial institutes and country). If the post belongs to the given domain, the post is passed to the next level, else the post is dropped.
4. Then check whether the post has any evidence of data breach exposure, and if so the post is sent for processing URLs. Else the post is sent to analyses the content.
5. If URL(s) is found in the post, download the content from the given URL(s) and send them to analyses the content for further processing.
6. The content in the post is checked against all predefined classes such as credit card, email only, private keys etc. If any class gives the output as true, post is considered as belonging to that class. One or more classes may give the output as true. This level defines criteria in each class to rank the post according to the sensitivity level.
7. Then predict the sensitivity label of the post using the statistics from the content classes.
8. Metadata of all the fetched post and textual content of each document are stored in the database.
9. Finally, the respective data owner will be notified about the data breach exposure with the predicted sensitivity label.

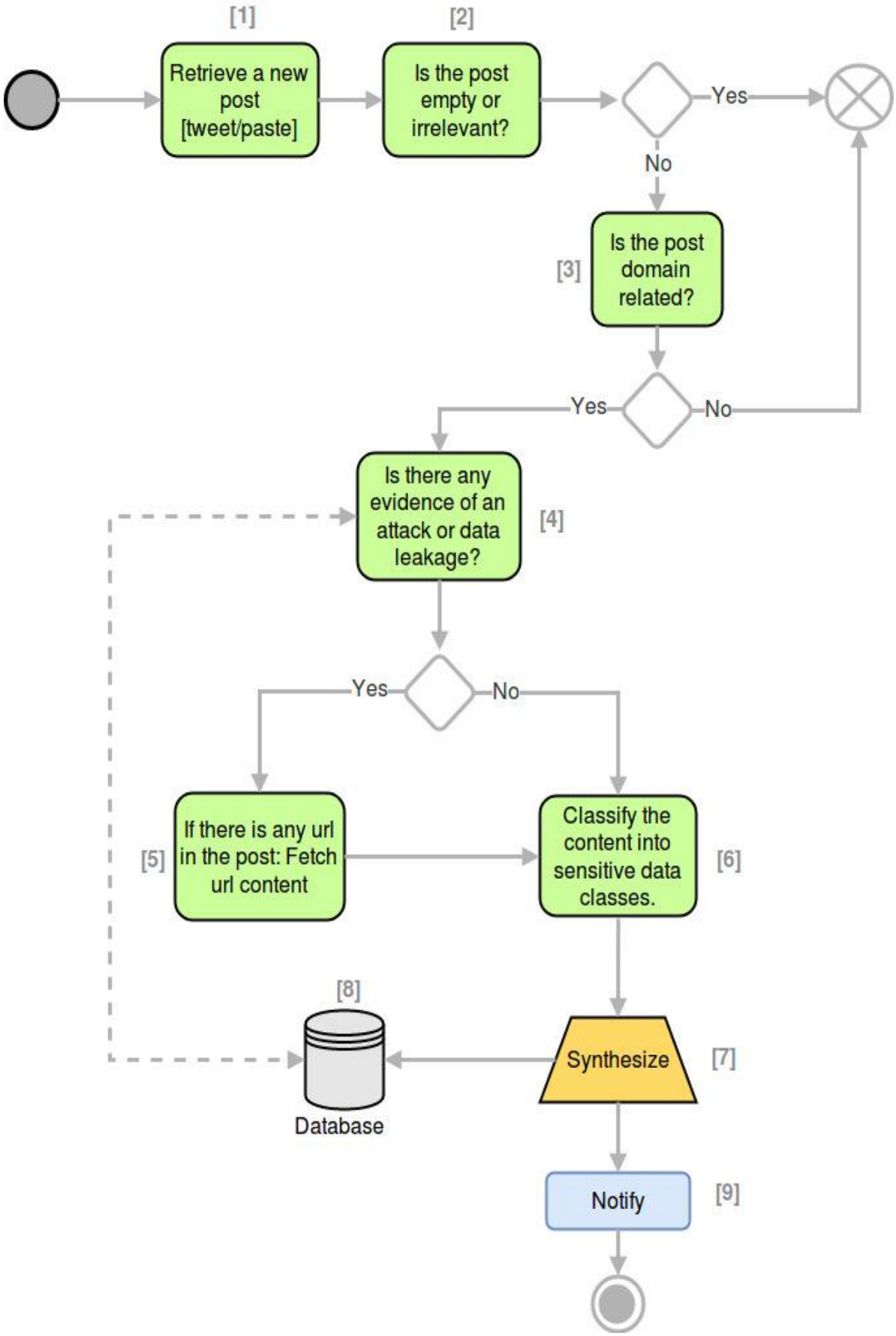


*Figure 3-2: Process used to classify a post*

## 3.3 Component-Level Architecture

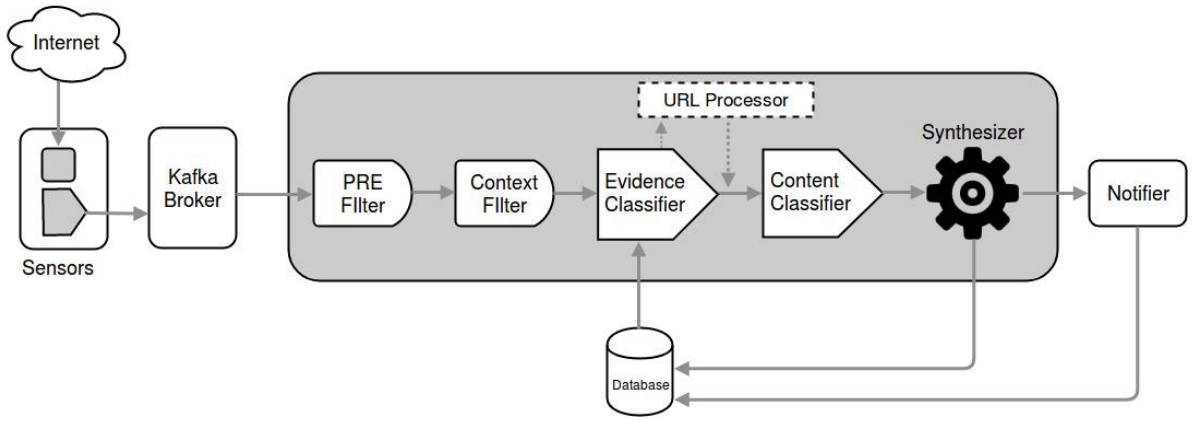


*Figure 3-3: Component architecture of LeakHawk 2.0*

LeakHawk 2.0 consists of several components which are presented next.

### 3.3.1 Sensors

LeakHawk 2.0 may contain any number of sensors that are used to pull data/posts from data breach exposure sites. To demonstrate the idea, we initially developed two sensors, namely Pastebin and Twitter sensors, which are used to fetch posts from Pastebin and Twitter. Task of the sensors is to retrieve the relevant feeds and forward them to the Kafka broker using Kafka producer as shown in Figure 3-3.

Bandwidth is a major requirement that has to be considered in each sensor. This section gives a brief explanation about the bandwidth requirement of each sensor according to calculated results.

#### 3.3.1.1 Pastebin Sensor

| | |
|---|---|
| Average Pastebin feed | = 24 posts per min |
| Average size of a post | = 12 KB |
| Bandwidth requirement | = 12×24 KB per min |
| | = 288 KB per min |
| Average bandwidth requirement | = 288/60 = 5 KBps |

#### 3.3.1.2 Twitter Sensor

| | |
|---|---|
| Average twitter feed | = 6,000 tweets per second |
| Average size of a tweet | = 6 KB |
| Bandwidth requirement | = 6×6000 KB per second |
| | = 36000 KB/s = 35.5 MBps |

While the bandwidth for Pastebin is not high, it can be high for Twitter. This can be reduced by relying on Twitter feeds filters that send only the requested type of contents based on users, hash tags, or content.

### 3.3.2 Pre Filter

The role of the Pre filter is to filter out irrelevant posts so that further processing of such posts is prevented which will save both time and processing power. The data feeds taken from the sensors are sent to respective Pre filters. There are two Pre filters for Pastebin and Twitter. The identified irrelevant posts for Pastebin and Twitter are trial and empty pastes, programming codes, game chats, sport commentaries, pornographic content and seasonal greetings content whereas trial and empty pastes and programming codes are excluded which are not frequent categories in Twitter. Since we need to infer a function from the training dataset of Pastebin posts labeled as relevant and irrelevant, pre filtering of Pastebin posts is identified as a supervised Machine-Learning problem, which can be used to predict the unseen posts. With a proper training corpus this is an achievable task. In Twitter due to the character limitation this cannot be identified as a text classification problem.

### 3.3.3 Context Filter

Context filter is used to filter the posts that are related to user defined information domains. For instance the information domain can be the banks in Sri Lanka. Thus, the Context filter screens out the non-related information and extracts only the posts related to the context, system is focused on, which will minimize the processing of non-related posts. Regular expressions and keyword matching is identified as a suitable technique to handle this task. Keyword list is maximized using WordNet API which provides connected words that will expand the scope.

Only one Context Filter is used in LeakHawk 2.0 as the information domain is common for any data feed. After the Context filter the post is sent to the respective Evidence classifier.

### 3.3.4 Evidence Classifier

Evidence classifier is used to detect whether an incoming post has an evidence of possible data breach exposure. There are two separate Evidence classifiers one for Pastebin and another for Twitter because they are two different supervised Machine-Learning problems. They are supervised Machine-Learning problems because we need to infer a function for the learning model from the training data labeled as having an evidence of data breach exposure or not. Two different Machine-Learning models needed to be created for both classifiers with two different datasets which can be used to classify any unseen post. Then the post is sent to URL processor or Content classifier depending on the presence of an evidence.

### 3.3.5 URL Processor

If the post has an evidence of a hacking attack or a data breach, it is sent to the URL processor which checks for URL(s), pull the contents from the URL(s), and then forwards the URL content to Content classifier. This component is common for any type of data feed.

### 3.3.6 Content Classifier

Content classifier is used to detect the content against predefined set of classes to find to which category the post belongs to. For instance the categories are credit card, database dumps, email conversations etc. and the post is categorized into one or more categories based on the content. There are two Content classifiers targeting Pastebin and Twitter. Each predefined class is considered as a classifier in Pastebin Content classifier and identified as a supervised Machine-Learning problem which needs to infer a function for learning model with labeled training data. Binary classification is performed in each classifier. The post is categorized into a class when the binary classification gives the result as true. However, as tweets does not contain much content in the tweet itself, this cannot be considered as a text classification problem.

### 3.3.7 Synthesizer

All the posts that have contents related to a possible data leakage or an evidence of hacking attack have a certain level of sensitivity. Synthesizer is used to rank the sensitivity of such posts according to the semantics and magnitude of numbers in the content. For instance, semantics in the case whether the post has information on a possible data breach exposure and magnitude in the case whether the post has information on one credit card related information or several hundreds of credit card related information have different levels of importance as

the magnitude of the breach or its impact could vary. Synthesizer predicts a label for the sensitivity of the post as CRITICAL, HIGH, or LOW as per the content of the post, its semantics and magnitude, which is illustrated in Table 7. This label is used to notify the users in case of a possible data leakage or an evidence of hacking attack.

To illustrate the role played by each component let us consider an example of a post passing through each component. Suppose the post in Figure 3-4 is retrieved from Pastebin sensor. Then the flow of the post through each LeakHawk component is as follows:

```
Bank of Ceylon hacked!!!

LEAKED CREDIT CARD INFO! #Anon #BentThimble #Credit #MasterCardLeak&Random
5524301014857997, CVC2, 279, 11/2019
5238516770450516, CVC2, 880, 08/2020
5329446731722615, CVC2, 604, 08/2018
5416186536152204, CVC2, 277, 02/2019
5286028741369362, CVC2, 432, 09/2020
5138075127336838, CVC2, 644, 02/2021
5598540919372708, CVC2, 538, 09/2020
5485812259197896, CVC2, 112, 12/2020
5328138410217311, CVC2, 116, 02/2021
5149698785681471, CVC2, 957, 01/2018
5514935152567728, CVC2, 100, 03/2021
5210506298661423, CVC2, 914, 12/2020
5291744202034252, CVC2, 564, 08/2021
5242940611138706, CVC2, 153, 03/2020
5507319343606068, CVC2, 533, 02/2021
5498000149843474, CVC2, 889, 03/2019
5464218814977464, CVC2, 559, 02/2021
5556295148512067, CVC2, 635, 10/2019
5188102080541995, CVC2, 194, 12/2019
5140699202363072, CVC2, 280, 08/2018
5197987529522971, CVC2, 734, 09/2019
5538740014181136, CVC2, 917, 06/2021
```

*Figure 3-4: Example Pastebin post with data breach*

- Pre filter – Filter in
    - The post does not contain any keyword identified as irrelevant (e.g., Code words, Gaming chats, Pornographic words, etc.). Hence, will be considered for further processing.
- Context filter - Filter in
    - The post contains keywords related to Sri Lankan domain, e.g., Bank of Ceylon
- Evidence classifier - Pass
    - The post contains evidence of a data breach as it contains the word “hacked” and include leaked credit card information.

- URL processor - Not processed as no URL(s) are found. Passed to Content classifier.
- Content classifier - Passed
    - The post belongs to Credit Card (CC) class as it contains both keywords and credit card content.
- Synthesizer - The sensitivity level of the post is marked as CRITICAL
    - Post belongs to Credit Card class and the credit card no count is 22 which is detected as CRITICAL as the threshold to be detected as CRITICAL is 20.

## 3.4 LeakHawk Topology

This section elaborates the topology of LeakHawk from the sensors to the Synthesizer.

### 3.4.1 Apache Kafka

Apache Kafka is a real-time message distribution platform which follows publish-subscribe messaging strategy. In publish-subscribe messaging strategy there are producers who pushes the messages to Kafka brokers as per a predefined topic. Kafka brokers queue and publish the messages so that Kafka consumers could pull them by subscribing. Zookeeper is used to save the states of Kafka brokers and share state between brokers. A set of Kafka brokers maintain a set of topics. In case of Kafka consumers, they pull the messages as per the corresponding topic. Kafka is fast, scalable, efficient, persistent and fault tolerant. Due to performance characteristics and ability of parallelizing consumption of messages Kafka is used in LeakHawk to queue messages coming from both Pastebin and Twitter in a Kafka broker in real-time and consume the messages parallely.

### 3.4.2 Apache Storm

Apache Storm is an open source real-time Big Data processing platform. Although Apache Spark and Apache Flink both has features similar to Apache Storm, they mainly support processing of micro batches where Storm uses event processing and has relatively low latency. Therefore, LeakHawk is developed on top of Apache Storm Big Data processing framework. Storm converts data streams from different data feeds into sequence of tuples known as *stream*. Tuples support all data items. There the aforementioned stream is considered as events, not as a batches. Apache Storm uses a master-slave architecture with Apache Zookeeper based coordination, where master is the Nimbus and slaves are supervisors. Zookeeper helps managing states of master and slaves. The basic components introduced in

Storm are spout and bolt which helps transforming streams. Storm typically processes real-time data and inputs coming from external messaging queuing platform like Apache Kafka. A spout is a source of stream for instance which may retrieve data from Kafka broker or directly from Twitter API and emit a stream of tweets as tuples. Bolts are used to process any number of streams and emit new streams with the help of workers. A bolt may comprise several workers which perform several tasks. A topology in Storm is the user defined real-time application logic helps in both design and implementation phases.

According to the flow illustrated in Figure 3-5 the Storm topology distributes the Twitter and Pastebin feeds to the relevant components as shown in Figure 3-5.

There are dedicated components for both Twitter and Pastebin such as Pre filter, Evidence classifier and Content classifier in which the processing method differs with the type of the feed. Some components are commonly used by both Pastebin and Twitter, e.g., Context filter, URL processor, and Synthesizer in which the processing method is similar regardless of the feed type. There are two separate sensors that work as Kafka producers to connect to Pastebin and Twitter and download the new posts and push them to Kafka broker. Two separate Storm spouts for Pastebin and Twitter are used as Kafka consumers to emit the posts from queues as a stream into LeakHawk core. All the other components except sensors and Kafka consumers work as Storm bolts.

Since in Pastebin a URL is sent as the content with API response, the content in the URL need to be fetched unlike in Twitter. So a separate post downloader bolt is added. There are two Pre filters and Content classifiers for both data sources since Pastebin Pre filter and Content classifier uses Machine Learning techniques and Twitter Pre filter and Content classifier uses keyword based and regular expression based processing styles. That is Machine Learning technique cannot be used in Twitter in both cases due to character limitation of the content. Context filter uses the same keyword based rules and regular expressions for both Pastebin and Twitter to check if an incoming post is in the defined domain. In Twitter mostly URLs for the content are posted as a tweet in case of a data breach exposure. There are two Evidence classifiers used for both data sources since both Evidence classifiers use Machine Learning techniques. That is because separate data models needs to be created for Pastebin and Twitter datasets. Since Synthesizer uses the results of class categorization of Content classifier, for both Pastebin and Twitter only one Synthesizer is used. The results of Synthesizer is used to notify the data owners.

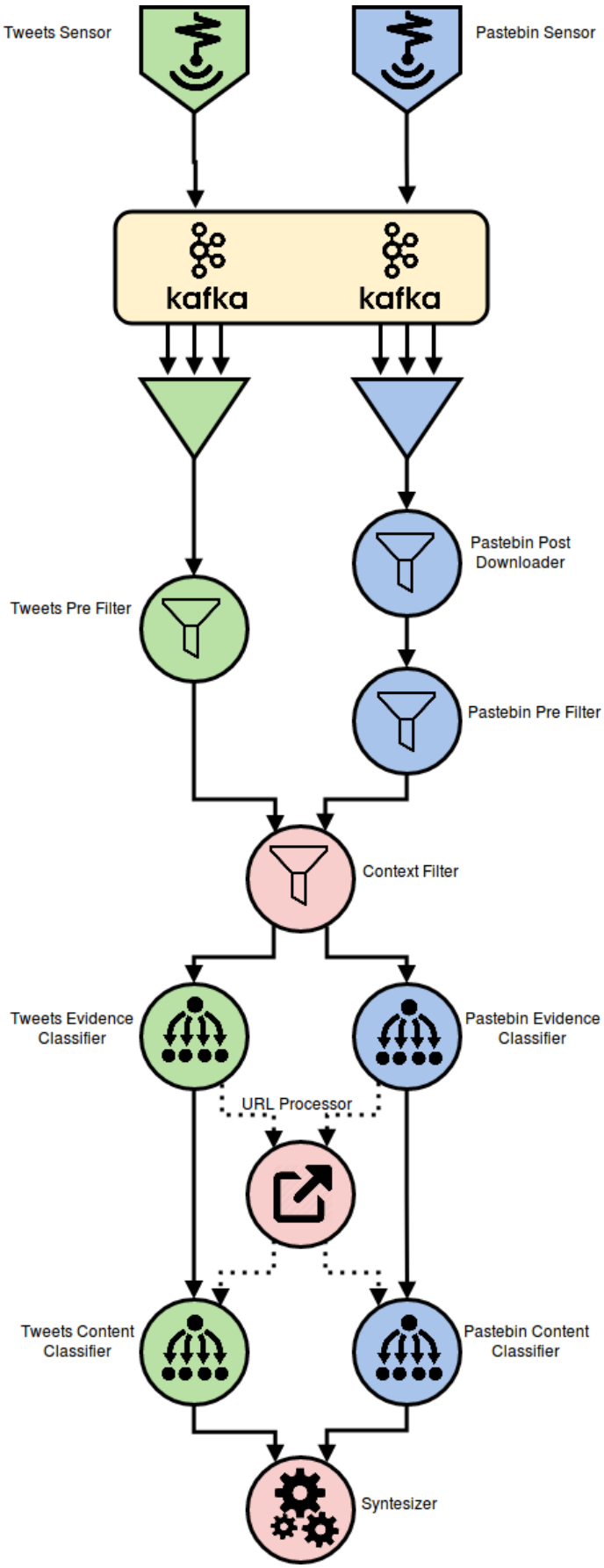


*Figure 3-5: Strom topology for the LeakHawk*

# 4 Implementation

This chapter gives a detailed description about implementation of LeakHawk. Section 4.1 describes the real-time stream processing technologies used. Section 4.2 describes the implementation details of sensors. Sections 4.3 illustrate the implementation of filters, classifiers, and other components in the system. LeakHawk 2.0 is implemented an open source contributed software application and it is available in GitHub [32].

## 4.1 Real-Time Stream Processing

As LeakHawk needs to process posts pulled from various sources in real time, Apache Storm comes in handy as explained in Chapter 3. LeakHawk is developed on top of a Storm topology as illustrated in Figure 3-5. Spouts and bolts of Storm makes it easy to process incoming streams in real time. Spout converts the data feed into a stream which is a set of tuples. Tuples flow in all bolts after the spout which are used for stream processing.

## 4.2 Sensor Implementation

Separate sensors are implemented for Pastebin and Twitter to get the data feeds into the system.

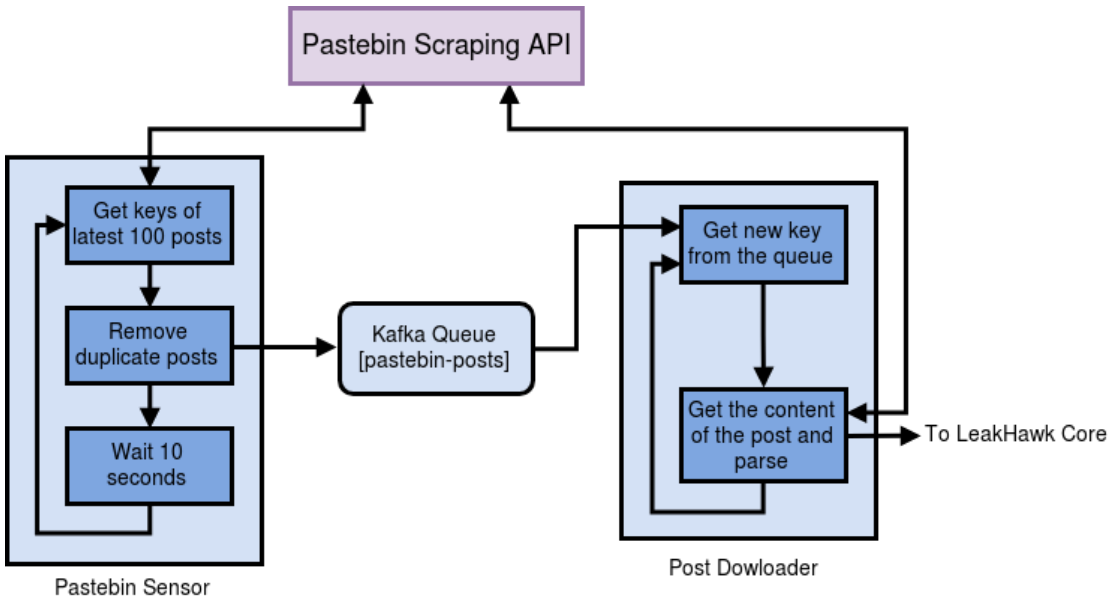


*Figure 4-1: Pastebin sensor implementation*

As illustrated in Figure 4-1, in the Pastebin sensor implementation 100 pastes are taken at a time using the Pastebin scraping API [33]. Then the sensor waits 10 sec before pooling again for new pastes. These numbers are set based on typical rate that posts appear on Pastebin, and can be adjusted accordingly. Depending on the rate that messages are posted, some of the posts may appear in successive 100 pasts pulled from Pastebin. In such cases we remove the duplicates. In Pastebin a URL pointing to the content is sent with the response, so then the post is sent to Post Download bolt to get the content for further processing.

Twitter sensor incorporates Twitter streaming API [34] to get Twitter feed in real time. Twitter 4J [35] is a third-party library used with Twitter API. The combination of Twitter 4J and Twitter API are used to get the Twitter feeds in real time. Because the text in the content is directly sent with the response unlike in Pastebin there's no need of a Post Download bolt, so the text itself can be directly used for processing.

## 4.3 Pre Filter Implementation

Pre filter is able to remove the irrelevant posts at the beginning of the process. With this filter LeakHawk can reduce most irrelevant content without processing them further. Separate Pre filters are implemented in the LeakHawk for different data feeds (Twitter feed, Pastebin feed). To implement a new Pre filter, the Pre filter class should extends from the LeakHawkFilter abstract class with implementing its abstract methods as in Figure 4-2.

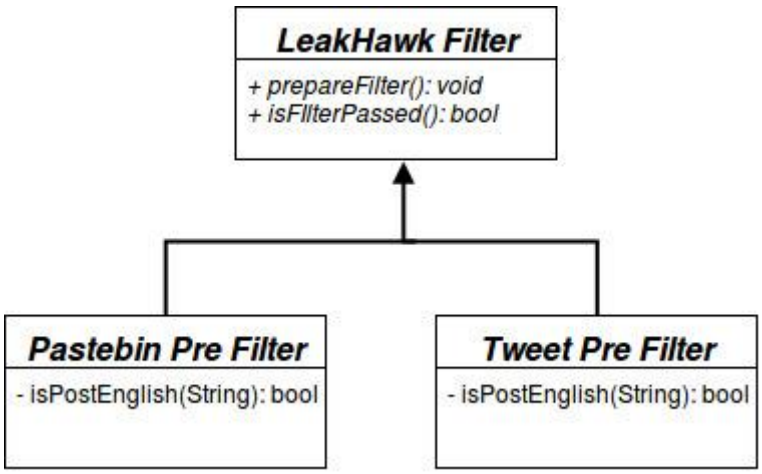


*Figure 4-2: Pre filter class diagram*

These two methods are used in every Pre Filter implementation helping to configure and check the irrelevancy of a poste. *prepareFilter()* method will run only once during the initialization of the bolt. This method can be used to initialize things that would be used in the

*isFilterPasssed()* method. *isFilterPassed()* method is used to identify the irrelevant posts. This method should return a Boolean value according to the irrelevancy of the post. If the post is irrelevant, the return value should be "false" and if the post is relevant the return value should be "true".

### 4.3.1 Pastebin Pre Filter

In the Pastebin Pre filter, text is preprocessed and binary classification method [36] is used to filter out non-related pastes from the LeakHawk core. During this stage Preprocessor filters out non English pastes and remove stopwords related to English language as illustrated in Figure 4-2. Stopword removal is carried out using the WEKA API [37]. Pastebin Pre Filter uses Apache Tika [38] for language detection. Because LeakHawk only considers posts in English, non-English posts need to be filtered out from the pre filter. Tika is useful in this case as metadata on language of the paste is not provided by Pastebin. Tika identifies the language of the incoming paste and helps to filter out that paste from the system.

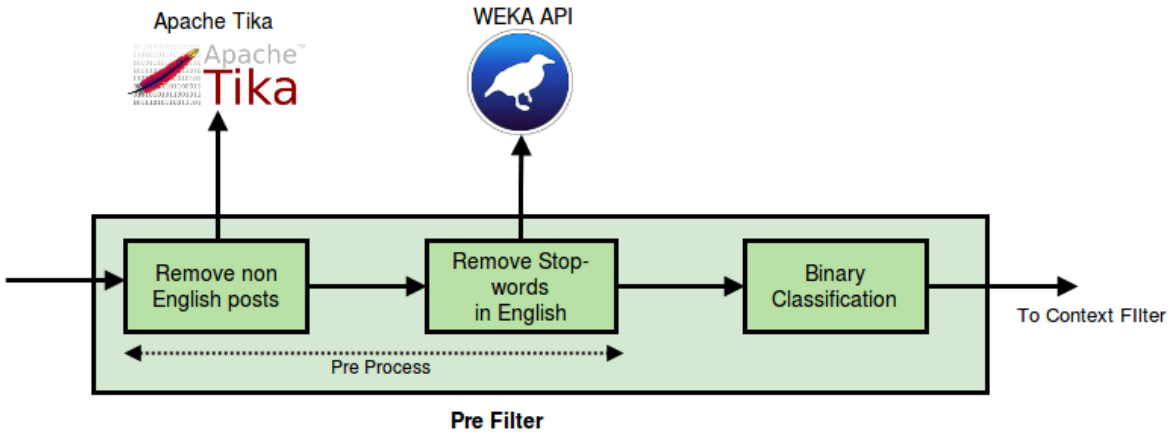


*Figure 4-3: Pre filter process*

Following categories were identified as giving the highest contribution to most number of irrelevant posts coming for the Pastebin, which needs to be filtered out from the system:

- Trial and empty pastes - Trial pasts are mostly used to check the whether Pastebin is working and empty pastes may accidently posted with empty content.
- Programming codes - Most of the developers use Pastebin to share code snippets.
- Game chats - Gaming community uses Pastebin to share secrets related to games and URL(s) to find gaming software.

- Sport commentaries - Sports related comments and greetings are mostly shared during matches.
- Pornographic content - Links to pornographic videos and used to share pornographic content.
- Seasonal greetings content - Mostly used to share seasonal greetings with friends in seasons like Christmas, Eid.

Pastebin Pre filter classifies these kind of irrelevant data using Weka classification process. *isFilterPassed()* method from the super class is overridden here and *isPassedPreFilter()* method is invoked inside that method. *isPassedPreFilter()* method is used to check whether the incoming post passes pre filter and move forward.

### 4.3.2 Twitter Pre Filter

In the Twitter Pre filter implementation, irrelevant Tweets are filtered out from the system to reduce the processing overhead. Here attributes given by Twitter streaming API has been used to make the implementation simpler. Twitter API's *lang* attribute is used to identify only the English posts and ignore posts in other language which are of less relevance. *Retweeted* attribute is used to avoid consideration of retweeted tweets which unnecessarily increases the workload. Basically this integrates keyword-based rules for the implementation. The keywords were identified from a set of categories such as Game chats, Sport commentaries, pornographic content, and Seasonal greetings content. These seemed to be the most common Twitter categories. As code snippets are not shared in Twitter due to character limitation it was not considered. In Twitter pre filter *isFilterPassed()* method is overridden to check whether the incoming tweet move forward the system or not. There *isContainKeyword()* method is used to match the incoming tweet against the predefined set of keywords and remove irrelevant tweets.

## 4.4 Context Filter Implementation

The filtered output from the Pre filter is passed through the Context filter. The Context filter is used to filter-in only the posts relevant to the defined domain. The context defines the information regarding a particular country, nation, organization or an individual that is unique for each entity. For an instance, the context may be the security incidents related to a particular

bank in Sri Lanka. This filter can be optionally used according to user preference. If the user needs to keep track of all the security related posts he/she can ignore the Context filter.

For every data feed, there is only one implementation of Context Filter, because the relevant context does not depend on the data feed. In the Context filter regular expressions and a set of keywords is used to describe the context and the system expands the word list using the NLP tool WordNet [39], which is an English lexical database of synonyms. Domain related keywords are identified using the created WordNet of interrelated words.

**Defining the Information Domain**

Defining the information domain related to a particular organization must be done considering multiple facts related to security and sensitivity. It requires the domain knowledge of a business domain expert, as well as an information security expert. Formulation of precise the keyword domain will improve the precision of the monitoring platform. To improve the accuracy of the detection rate, it is required to expand the set of keywords, to cover a domain of the target entity. This will introduce further false positives that will reduce the precision, but will maximize the recall. Expanding scope also provides the space for attack forecast and identify trending movements related to a particular target.


*Figure 4-4: Tweet related to Sri Lankan domain*

Named Entity Recognition (NER) technologies use keywords to identify the entities. Therefore, if the document does not contain the specific keywords defining the target object, monitoring platform will not consider that post as relevant. For instance, a post with an evidence of an attack may contain the phrase “series of defacement attacks against the

government websites of southeast Asia". Such a post will not be detected as relevant to Sri Lankan domain as the scope is larger than the defined domain. Figure 4-4 shows a post that will be detected as relevant to Sri Lankan domain as it contains words "Sri Lanka's Commercial Bank" and will be passed to next classifier.

In LeakHawk 2.0 we focus on the sensitive information leakages and evidence of hacking attacks related to Sri Lankan domain. An Information Template defined for Sri Lanka, with respective examples is illustrated in Table 4. Having an information template allows to cover all the words related to a particular domain and that will minimize the probability to miss any related words.

*Table 4: Information template defined for Sri Lanka*

| **Identifier** | **Description** | **Example** |
|---|---|---|
| Country Identification names | A particular country can be identified using different terms. Names of the major cities can be mentioned instead of the country name. In some cases, the country is referred with indirect terms. | Sri Lanka<br>Lanka<br>Ceylon<br>LK<br>Colombo<br>South Asia |
| Nation and communities | Sometimes without mentioning the country name, distinct communities are targeted. This should not include the domains, which could add a lot of false positives. | Sinhala<br>Sinhalese<br>Buddhist<br>Muslim |
| Unique identifier formats of the citizens | When a large community is targeted, unique identifiers could be exposed. Regular expressions to identify using such identifiers should be used. | National Identity Card number<br>Driving License Number<br>Passport Number |
| Domain names | Use of regular expressions to identify the domains names related to Sri Lanka. e.g. government websites (domain name ending with gov.lk)<br>LK domains in general (domain names ending with .lk).<br>Domain names containing Identification names related to Sri Lanka. | www.president.gov.lk<br>example.lk<br>example.lk.com<br>srilanka.com<br>lanka.org |
| IP addresses related to Sri Lanka | In certain cases, the IP addresses within the Sri Lanka could be involved in a particular attack. WHOIS database [40] can be utilized to identify the location of a particular IP address. | 112.134.100.10<br>222.165.128.4 |

| | | |
|---|---|---|
| Credit / Debit Card ranges | Bank Identification Number (BIN) ranges are defined uniquely to identify each issuing bank in the world.<br>This list should also cover any BIN ranges of the local payment brands (e.g., LankaPay) | |
| Popular characters in domain | This list may contain some popular characters who could be subjected to an online attack. | President of Sri Lanka<br>Prime minister of Sri Lanka<br>Popular businessmen |
| Major organizations and corporations | Certain posts may directly mention the organization names without mentioning the country name. So it is safe to search for those names separately. | Mobile and Internet service provider names (SLT, Dialog, Etisalat, etc.)<br>Sri Lankan organizations (Banks, Telecommunication companies, Insurance, Finance, Textile, etc.)<br>Corporations (Cargills Ceylon, Keels, Aitken Spence, Hemas, etc.)<br>Famous TV channels |

The defined attributes for a particular domain of Context Filter are implemented using keyword lists and regular expressions. Wordnet API [39] is used to expand the keyword list using connected words. For each received feed from Pre filter, Context filter will execute these logics, and only the positive matches are forwarded to Evidence Classifier. The models developed by the Evidence Classifier and the Content Classifier will only execute, if the Context filter is passed.

## 4.5 Evidence Classifier Implementation

LeakHawk use an Evidence classifier to identify whether the post is a sensitive one or not. Each data feed has its own Evidence classifier. This classifier should be implemented by extending *LeakHawkClassifier* abstract class as shown in Figure 4-5.

These two methods are used in every Evidence classifier implementation helping to configure and identify the post sensitivity. *prepareClassifier*() method will run only once and that would be in the initialization of the bolt. This method can be used to initialize the things that would be used in the *classifyPost()* method. *classifyPost*() method is used to classify the post into sensitivity category or non-sensitivity category.

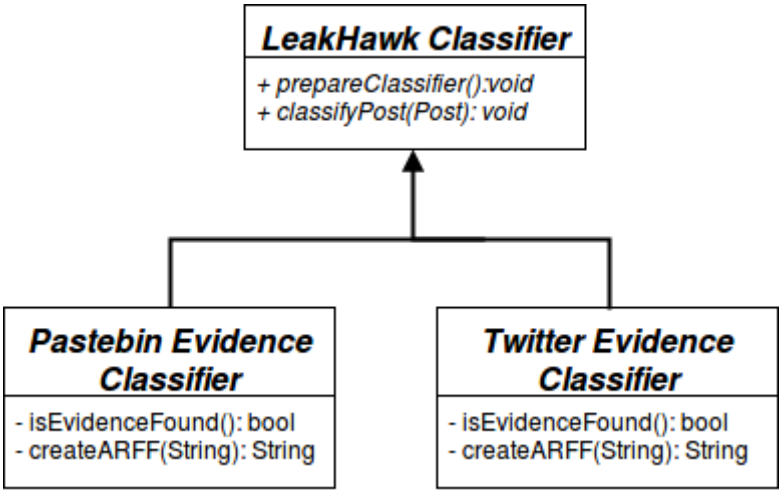


*Figure 4-5: Evidence classifier class diagram*

### 4.5.1 Pastebin

Pastebin Evidence classifier is used to classify an incoming Pastebin post using binary classification technique [36]. Classification is done to check whether the post has an evidence of hacking attack or not. Unigrams, bigrams, and trigrams related to most commonly used hacking attacks related keywords, hackers' names, hackers' slogans, etc., are checked against the incoming post content and titles to identify whether there is evidence of hacking attack or not. *classifyPost*() method is overridden here to predict whether the incoming Pastebin post has an evidence of data breach exposure. Java WEKA API [37] was used for the classification process.

All the posts that come into Evidence classifier are sent to the Content classifier and if that post contains a set of URLs the post is sent to URL Processor to check the content against any possible data breach exposure.

### 4.5.2 Twitter

In Twitter Evidence classifier implementation, binary classification method has been used to identify whether the incoming tweet has an evidence of data breach exposure or not. This classification is also done using Java WEKA API [37]. Most commonly used unigrams, bigrams, and trigrams related to hacking attacks, hacker group names, etc., are identified and used in the classification process. Once classified, all the posts that come into Twitter Evidence classifier are sent to the Twitter Content classifier.

## 4.6 Content Classifier Implementation

LeakHawk use content classifier to divide the post into the correct sensitivity category. For instance, if the post contains data about a credit card dump that should be categorized under credit card related data breach exposure. Each data feed has its own content classifier. These classifiers should be implemented by extending from LeakHawkClassifier abstract class.

The following two methods are used in every Content Classifier implementation helping to configure and identify the post sensitivity. *prepareClassifier()* method will be run only once and that would be in the initialization of the bolt. This method can be used to initialize the things that would use in the *classifyPost()* method. *classifyPost()* method is used to classify the post into different categories define by the user.

### 4.6.1 Pastebin

The Pastebin Content Classifier categorizes the incoming post into one or more from nine categories (see Table 5).

*Table 5: Categories for Pastebin content classifiers*

| Classifier | Abbreviated Name |
|---|---|
| Credit Card | CC |
| Configuration Files | CF |
| DNS Attack | DA |
| Database Dump | DB |
| Email Conversation | EC |
| Email Only | EO |
| Private keys | PK |
| User Credentials | UC |
| Website Defacement | WD |

Each of the above mentioned classifiers classifies post using binary classification, which is a supervised Machine Learning solution. These classifiers are written as a sub-class which needs to be extended from the *ContentClassifier* abstract class.

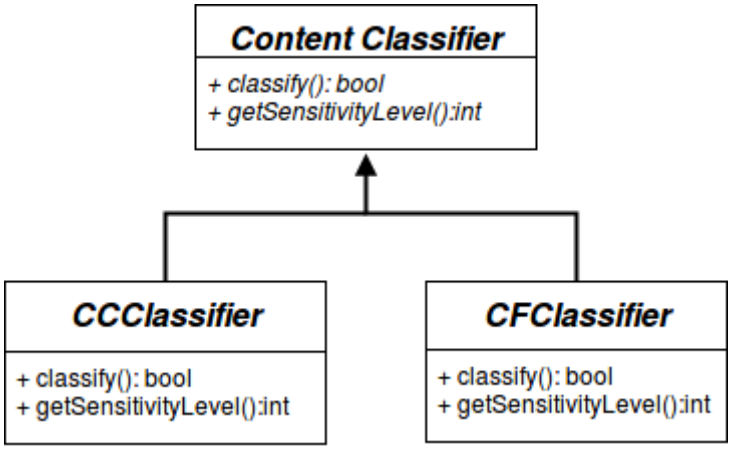


*Figure 4-6: Example content classifier class diagram*

As shown in the example implementation in the Figure 4-6, every classifier class should override the *classify()* and *getSensitivityLevel()* methods. *classify*() method will process the post and will play a binary classification on the post. This classification result will be return in the classify method. The classification methods in the inbuilt classify classes can be easily customized by overriding the *classify()* method. *getSensitivityLevel()* method should return the sensitivity level (LOW, HIGH, or CRITICAL) of the post. Through a customized implementation of this method by overriding the method will make it easy to change the way of declaration of sensitivity level criteria.

Users can add new classification categories easily to the system by adding a new subclass extending *ContentClassifier* class into Content folder by overriding the two methods *classify()* and *getSensitivityLevel()*. Custom annotations are used to identify the classification classes, so user has to use *ContentPattern* custom annotations to the newly added classification class. With this annotation user has to provide pattern name and classification model file path. As implemented in the *PastebinContentClassifier*, posts go through all the nine classification classes mentioned above, and if it is classified as true then the post will be categorized under that class. Classification may give result as true for several classification classes, so one post may be categorized into one or more of the nine defined classes. All the posts that come into Pastebin Content classifier are sent to the Synthesizer.

### 4.6.2 Twitter

Twitter Content classifier uses keyword based and regular-expression based rules for the categorization. Most commonly used keywords in identified categories are used to get a

match. This classifier enforces seven categories of possible data breach exposures as shown in Table 4.x.

*Table 6: Categories for the twitter content classifiers*

| Classifier | Abbreviated Name |
|---|---|
| Credit Card | CC |
| DNS Attack | DA |
| Database Dump | DB |
| Email Only | EO |
| Private keys | PK |
| User Credentials | UC |
| Website Defacement | WD |

Inside *classifyPost* method keywords are matched and if a match is found to a particular category, the tweet is categorized into that category. If the post matches to several categories, it is categorized under several categories. Finally, all the posts that come into Twitter Content classifier are sent to the Synthesizer.

## 4.7 Synthesizer

Synthesizer is used to predict the sensitivity level of the incoming post as CRITICAL, HIGH, or LOW. To synthesize Pastebin posts *synthesizePastebinPosts()* method is used and to synthesize tweets *synthesizeTweets()* method is used.

*synthesizePastebinPosts()* method is implemented to predict the sensitivity level of the post by comparing the sensitivity levels predicted from each one of nine classifiers in Content classifier. Sensitivity prediction is mainly done by considering the results of Pastebin Content classifier. The highest level predicted from the classifiers is taken as the sensitivity level of the post. As illustrated in Table 7 Synthesizer predicts a label for the sensitivity of the post as CRITICAL, HIGH, or LOW as per the content of the post, its semantics, and number of items got compromised.

*Table 7: Sensitivity levels for the Synthesizer*

| | CRITICAL | HIGH | LOW |
|---|---|---|---|
| **Credit card dumps** | Credit card numbers > 20 | 5 < Credit card numbers < 20 | Credit card numbers < 5 |
| **Configuration files** | The post contains passwords | | |
| **Defacement attack** | Matched keywords related to domain > 10 | Matched keywords related to domain < 10 | |
| **Email conversation** | Matched keywords related to email conversations > 0 | | |
| **Private keys** | Presence of private keys | | |
| **Email only list** | | Email count > 50 | Email count < 50 |
| **User credentials** | Hash count > 20 | 5 < Hash count < 20 | Hash count < 5 |
| **Web Defacement** | URL count >20 | 5 < URL count < 20 | URL count < 5 |
| **DB dumps** | | Presence of DB dumps | |

## 4.8 LeakHawk Class Diagram

Main class diagram of LeakHawk 2.0 is show in the figure 4-7.

LeakHawkBolt abstract class is extended from *BasicRichBolt* super class of Apache Storm in order to make the implementation simpler and let a person without expert knowledge on Storm could use the system without any issues. LeakHawkFilter and *LeakHawkClassifier* abstract classes and *PostDownloader* and *URLProceesor* concrete classes are extended from *LeakHawkBolt* class by overriding *perpareBolt(), getBoltName(), execute(),* and *declareOutputStreams()* methods.

*PastebinPreFilter*, *TwitterPreFilter*, and *ContextFilter* classes extend the LeakHawlFilter abstract class. The level of abstraction in the design has made lower level classes to be implemented without directly knowing the behaviour of bolts in Storm. All the classifiers like Evidence classifier, Content classifier and Synthesizer are extended from *LeakHawkClassifier* abstract class by overriding necessary methods.

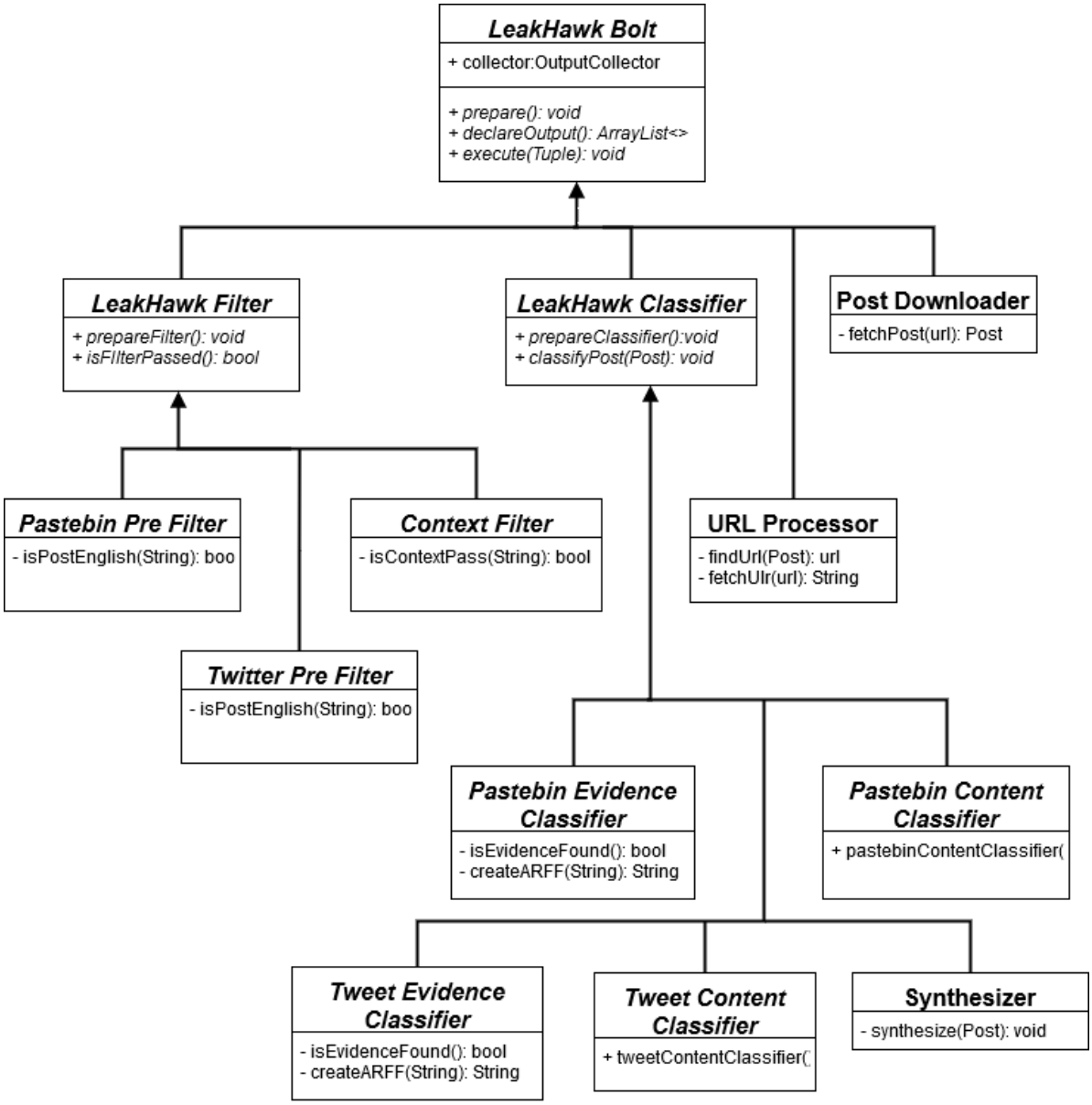


*Figure 4-7: Main class diagram of LeakHawk 2.0*

Figure 4-8 shows how sensors are designed to be implemented. LeakHawkProducer class returns a *KafkaProducer* to *LeakHawkSensor* class which turns use it to queue messages from different data origins. LeakHawkSensor class is a thread itself and *PastebinSensor* and *TweetsSensor* are subclasses acting as threads while retrieving content from data origins.

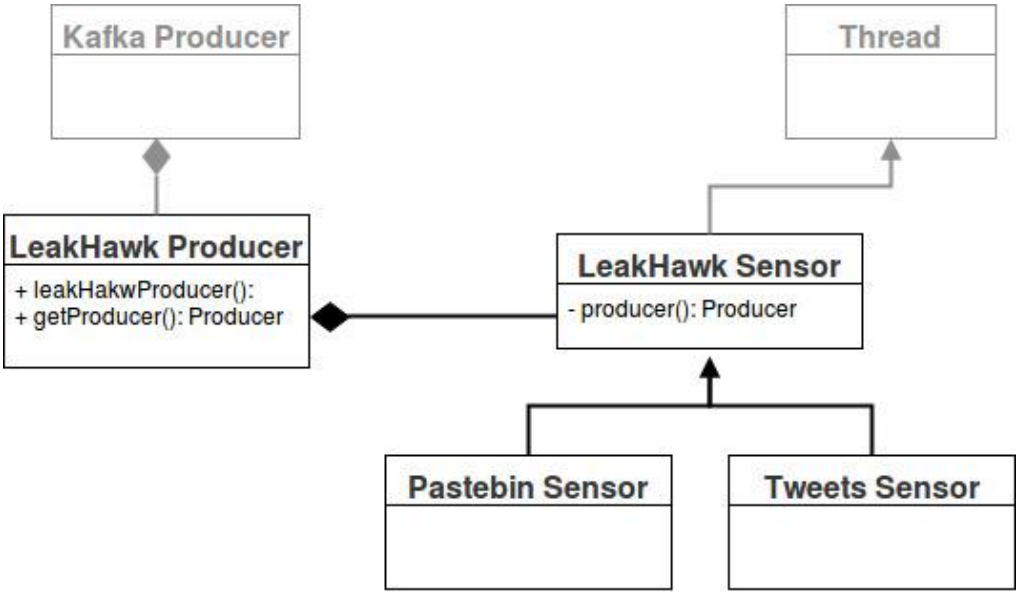


*Figure 4-8: Sensor class diagram of LeakHawk 2.0*

## 4.9 Dashboard Implementation

LeakHawk 2.0 contain a user dashboard which enables the user to interact with the LeakHawk System as well as see the results. Dashboard is implemented with AngularJS front end and SpringBoot backend. User functionalities of Dashboard can be identified as follows:

- View sensitive post list
- View each sensitive post detail
- View posts by sensitivity level
- View post counts going through the classifiers
- View analysis of filters and classifiers
- Start/Stop LeakHawk system
- Add/Stop data feed to leakhawk
- Edit settings of LeakHawk

LeakHawk system is built with two maven modules called “leakhawk-core” and “leakhawk-monitor”. “leakhawk-core” module contains the leakhawk system core functionalities and “leakhawk-monitor” contains the dashboard REST API and the web application. Maven multiple modules concept is used in the system and “leakhawk-monitor” module contains the “leakhawk-core” module as a dependency. With a single build, leakhawk-core module will be compiled and it will be added to the leakhawk-monitor build as a dependency.

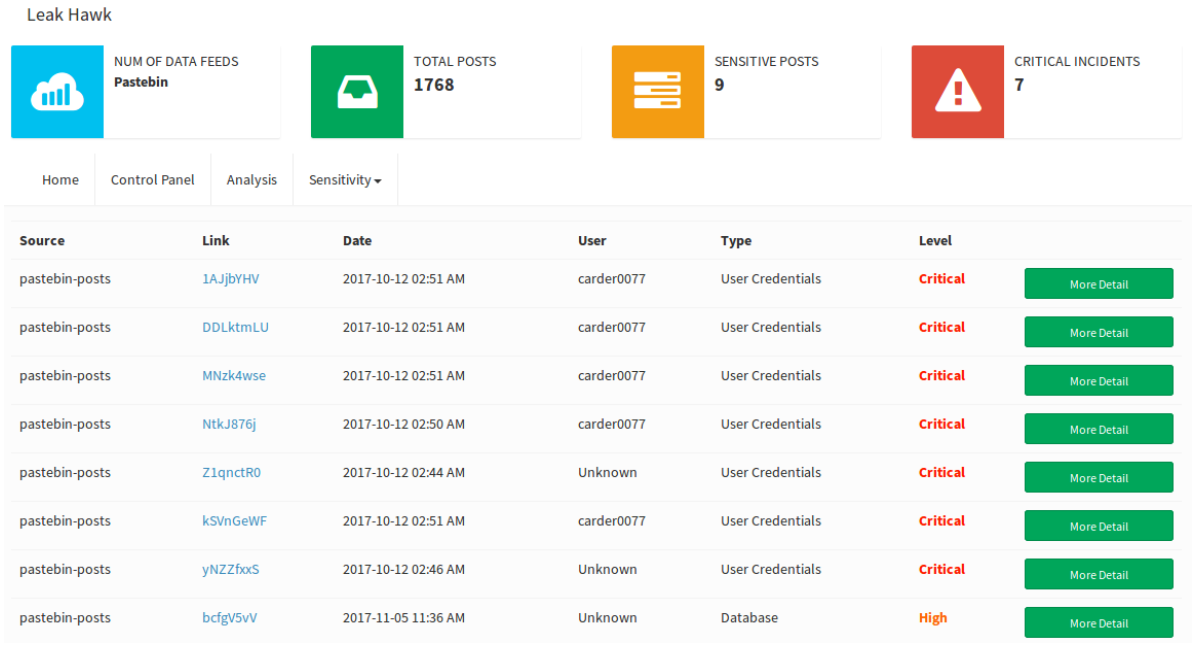


*Figure 4-9: Main view of the Dashboard*

The administrators can view the sensitive incidents in the application main view. (See Figure 4-9) This view will only provide brief details about the incident. Further admins will also be able to view the sensitive incident through the provided link. Admin can see more details about the incident on see “Incident Details” window (see Figure 4-10). Users can control LeakHawk system through user interface. Figure 4-11 shows the Control Panel, where users can start the LeakHawk system and add data feeds through this interface and set configuration parameters.

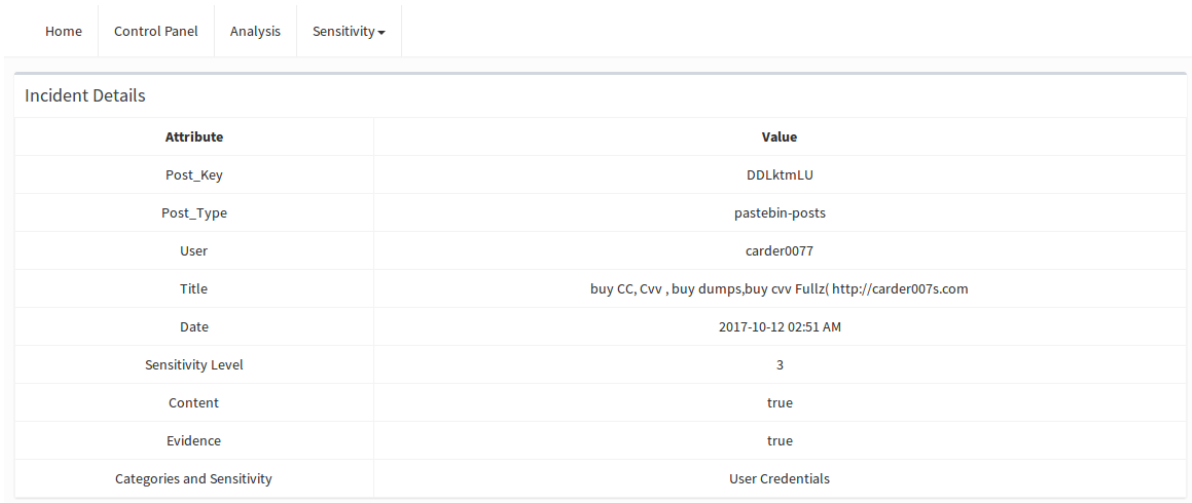


*Figure 4-10: Incident details window*

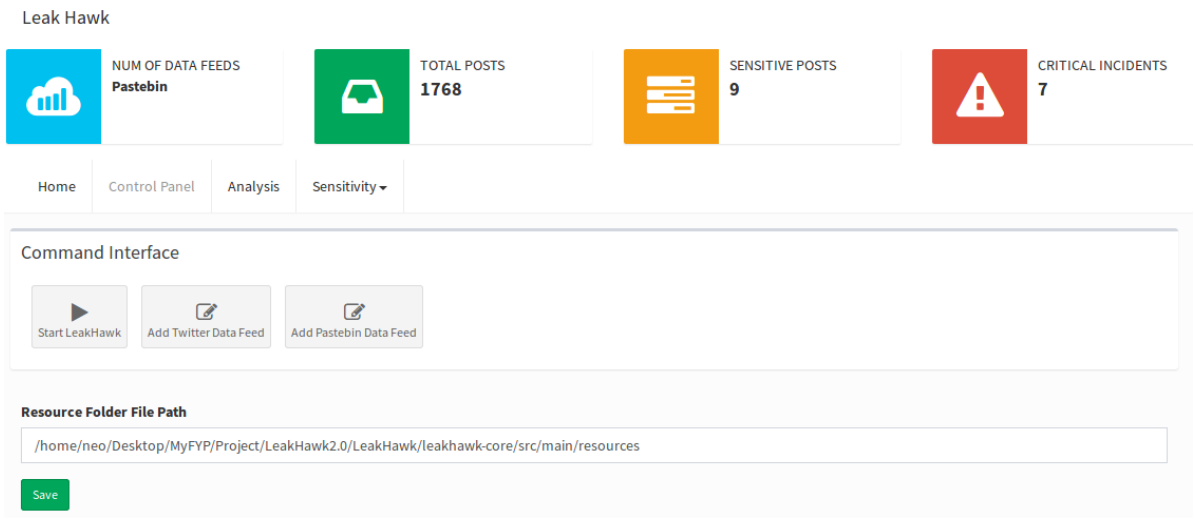


*Figure 4-11: Control panel in the Dashboard*

Admin can further analyze the overall statistics of data leakage detection of LeakHawk through graphs with the window shown in Figure 4-12. It shows the statistics about the filters and how much data are filtered by filters and how much data are classified by classifiers.

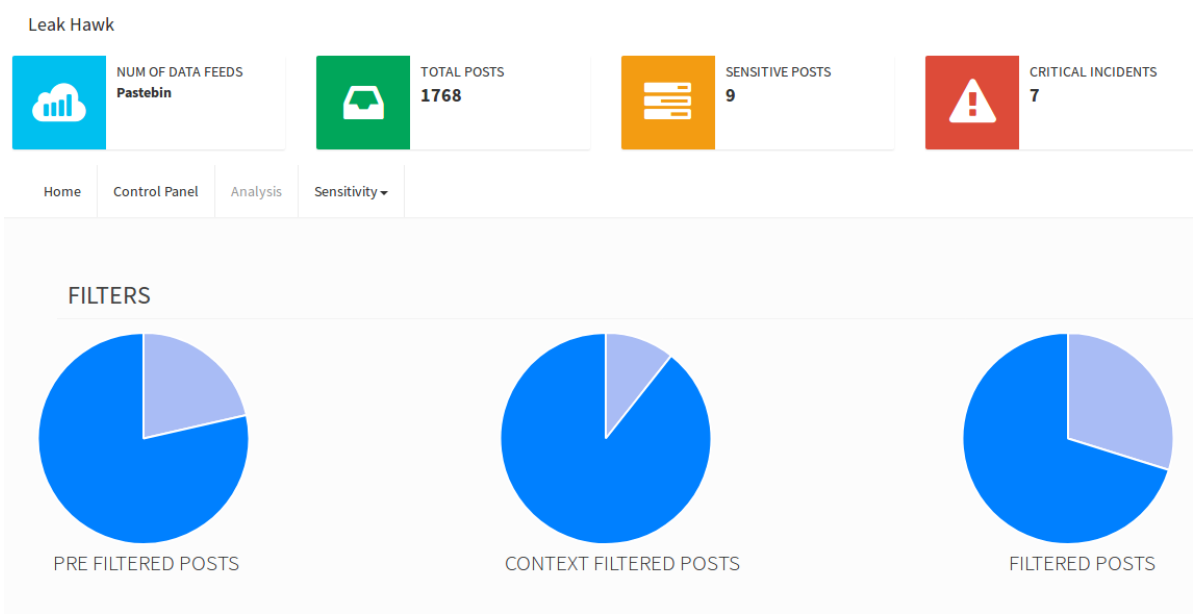


*Figure 4-12: Sample statics view in the Dashboard*

# 5 Performance Analysis

LeakHawk's multi-layer architecture includes multiple components designed to enhance system performance, while minimizing the number of false-negatives and maximizing recall. This section analyses the performance with regard to accuracy and time of the components separately and throughput, memory usage, and network usage of the overall system. Section 5.1 analyses the performance of the sensors used by LeakHawk 2.0. Section 5.2 analyses the accuracy of both Pastebin and Twitter filters and classifiers separately. Section 5.3 shows comparison between LeakHawk 1.0 and 2.0. Analysis the overall system performance of LeakHawk with regard to time, throughput, and memory usage is presented in Section 5.4.

## 5.1 Analysis of filters and Classifiers

We evaluate the performance on a single node with the following configuration:

Model of the computer - HP ProBook 4540s Notebook
CPU - Intel Core i5-3230M running at 2.6GHz (32KiB L1, 256KiB L2 and 3MiB L3 cache)
Memory - 8GB DDR3 RAM running at 1600 MHz
Operating System - Ubuntu 16.04.3 LTS x86_64

LeakHawk 2.0 has mainly two filters, namely Pre filter and Context filter and two classifiers, namely Evidence and Content classifier. This section analyses the precision and recall of all the components by providing separate datasets for each filter and classifier. Unique datasets are used in the accuracy analysis of each component to match their requirements. For instance the dataset used for Pre filter will not match the features of Context filter or Evidence classifier and separate datasets should be used for each class in Content classifier to match the sensitive content.

### 5.1.1 Pastebin Pre Filter

The Pre filter screens out the posts, which are non-sensitive in nature, such as video game chat sessions, pornographic content, and torrent information. It also eliminates non-textual posts such as binary files. As the average number of posts made in Pastebin is less than 50, this filter was not useful in that scenario except for the exclusion of binary inputs. However, when the model is extended to support Twitter feeds, Pre filter effectively improves the performance of the subsequent filters and classifiers by removing unrelated posts beforehand.

#### 5.1.1.1 Training Pre filter model

The training corpus used to create Machine Learning model contained 2,011 positive posts and 714 negative posts, which were used both as the training and testing set during the validation process. Ten-fold cross validation was done on the dataset to get more precise results.

The performance results obtained after cross validation on Random Forest, Support vector machine, and Naive Bayes multi-nominal algorithms are illustrated in Figure 5-1, Figure 5-2 and Figure 5-3 respectively. Based on these results RandomForest algorithm gives the best results while classifying the irrelevant posts.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2181               80.0367 %
Incorrectly Classified Instances       544               19.9633 %
Kappa statistic                          0.3914
Mean absolute error                      0.2935
Root mean squared error                  0.3951
Relative absolute error                 75.8864 %
Root relative squared error             89.8481 %
Total Number of Instances             2725

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.948    0.616    0.813      0.948    0.875      0.422    0.692     0.832     pos
                 0.384    0.052    0.725      0.384    0.502      0.422    0.692     0.564     neg
Weighted Avg.    0.800    0.468    0.790      0.800    0.777      0.422    0.692     0.762

=== Confusion Matrix ===

    a    b   <-- classified as
 1907  104 |    a = pos
  440  274 |    b = neg
```

*Figure 5-1: Pre filter model using Random Forest algorithm*

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2114               77.578  %
Incorrectly Classified Instances       611               22.422  %
Kappa statistic                          0.34
Mean absolute error                      0.3279
Root mean squared error                  0.4377
Relative absolute error                 84.7603 %
Root relative squared error             99.5333 %
Total Number of Instances             2725

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.915    0.616    0.807      0.915    0.858      0.355    0.626     0.782     pos
                 0.384    0.085    0.616      0.384    0.473      0.355    0.627     0.490     neg
Weighted Avg.    0.776    0.477    0.757      0.776    0.757      0.355    0.626     0.706

=== Confusion Matrix ===

    a    b   <-- classified as
 1840  171 |    a = pos
  440  274 |    b = neg
```

*Figure 5-2: Pre filter model using Support vector machine algorithm*

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2035               74.6789 %
Incorrectly Classified Instances       690               25.3211 %
Kappa statistic                          0.0612
Mean absolute error                      0.2532
Root mean squared error                  0.5032
Relative absolute error                 65.4593 %
Root relative squared error            114.4332 %
Total Number of Instances             2725

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.995    0.952    0.746      0.995    0.853      0.149    0.521     0.746     pos
                 0.048    0.005    0.773      0.048    0.090      0.149    0.521     0.286     neg
Weighted Avg.    0.747    0.704    0.753      0.747    0.653      0.149    0.521     0.626

=== Confusion Matrix ===

    a    b   <-- classified as
 2001   10 |    a = pos
  680   34 |    b = neg
```

*Figure 5-3: Pre filter model using Naive Bayes Multinomial algorithm*

#### 5.1.1.2 Performance testing of Pre filter

Table 8 illustrates the results of seeding 2,725 samples of textual documents across the Pre filter. The seed contains 2,011 manually labeled posts that are pre validated as related posts. Ideally, the filter should identify 2,011 positive samples and 714 negative samples (total number of posts used for testing is 2,725). The table lists the positive matches selected by the

Pre filter. *True positives* denote the correct matches, while *False Positives* denote the number of documents selected by the filter which is not relevant.

*Table 8: Pastebin Pre filter analysis*

| **Positive** | | **Negative** | |
|---|---|---|---|
| 2011 | | 714 | |
| **True Positive** | **False Negative** | **True Negative** | **False Positive** |
| 1997 | 14 | 571 | 143 |
| 99.31% | 0.69% | 79.97% | 20.03% |

As per the table, following key observations are made:

$$\text{Precision} = \frac{1997}{1997+143} = 93.32\%$$

$$\text{Recall} = \frac{1997}{1997+14} = 99.31\%$$

It can be seen that false positives are a little bit higher as the Pre filter model accuracy was 80.04%. This is because the positive posts sometimes contain the keywords used to filter out the unrelated posts. For instance a sensitive post might contains words such as “Greetings, Best of Luck, Happy new Year, etc.”

### 5.1.2 Context Filter

The Context filter is a common component for every feed retrieved by LeakHawk, which allows the user to define the information domain, which is used by the LeakHawk Core as the context for monitoring pre-defined targets. Table 9 illustrates the results of seeding 2,700 samples of textual documents across the Context Filter. The seed contains 1,200 manually labeled posts that are pre validated as related to Sri Lanka. Ideally, the filter should identify 1,200 positive samples and 1,500 negative samples (total number of posts used for testing is 2,700). The table lists the positive matches selected by the Context Filter. True positives denote the correct matches related to Sri Lanka, while False Positives denote the number of documents selected by the filter which is not relevant to Sri Lanka.

*Table 9: Context filter performance analysis*

| **Positive** | | **Negative** | |
|---|---|---|---|
| 1200 | | 1500 | |
| **True Positive** | **False Negative** | **True Negative** | **False Positive** |
| 1200 | 0 | 1237 | 263 |
| 100% | 0% | 82.47% | 17.53% |

According to the labeled dataset, following key observations are made:

Precision = $\frac{1200}{1200+263}$ =82.02%

Recall = $\frac{1200}{1200+0}$ = 100%

Keywords such as “Lanka”, “Sri Lanka” and “LK” are accountable for most of the results (irrespective of the accuracy). However, the usage of “LK” introduces a considerable number of false positives. Pattern matching methods identify certain results, which are not captured by the above keywords but result in many false positives. Therefore, it is evident that the use of multiple identifiers is necessary for the successful identification of positive instances with minimal false-negatives.

Identifying all the keywords and regular expressions is a tedious task, which involves a considerable amount of manual effort. There may be many other words that can be used to catch the domain related words.

### 5.1.3 Pastebin Evidence Classifier

Once the posts pass the Context filter they reach the Evidence classifier and it checks for evidence of hacking attacks and data breaches. A model with 94.16% accuracy is used in the Evidence classifier.

#### 5.1.3.1 Training Evidence classifier model

A training corpus of 1,542 was used which contained 1,004 negative posts and 538 positive posts to train the classifier which became the test and training set used for cross validation. Here ten-fold cross validation was performed with the same training corpus used above as the

test dataset and training dataset. Validation results of the dataset for Random Forest, Naive Bayes Multi-nominal, and Support Vector Machine classifiers are shown in Figure 5-3, Figure 5-4 and Figure 5-6 respectively. Based on the results Random Forest algorithm has better ability to classify the posts for evidence.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1452               94.1634 %
Incorrectly Classified Instances        90                5.8366 %
Kappa statistic                          0.8684
Mean absolute error                      0.108
Root mean squared error                  0.234
Relative absolute error                 23.7728 %
Root relative squared error             49.0895 %
Total Number of Instances             1542

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.864    0.017    0.965      0.864    0.912      0.871    0.914     0.908     pos
                 0.983    0.136    0.931      0.983    0.956      0.871    0.914     0.914     neg
Weighted Avg.    0.942    0.094    0.943      0.942    0.941      0.871    0.914     0.912

=== Confusion Matrix ===

   a   b   <-- classified as
 465  73 |   a = pos
  17 987 |   b = neg
```

*Figure 5-4: Evidence classifier model using Random Forest algorithm*

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1236               80.1556 %
Incorrectly Classified Instances       306               19.8444 %
Kappa statistic                          0.4993
Mean absolute error                      0.4107
Root mean squared error                  0.4426
Relative absolute error                 90.3846 %
Root relative squared error             92.868  %
Total Number of Instances             1542

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.441    0.005    0.979      0.441    0.608      0.571    0.709     0.746     pos
                 0.995    0.559    0.768      0.995    0.867      0.571    0.709     0.796     neg
Weighted Avg.    0.802    0.366    0.842      0.802    0.777      0.571    0.709     0.779

=== Confusion Matrix ===

   a   b   <-- classified as
 237 301 |   a = pos
   5 999 |   b = neg
```

*Figure 5-5: Evidence classifier model using Naive Bayes Multinomial algorithm*

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1447               93.8392 %
Incorrectly Classified Instances        95                6.1608 %
Kappa statistic                          0.8604
Mean absolute error                      0.0616
Root mean squared error                  0.2482
Relative absolute error                 13.5581 %
Root relative squared error             52.0768 %
Total Number of Instances             1542

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.849    0.014    0.970      0.849    0.906      0.865    0.918     0.877     pos
                 0.986    0.151    0.924      0.986    0.954      0.865    0.918     0.921     neg
Weighted Avg.    0.938    0.103    0.940      0.938    0.937      0.865    0.918     0.905

=== Confusion Matrix ===

   a   b   <-- classified as
 457  81 |   a = pos
  14 990 |   b = neg
```

*Figure 5-6: Evidence classifier model using Support Vector Machine algorithm*

### 5.1.3.2 Performance testing of Evidence classifier

Table 10 illustrates the results of seeding 1,818 samples of posts across the Evidence classifier. The seed contains 860 manually labeled posts that are pre validated as evidence containing (total number of posts used for testing is 1,818). Ideally, the filter should identify 860 positive samples and 958 negative samples. The table lists the positive matches selected by the Evidence classifier. True positives denote the correct matches, while false positive denotes the number of posts selected by the classifier which do not contain evidence of hacking attack or data breach.

*Table 10: Evidence classifier performance analysis*

| **Positive** | | **Negative** | |
|---|---|---|---|
| 860 | | 958 | |
| **True Positive** | **False Negative** | **True Negative** | **False Positive** |
| 814 | 46 | 916 | 42 |
| 94.65% | 5.35% | 95.62% | 4.38% |

As per the table, following key observations are made:

$$\text{Precision} = \frac{814}{814+42} = 95.09\%$$

$$\text{Recall} = \frac{814}{814+46} = 94.65\%$$

The set of attributes considered when creating the Evidence classifier model might contain some words that are not available in most of the posts. The model can be created using a well analyzed and optimized set of attributes.

### 5.1.4 Pastebin Content Classifier

Once the posts pass the Evidence classifier they reach the Content classifier and it checks whether the post contains sensitive data such as credit cards, emails, and passwords. Content classifier has nine classes with separate models, each which checks the posts for the availability of the class content.

Table 11 illustrates the results of seeding samples of posts across each class of the Content classifier. Each class is analyzed with different data sets to match the requirements. Table 12 presents the precision and recall values of each class in Content classifier. Figure 12 illustrates the distribution of precision and recall values of each content class. True positives denote the correct matches, while false positives denote the number of posts selected by the classes which do not contain sensitive content.

*Table 11: Pastebin Content classifier performance analysis*

| Content Classes | Positive | Negative | True Positive | False Negative | True Negative | False Positive |
|---|---|---|---|---|---|---|
| [CC] Credit Card | 299 | 300 | 296 | 3 | 298 | 2 |
| [UC] User Credentials | 350 | 347 | 302 | 48 | 305 | 42 |
| [DB] Database | 159 | 166 | 154 | 5 | 130 | 36 |
| [DA] DNS Attack | 100 | 100 | 98 | 2 | 89 | 11 |
| [EO] Email Only | 166 | 166 | 166 | 0 | 165 | 1 |
| [PK] Private Key | 100 | 100 | 97 | 3 | 100 | 0 |
| [EC] Email Conversation | 60 | 60 | 59 | 1 | 59 | 1 |
| [CF] Configuration Files | 164 | 164 | 164 | 0 | 133 | 31 |
| [WD] Website Defacement | 274 | 274 | 256 | 18 | 218 | 56 |

*Table 12: Content classifier accuracy analysis*

| Class | Precision | Recall |
|---|---|---|
| CC | 98.99% | 99.32% |
| UC | 86.29% | 87.79% |
| DB | 96.86% | 81.05% |
| DA | 98% | 89.9% |
| EO | 100% | 99.4% |
| PK | 97% | 100% |
| EC | 98.33% | 98.33% |
| CF | 100% | 84.1% |
| WD | 93.43% | 82.05% |

As per the graph, the Content classes Credit Card (CC), Database (DB), DNS Attack (DA), Email Only (EO), Private Key (PK), Email Conversation (EC), and Configuration Files (CF) has better performance in terms of precision. Email Only (EO) and Configuration Files (CF) have 100% precision that indicates when the classifier predicts a set of inputs as Email only or Configuration Files, that positive dataset will contain the majority of these classes in the dataset with significant sensitivity. The CC, EO, PK, and EC classes indicate better performance in terms of recall. Further analysis suggests that the majority of false negatives associated with the UC are the dumps with passwords (not containing attributes that can be extracted with patterns such as e-mails and hashes). Heuristics defined for the UC are not dominant enough to identify particular password dumps.

### 5.1.5 Twitter Pre Filter

Average Twitter feed is about 6,000 tweets per second and Twitter Pre filter comes in handy to improve the performance of the subsequent filters and classifiers by removing unrelated posts beforehand. Table 13 illustrates the results of seeding 1,803 samples of tweets across Twitter Pre filter. The seed contains 853 manually labeled posts that are pre validated as related posts. Total number of posts used for testing is 1,803. Ideally, the filter should identify 853 positive samples and 950 negative samples. The table lists the positive matches selected by the Pre filter. True positives denote the correct matches, while false positives denote the number of documents selected by the filter which is not relevant.

*Table 13: Twitter Pre filter performance analysis*

| **Positive** | | **Negative** | |
|---|---|---|---|
| 853 | | 950 | |
| **True Positive** | **False Negative** | **True Negative** | **False Positive** |
| 640 | 213 | 747 | 203 |
| 75.03% | 24.97% | 78.63% | 21.37% |

According to the labeled dataset, following key observations are made:

$$\text{Precision} = \frac{640}{640+203} = 75.92\%$$

$$\text{Recall} = \frac{640}{640+213} = 75.03\%$$

The precision has decreased because some posts in the selected negative dataset does not contain the identified irrelevant words. Recall has reduced as the positive dataset contains some words identified as irrelevant. To increase the precision and recall the keywords used in Twitter Pre filter should be optimized and the dataset should be selected more accurately.

### 5.1.6 Twitter Evidence Classifier

Once the tweets pass the Context filter they reach the Evidence classifier and it checks for evidence of hacking attacks and data breaches. A model with 99.66% accuracy is used in the Twitter Evidence classifier.

Validation results of the dataset for Random Forest algorithm is show in the Figure 5-7.

Table 14 illustrates the results of seeding 971 samples of posts across the Evidence classifier. The seed contains 485 manually labeled posts that are pre validated as evidence containing. Total number of posts used for testing is 971. Ideally, the filter should identify 485 positive samples and 486 negative samples. The table lists the positive matches selected by the Evidence classifier. True positives denote the correct matches, while false positive denotes the number of posts selected by the classifier which do not contain evidence of hacking attack or data breach.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         968               99.691  %
Incorrectly Classified Instances         3                0.309  %
Kappa statistic                          0.9938
Mean absolute error                      0.0089
Root mean squared error                  0.0617
Relative absolute error                  1.7876 %
Root relative squared error             12.3437 %
Total Number of Instances              971

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.006    0.994      1.000    0.997      0.994    0.998     0.997     pos
                 0.994    0.000    1.000      0.994    0.997      0.994    0.998     0.998     neg
Weighted Avg.    0.997    0.003    0.997      0.997    0.997      0.994    0.998     0.997

=== Confusion Matrix ===

   a   b   <-- classified as
 485   0 |   a = pos
   3 483 |   b = neg
```

*Figure 5-7: Evidence classifier model using Random Forest algorithm*

*Table 14: Twitter Evidence classifier performance analysis*

| **Positive** | | **Negative** | |
|---|---|---|---|
| 485 | | 486 | |
| **True Positive** | **False Negative** | **True Negative** | **False Positive** |
| 485 | 0 | 483 | 3 |
| 100% | 0% | 99.38% | 0.62% |

According to the labeled dataset, following key observations are made:

$$\text{Precision} = \frac{485}{485+3} = 99.39\%$$

$$\text{Recall} = \frac{485}{485} = 100\%$$

The main reason for the high precision and recall can be identified as the higher accuracy in the evidence model and the selection of an optimized dataset.

## 5.2 Comparison between LeakHawk 1.0 and 2.0

LeakHawk 1.0 only has Pastebin components, so each component is analyzed with same dataset used to test LeakHawk 2.0 and the results are compared in this section.

### 5.2.1 Pre Filter of LeakHawk 1.0

Table 7-8 illustrates the results of seeding 2,725 samples of posts across Pre filter of LeakHawk 1.0. The seed contains 853 manually labeled posts that are pre validated as related posts. Ideally, the filter should identify 853 positive samples and 950 negative samples.

*Table 15: LeakHawk 1.0 Pre filter performance analysis*

| **Positive** | | **Negative** | |
|---|---|---|---|
| 2011 | | 714 | |
| **True Positive** | **False Negative** | **True Negative** | **False Positive** |
| 1498 | 513 | 259 | 455 |
| 74.49% | 25.51% | 36.27% | 63.73% |

Precision = $\frac{1498}{1498+455}$ = 76.70%

Recall = $\frac{1498}{1498+513}$ = 74.49%

Both the precision and recall is low since the set of irrelevant words is not optimized. For instance, some code words such as for, else, public, return etc. may be there in sensitive, relevant posts.

**Comparison of Pre Filter**

*Table 16: Comparison of Pre filter*

| | **v1.0** | **v2.0** |
|---|---|---|
| **Precision** | 76.70% | 93.32% |
| **Recall** | 74.49% | 99.31% |

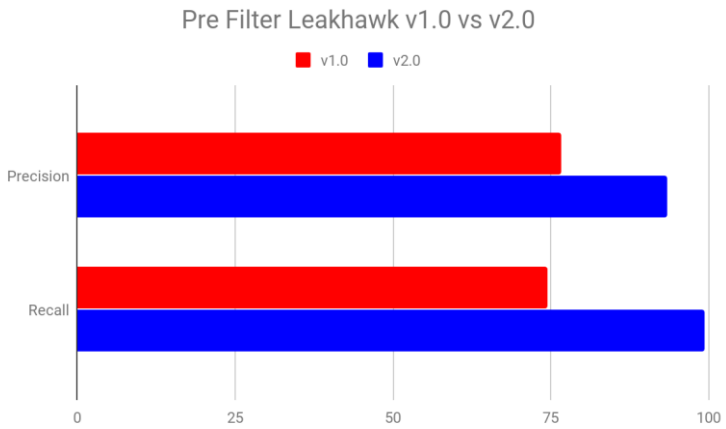


*Figure 5-8: Comparison of Pre filter*

According to Table 16 it is evident that the precision and recall of Pastebin Pre filter in LeakHawk 2.0 is much better than LeakHawk 1.0. The main reason for the improvement can be recognized as the implementation of a model with the accuracy of 84.62% and selection of a proper dataset for training the model, as well as analyzing the performance.

### 5.2.2 Context Filter of LeakHawk 1.0

Table 7-10 illustrates the results of seeding 2,700 samples of posts across Context filter of LeakHawk 1.0. The seed contains 1,200 manually labeled posts that are pre validated as related posts. Ideally, the filter should identify 1,200 positive samples and 1,500 negative samples.

*Table 17: LeakHawk 1.0 Context filter performance analysis*

| **Positive** | | **Negative** | |
|---|---|---|---|
| 1200 | | 1500 | |
| **True Positive** | **False Negative** | **True Negative** | **False Positive** |
| 955 | 245 | 1392 | 108 |
| 79.58% | 20.42% | 92.81% | 7.19% |

$$\text{Precision} = \frac{955}{955+108} = 89.84\%$$

$$\text{Recall} = \frac{955}{955+245} = 79.58\%$$

Precision and recall is a little bit low as the domain related words are not well optimized and with time the word set would be changed slightly.

**Comparison of Context filter**

*Table 18: Comparison of Context filter*

| | **v1.0** | **v2.0** |
|---|---|---|
| **Precision** | 89.84% | 82.02% |
| **Recall** | 79.58% | 100% |

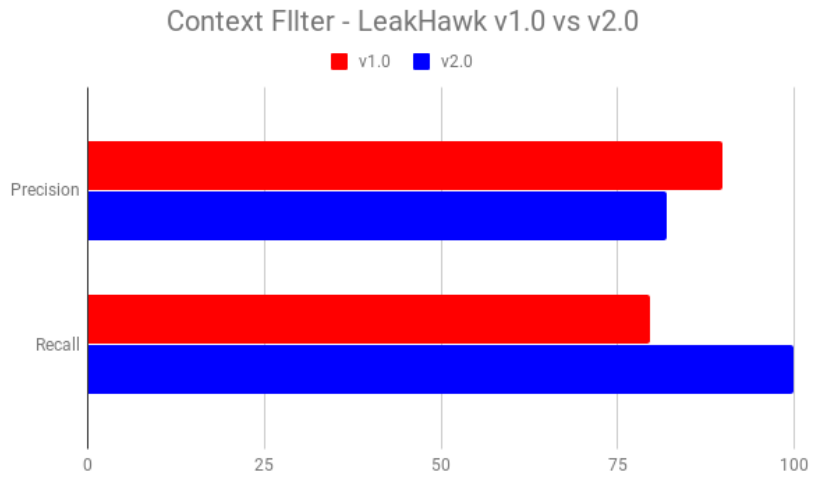


*Figure 5-9: Comparison of Context filter*

According to Table 18 it is found that the precision of Context filter in LeakHawk 2.0 is a bit lower compared to LeakHawk 1.0, while the recall is greatly improved in LeakHawk 2.0. False negatives are almost not detected in LeakHawk 2.0 while in LeakHawk 1.0 there is 20.42% probability of detecting false negatives. The main reason for the improvement can be recognized as the implementation of a model with the accuracy of 84.62% and selection of a proper dataset for training the model as well as analyzing the performance.

### 5.2.3 Evidence Classifier of LeakHawk 1.0

Table 19 illustrates the results of seeding 1,818 samples of posts across Evidence classifier of LeakHawk 1.0. The seed contains 860 manually labeled posts that are pre validated as related posts. Ideally, the filter should identify 860 positive samples and 958 negative samples.

*Table 19: LeakHawk 1.0 Evidence classifier performance analysis*

| **Positive** | | **Negative** | |
|---|---|---|---|
| 860 | | 958 | |
| **True Positive** | **False Negative** | **True Negative** | **False Positive** |
| 650 | 210 | 870 | 88 |
| 75.55% | 24.45% | 90.83% | 9.17% |

Precision = $\frac{650}{650+88}$ = 88.08%

Recall = $\frac{650}{650+210}$ = 75.58%

The recall is less because the words selected to identify the evidences does not contain all the possible evidence related words.

### Comparison of Evidence Classifier

*Table 20: Comparison of Evidence Classifier*

| | **v1.0** | **v2.0** |
|---|---|---|
| **Precision** | 88.08% | 95.09% |
| **Recall** | 75.58% | 94.65% |

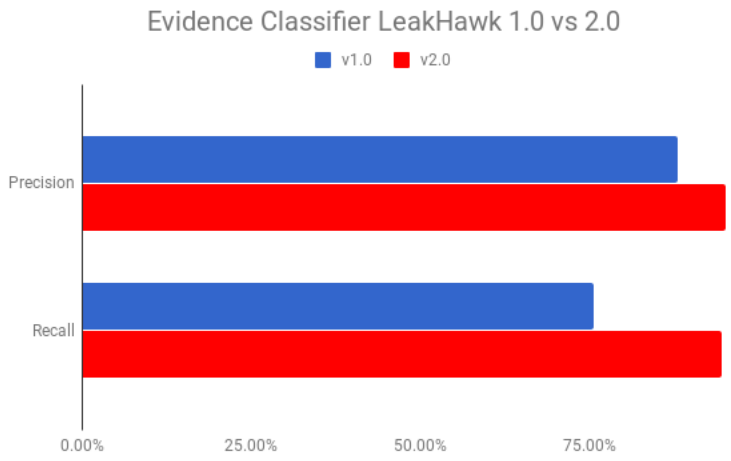


*Figure 5-10: Comparison of Evidence Classifier*

### 5.2.4 Content Classifier of LeakHawk 1.0

Table 21 illustrates the results of seeding samples of posts across each class of the Content classifier in LeakHawk 1.0. Each class is analyzed with different data sets that were used in LeakHawk 2.0.

*Table 21: LeakHawk 1.0 Content Classifiers performance analysis*

| Content Classes | Positive | Negative | True Positive | False Positive | True Negative | False Negative |
|---|---|---|---|---|---|---|
| [CC] Credit Card | 299 | 300 | 264 | 35 | 298 | 2 |
| [UC] User Credentials | 350 | 347 | 329 | 21 | 304 | 43 |
| [DB] Database | 159 | 166 | 150 | 9 | 153 | 13 |
| [DA] DNS Attack | 100 | 100 | 85 | 15 | 89 | 11 |
| [EO] Email Only | 166 | 166 | 163 | 3 | 139 | 27 |
| [PK] Private Key | 100 | 100 | 90 | 10 | 100 | 0 |
| [EC] Email Conversation | 60 | 60 | 48 | 12 | 53 | 7 |
| [CF] Configuration Files | 164 | 164 | 163 | 1 | 159 | 5 |
| [WD] Website Defacement | 274 | 274 | 239 | 35 | 237 | 37 |

### Comparison of Content Classifier

*Table 22: Comparison of Content Classifiers*

| Class | Precision | | Recall | |
|---|---|---|---|---|
| | v1.0 | v2.0 | v1.0 | v2.0 |
| CC | 99.23% | 99.32% | 88.29% | 98.99% |
| UC | 88.44% | 87.79% | 94% | 86.29% |
| DB | 92.02% | 81.05% | 94.34% | 96.86% |
| DA | 88.54% | 89.91% | 85% | 98% |
| EO | 85.79% | 99.40% | 98.19% | 100% |
| PK | 100% | 100% | 90% | 97% |
| EC | 87.28% | 98.33% | 80% | 98.33% |
| CF | 97.02% | 84.10% | 99.39% | 100% |
| WD | 86.59% | 82.05% | 87.23% | 93.3% |
| Average | 91.66% | 91.32% | 90.71 | 96.53% |

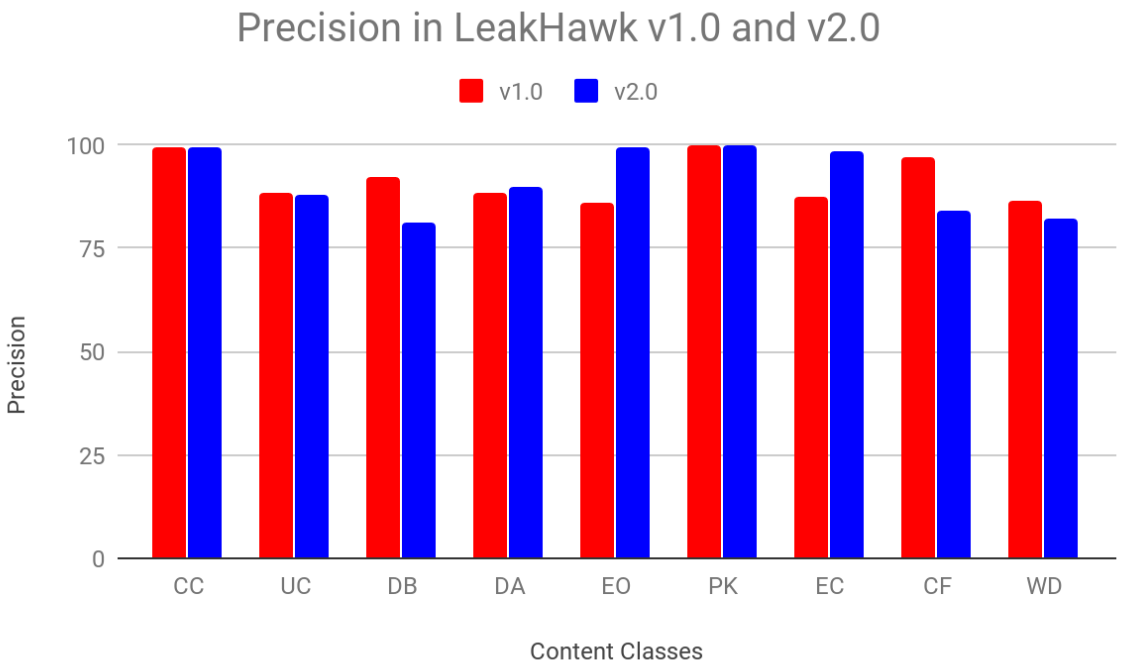


*Figure 5-11: Precision comparison of Content Classifiers*

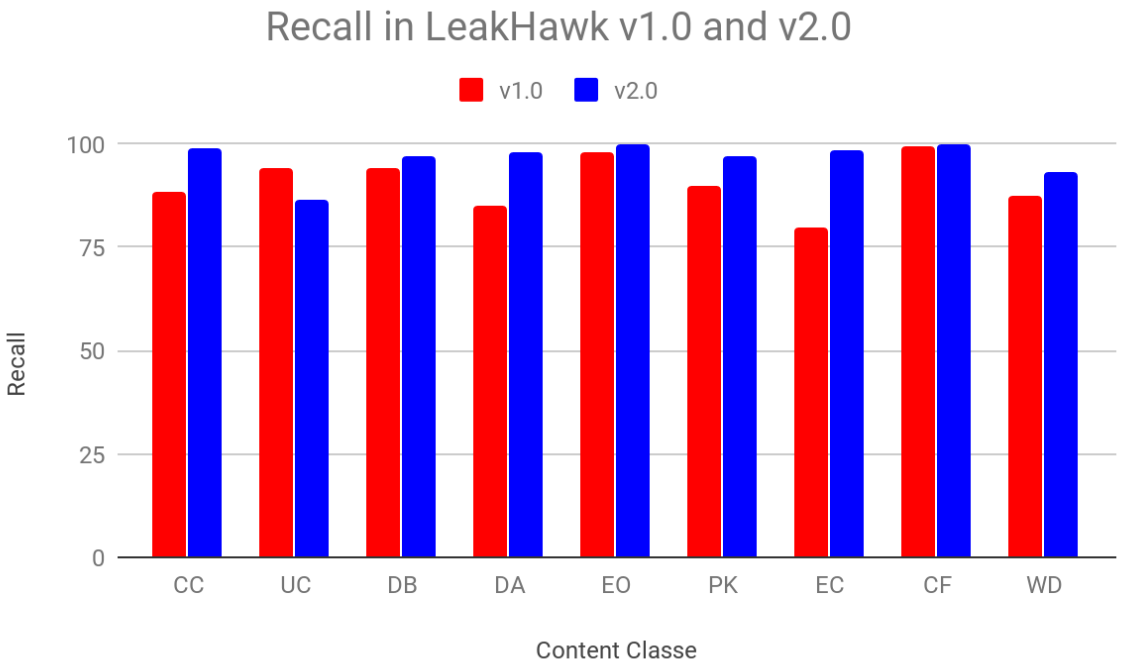


*Figure 5-12: Recall comparison of Content Classifiers*

According to Table 22 it is seen that the precision and recall of Pastebin Content classifier in LeakHawk 2.0 performs better than LeakHawk 1.0. There are some exception classes such as

DB, CF, and WD where precision is a little bit lower than LeakHawk 1.0, but the recall of that classes is improved than the earlier version. When considering the Content classifier as a whole it can be concluded that LeakHawk 2.0 analyses Pastebin posts better than LeakHawk 1.0.

## 5.3 Overall performance of the LeakHawk

This section illustrates the overall performance of LeakHawk flow considering all the components as a whole. Sets of posts containing both sensitive and irrelevant posts with varying post count are fed to the system and the time is analyzed for both Pastebin and Twitter.

### 5.3.1 End-to-End time to process Pastebin-Posts

This section shows the time taken to process different number of Pastebin posts. According to the results it is evident that the system can process Pastebin posts efficiently. Average Pastebin feed is about 24 posts per minute as analyzed by LeakHawk 1.0 author [6], but LeakHawk 1.0 could not process that much of load. LeakHawk 2.0 can process 100 Pastebin posts within 34 seconds (according to the average time in the Table 23) which is a greater improvement. This speed up is due to the integration of Apache Kafka for queuing and Apache Storm for processing. Spouts and bolts of Storm makes it easy to process incoming streams in real time and parallel.

*Table 23: Time takes to process Pastebin posts*

| **Number of posts** | 100 | 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 4000 |
|---|---|---|---|---|---|---|---|---|---|
| **Time to process (s)** | 34 | 138 | 269 | 399 | 531 | 672 | 796 | 935 | 1077 |

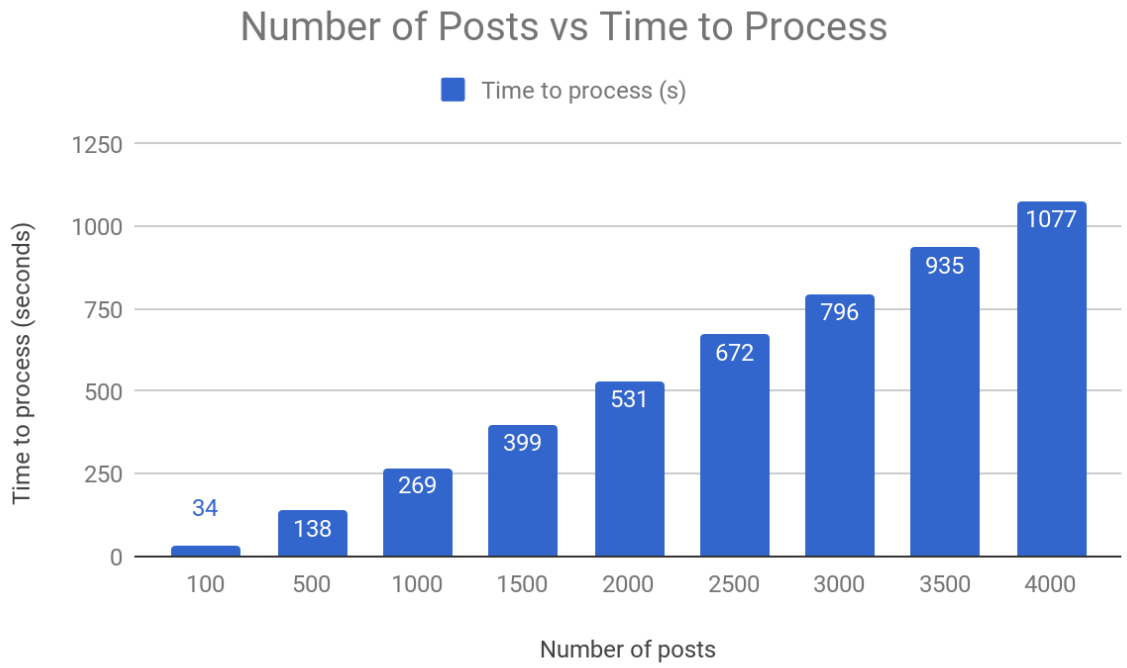


*Figure 5-13: Number of Pastebin posts vs Time to process*

### 5.3.2 End-to-End time to process Tweets

This section shows the time taken to process different number of tweets in the system. As seen in Figure 5-14 it can be concluded that the system can process tweets according to an average level. Average Twitter feed is about 6,000 posts per second and LeakHawk 2.0 can process 5,000 tweets within 2 sec. According to the results illustrated in Table 7-x, the system cannot process all tweets retrieved, using one machine. To improve the process capacity, at least two machines must be utilized.

*Table 24: Time takes to process tweets*

| **Number of tweets** | 5000 | 10000 | 20000 | 30000 | 40000 | 50000 | 60000 |
|---|---|---|---|---|---|---|---|
| **Time to process(s)** | 2 | 3 | 6 | 8 | 11 | 13 | 16 |

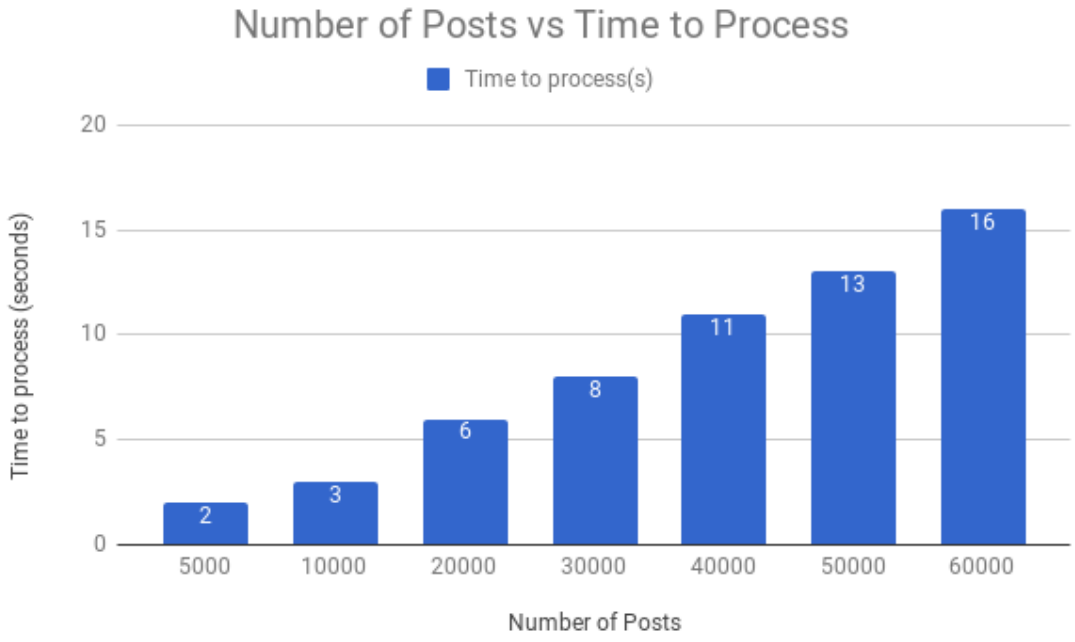


*Figure 5-14: Number of tweets vs Time to process*

### 5.3.3 Processor and Memory usage

This section illustrates the CPU and memory usage by the overall system when running in the environment described in section 5.1 for Pastebin and Twitter. Figure 5-15 shows the processor and memory usage when processing Pastebin feed provided as 1 post per second (Pastebin average feed is 24 posts per minute). When running for Pastebin, there is maximum CPU utilization as shown in Figure 5-15 and memory usage is approximately 70%. The reason for the utilization can be identified as, although Pastebin post feed is nearly 24 posts per minute, the size of a post is considerably large, and need more processing as well as memory power.

Figure 5-16 shows the processor and memory usage when processing Twitter feed provided as 6,000 tweets per second (Twitter average feed is 6000 posts per second). When running for Twitter, there is average CPU utilization as shown in Figure 5-16 and memory usage is approximately 70%. The reason for the utilization can be identified as, although Twitter post feed is nearly 6,000 tweets per second, as tweet size is 140 characters maximum, it needs only little amount of processing as well as memory power.

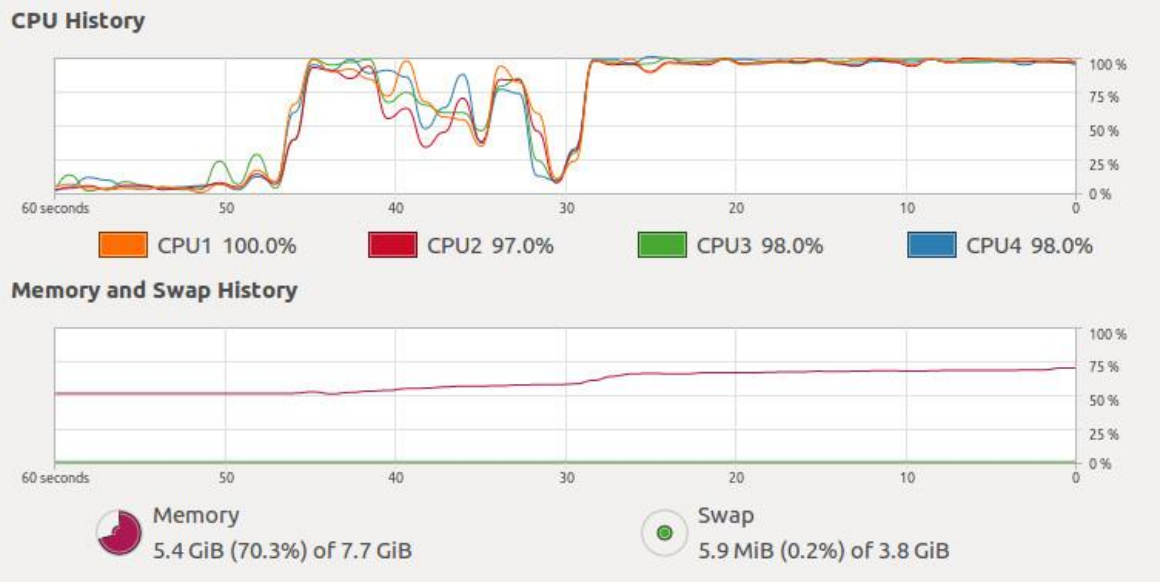


*Figure 5-15: Process and Memory usage to process Pastebin posts*

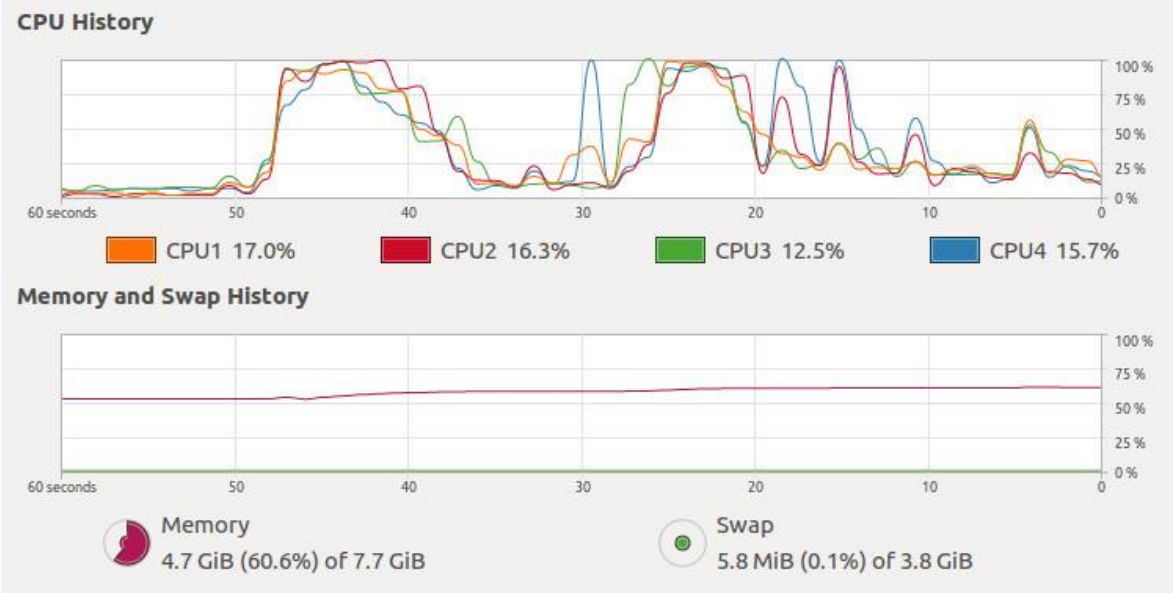


*Figure 5-16: Process and Memory usage to process Tweets*

In conclusion, LeakHawk 2.0 implementation provides a much better performance at the expense of accuracy, time, processing power and memory. It processes Pastebin posts faster and accurate than LeakHawk 1.0 implementation. The processor and memory utilization is satisfiable with regard to Pastebin feeds and with respect to Twitter feeds at least two machines should be used for better performance.

# 6 Summary

LeakHawk 2.0 is an open source contributed software application which is an extension of LeakHawk 1.0 which was a PoC implementation. LeakHawk is a real time, scalable automated framework that can detect data leakages and evidence of hacking attacks related to Sri Lankan domain that happened through text sharing sites (Pastebin) and social media sites (Twitter).

A main feature of LeakHawk 2.0 comparatively is the modularization of the system to make it easy for custom implementations. The level of abstraction of filters and classifiers has made this simpler. Moreover, LeakHawk 2.0 supports addition of multiple sensors where we connected Twitter apart from Pastebin. The custom implementations of filters and classifiers mainly uses Machine-Learning based and keyword-based methodologies for the implementation. This has reduced the occurrence of false negatives in the system by improving the recall and minimizes false positives. Automation of manual process of identifying data leakages and evidence of hacking attacks has made a challenging effort which is a very valuable approach for particular domains like financial domain, and data security domain. Moreover, the new platform itself is scalable which has made paths to integrate new data sources like Facebook, define new information domains and add new categories to categorize input data into relevant type. A valuable aspect to be considered is the system guarantees all the incoming posts to the system are processed without any data loss. Along with that the ability of the system to predict the sensitivity level of a given post is a significant feature that adds a value to the system.

Defining the information domain is a sort of manual process on the platform that contains unique attributes of a particular party. The scope and the complexity that the information model covers will affect the precise detection of data breach exposures.

LeakHawk 2.0 employs nine Machine Learning based classifiers to predict the sensitivity of the incoming posts where precision ranges from 81% to 100% with an average of 91% whereas recall ranges from 86% to 100% with an average of 96%. LeakHawk 1.0 incorporated a set of ten machine-learning based text classifiers for the severity classification with precision varying between 45%-95% with an average of 82% and recall ranging between 35%-98% with an average of 80%. According to the performance analysis results, it is evident that the system can process Pastebin posts efficiently. LeakHawk 2.0 can process 100 Pastebin posts

within 34 seconds which is a greater improvement. LeakHawk 2.0 implementation provides much better performance at the expense of accuracy, time, processing power and memory compared to LeakHawk 1.0.

## 6.1 Future Work

Functional and performance aspects of LeakHawk when dealing with social media feeds other than Twitter are not evaluated in this research. To generalize LeakHawk for other Pastebin applications, other than www.pastebin.com, several enhancements are necessary for the sensors. These changes are mostly needed due to the fact that Paste sites differ in the availability of an API, search functions, and access limitations.

Furthermore, the current dashboard can be designed to enhance the management and usability along with multiple alerting mechanisms. Dashboard can be further improved to allow users who uses the system to register in the system which needs to be validated by an admin. Apart from that dashboard can be improved to provide notifications to the registered users in a data breach exposure related to them. Also, a process can be integrated to automatically generate notifying emails in case of a data breach exposure. The performance of the LeakHawk can be significantly improved by integrating canary traps [41].

While overall accuracy of LeakHawk 2.0 is significantly better, there are some components that can be improved further. Precision of the Context classifier of LeakHawk 2.0 is 82.02% and that can be improved with the utilization of an optimized information template. Also in the Content classifier, classes DB, CF, and WD shows lower precision which need to enhance.

# 7 Bibliography


[1] "Commercial Bank of Ceylon Hacked? - BankInfoSecurity," 03 May 2016. [Online]. Available: https://www.bankinfosecurity.com/commercial-bank-ceylon-apparently-hacked-a-9103. [Accessed 12 October 2017].

[2] "—Pastebin.com - #1 paste tool since 2002!," [Online]. Available: http://pastebin.com/. [Accessed 25 March 2017].

[3] "—AnonymousSriLanka‘s Pastebin - Pastebin.com.," [Online]. Available: http://pastebin.com/u/AnonymousSriLanka.. [Accessed June 2017].

[4] "—Davyjones‘s Pastebin - Pastebin.com.," [Online]. Available: http://pastebin.com/u/davyjones.. [Accessed June 2017].

[5] 15 May 2016. [Online]. Available: http://www.asianmirror.lk/news/item/16544-commercial-bank-of-ceylon-hacked. [Accessed June 2017].

[6] N. Herath, "WEB INFORMATION EXTRACTION SYSTEM TO SENSE INFORMATION LEAKAGE," University of Moratuwa, Sri Lanka, 2016.

[7] "What is a Data Breach? - Definition from Techopedia," [Online]. Available: https://www.techopedia.com/definition/13601/data-breach. [Accessed October 2017].

[8] "Data Breach Definition | Investopedia," [Online]. Available: http://www.investopedia.com/terms/d/data-breach.asp. [Accessed October 2017].

[9] "How Harmful Can a Data Breach Be?," September 2015. [Online]. Available: http://resources.infosecinstitute.com/the-cost-of-a-data-breach-how-harmful-can-a-data-breach-be/#gref. [Accessed October 2017].

[10] "Hacking, Data Breaches & Cyber Warfare | IT Consulting & Technology," [Online]. Available: http://gcgcom.com/hacking-databreaches-cyber-warefare/. [Accessed June 2017].

[11] "The Use of Pastebin for Sharing Stolen Data," [Online]. Available: https://zeltser.com/pastebin-used-for-sharing-stolen-data/. [Accessed October 2017].

[12] "Trending Pastes at Pastebin.com," [Online]. Available: http://pastebin.com/trends.. [Accessed 2017].

[13] "Pastes Archive - Pastebin.com," [Online]. Available: http://pastebin.com/archive..

[14] "700,000 Dropbox credentials hacked, hacker leaks _Dropbox Hacks Teasers‘," [Online]. Available: http://www.techworm.net/2014/10/700000-dropbox-credentials-hacked-hacker-leaks-dropbox-hacks-teasers-pastebin.html.. [Accessed 2017].

[15] "M. R. O. 17 and 2014 Internet, —Facebook is taking a proactive approach to," [Online]. Available: http://www.techradar.com/news/internet/facebook-is-taking-a-proactiveapproach-to-fighting-password-leaks-1269726.. [Accessed 2017].

[16] "Dump Monitor (@dumpmon) | Twitter," [Online]. Available: https://twitter.com/dumpmon?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor. [Accessed 2017].

[17] "matthewdfuller/pastebin-find: Python script to monitor new Pastebin pastes for a provided search term," [Online]. Available: https://github.com/matthewdfuller/pastebin-find. [Accessed 2017].

[18] "Google Alerts - Monitor the Web for interesting new content," [Online]. Available: https://www.google.com/alerts.. [Accessed 2017].

[19] "xme/pastemon: pastebin.com Content Monitoring Tool.," [Online]. Available: https://github.com/xme/pastemon. [Accessed 2017].

[20] "LeakedIn," [Online]. Available: http://www.leakedin.com/.

[21] T. R. S. Muhammad Hussain Iqbal, "Big Data Analysis: Apache Storm Perspective," Faculty of Computing, SZABIST Dubai.

[22] "Apache HBase – Apache HBas," [Online]. Available: https://hbase.apache.org/. [Accessed 2017].

[23] "Welcome to Apache™ Hadoop®!," [Online]. Available: http://hadoop.apache.org/. [Accessed 2017].

[24] " Apache Spark™ - Lightning-Fast Cluster Computing," [Online]. Available: https://spark.apache.org/. [Accessed 2017].

[25] "S4: Distributed Stream Computing Platform from Yahoo! – MOA Massive Online Analysis," [Online]. Available: https://moa.cms.waikato.ac.nz/s4-distributed-stream-computing-platform-from-yahoo/. [Accessed 2017].

[26] "Facility detection and popularity assessment from text classification of social media and crowdsourced data," October 2016 . [Online]. Available: https://www.researchgate.net/publication/310360043_Facility_detection_and_popularity_assessment_from_text_classification_of_social_media_and_crowdsourced_data. [Accessed June 2017].

[27] [Online]. Available: http://140.123.102.14:8080/reportSys/file/paper/604410151/604410151_2_paper.pdf. [Accessed 2017].

[28] [Online]. Available: http://140.123.102.14:8080/reportSys/file/paper/604410151/604410151_2_paper.pdf.

[29] "Ontology-based Supervised Text Classification in a Big Data and Real Time Environment (PDF Download Available)," April 2016. [Online]. Available: [xx] (https://www.researchgate.net/publication/301199616_Ontology-based_Supervised_Text_Classification_in_a_Big_Data_and_Real_Time_Environment ). [Accessed 2017].

[30] "Early Detection of Spam Mobile Apps," [Online]. Available: http://spirit.cs.ucdavis.edu/pubs/conf/www15.pdf. [Accessed 2017].

[31] "Apache Kafka," [Online]. Available: https://kafka.apache.org/. [Accessed 2017].

[32] [Online]. Available: https://github.com/isuru-c/LeakHawk.

[33] [Online]. Available: https://pastebin.com/api_scraping.php.

[34] "Docs — Twitter Developers," [Online]. Available: https://developer.twitter.com/en/docs. [Accessed 2017].

[35] "Twitter4J - A Java library for the Twitter API," [Online]. Available: http://twitter4j.org/en/index.html. [Accessed 2017].

[36] "Binary Classification - Amazon Machine Learning," [Online]. Available: http://docs.aws.amazon.com/machine-learning/latest/dg/binary-classification.html. [Accessed 2017].

[37] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java," [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/documentation.html. [Accessed 2017].

[38] "Apache Tika – Apache Tika," [Online]. Available: https://tika.apache.org/. [Accessed 2017].

[39] "About WordNet - WordNet - About WordNet," [Online]. Available: https://wordnet.princeton.edu/. [Accessed 2017].

[40] "Thomson Reuters | Open Calais API," [Online]. Available: http://www.opencalais.com/opencalais-api/. [Accessed 2017].

[41] "Canary Trap Explained - Simplicable," [Online]. Available: https://arch.simplicable.com/arch/new/what-is-a-canary-trap.

[42] "Welcome to Apache™ Hadoop®!," [Online]. Available: http://hadoop.apache.org/. [Accessed 2017].