**Malicious Web Content Detection using Machine Learning**

Submitted in partial fulfillment of the requirements

Of the degree of

Bachelor of Engineering

By

Anand Desai

Roll No: 2013130010

Janvi Jatakia

Roll No: 2013130021

Rohit Naik

Roll No: 2013130034

Supervisor (s):

Prof. Nataasha Raul



Computer Engineering Department

Sardar Patel Institute of Technology

2016-17

**CERTIFICATE**

This is to certify that the project entitled Malicious Web content Detection using Machine Learning is a bonafide work of Anand Desai (2013130010), Janvi Jatakia (2013130021) and Rohit Naik (2013130034) submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering in Computer Science.

(Name and sign) (Name and sign)

Supervisor/Guide Co-Supervisor/Guide

(Name and sign) (Name and sign)

Head of Department Principal

**Project Report Approval for Bachelor of Engineering**

Project report entitled ***Malicious Web Content Detection using Machine Learning***by ***Anand Desai, Janvi Jatakia & Rohit Naik*** is approved for the degree of *Computer Engineering.*

Examiners

1. ---------------------------------------------

2. ---------------------------------------------

Date:

Place:

**Acknowledgement**

We would like to express our special thanks of gratitude to our teacher Prof. Nataasha Raul who gave us the opportunity to do this project. We are grateful to her for her guidance, invaluable inputs and consistent encouragement we received throughout the completion of this report. Ma’am has always made herself available to clarify our doubts despite her busy schedules and we are grateful for that.

We would like to acknowledge with much appreciation other faculty members of Computer Department, who gave us permission to use the labs to complete project related work, and helped us along the way.

Lastly, we would also like to thank our parents and friends who helped us a lot in finishing this project within the limited time.

Thank you.

**Table of Contents**

1. Abstract 1

2. Scope 2

3. Introduction 3

4. Literature Review 4

5. Hardware and Software Requirements 6

5.1 Hardware 6

5.2 Software 6

6. Diagrams 7

6.1 Use Case Diagram 7

6.2 Sequence Diagram 8

6.3 Activity Diagram 9

6.3.1 Activity Diagram For Training 9

6.3.2 Activity Diagram For Testing 9

7. Implementation Plan 10

7.1 Obtaining Dataset 10

7.2 Feature selection 10

7.3 Operations on the Dataset and Choosing the best algorithm

for classifying data 11

7.4 Feature Extraction Plans 12

8. Timeline for the project 14

9. References 15

**1. Abstract**

In today’s fast paced lifestyle it is very difficult to maintain multiple keys for various locks. With more number of keys, comes the possibility of misplacing a key. So this project aims at creating a smart lock which can be accessed through a mobile phone via an android app. Further the smart locks are vulnerable to various security attacks such as the Man in the Middle attack. Hence we will be providing with an added layer of security by implementing image steganography along with the encryption process. This will not only help us to access our locks through our mobile phones but will also provide more security than the other smart locks.

**2. Scope**

The project is aimed at making use of a Smart-Lock in place of the manual Lock and Key. We are also providing a better solution to Smart or Intelligent Lock system using a concept called Image Steganography. Steganography is the practice of hiding a secret message within an ordinary message, and subsequent extraction of the message at its destination. By using image steganography, the virtual key to such electronic locks can be shared and accessed using ordinary images. Our project aims at exploiting this aspect of IoT and hidden cryptography to provide an efficient and secure lock system.

**3. Introduction**

After the rise of e-commerce, social media and messenger bots, the next generation of internet belongs to connecting things, gadgets and devices : Internet of Things(IoT). These things range from sensors, security cameras to vehicles, household appliances and machines. The advantages in terms of convenience, efficiency and comfort make IoT a promising prospect. However, data confidentiality, privacy and security present challenges that are yet to be tackled successfully.  With technology ushering into spaces like security system, electronic door locks have managed to replace traditional door locks. Although Smart Lock system would make our lives easier, security experts have voiced their concerns on the potential security problems faced by these locks.

We make use of Bluetooth Low energy protocol to connect the Smart Lock with our mobile phones (Android App). Bluetooth Low Energy is designed to be power efficient and has been popular for transporting data between smartphones and IoT devices , smart homes, medical equipment and physical access control devices. One such vulnerability of BLE protocol is the Man in the Middle (MITM) attack.

Hence, the goal of this project is to provide a better solution for Smart Lock systems using concept called Image Steganography. Image Steganography helps to protect our system from the vulnerabilities of BLE protocol which will be used by our smart lock. By using image steganography, we aim at exploiting this aspect of IoT and hidden cryptography to provide an efficient and secure lock system.

**4. Literature Review**

According to the survey conducted in “Survey on Malicious Web Pages Detection Technique”, the authors mention that the web attacks are on a rise. There have been nearly 33,000 phishing attacks globally each month in the year 2012, totalling a loss of $687 million. With such a hike in the number of web attacks, there is a dire need of a system which prevents such types of webpage attacks [2]. The user goes to a website unknowingly which might not be safe for the user and ends up either giving his credentials or downloading malicious data to other intruders and hackers. So to prevent this type of attacks, a tool is needed which examines the URL entered by the user and checks if the website is malicious or not. In the paper “Identifying Vulnerable Websites by Analysis of Common Strings in Phishing URLs “, the authors mention about the rise in the phishing websites and method they used to detect and prevent it. They implemented Largest Common Substring method to identify if the webpage is phishing or no. They prepared a database of phishing websites of their own and found the LCS to see if the new web-page is phishing or not [3]. In the paper “On URL Classification”, the authors mention about how URLs can be used to use the victim's computer resources for different attacks like phishing, denial of service. Also a comparison between machine learning and non-machine learning approaches is done. The results show that machine learning techniques are better for detection [5]. So instead of using the traditional techniques, we plan to integrate the concept of machine learning in our tool.

Now, as we have to examine the URL for deciding whether it is a malicious website or benign, we will have to extract the features of the website. In the paper “Feature Extraction Process: A Phishing Detection Approach “, the author talks about how the features can be extracted from a URL. The author finds 17 features which can be extracted from the URL based on which a URL can be declared as phishing or no [6]. In our project, we plan to extract 24 features which will enable more accuracy in declaring a URL as malicious or benign. In the paper "Feature extraction and classification phishing websites based on URL," M. Aydin and N. Baykal used the feature extraction technique to form a feature matrix using which they classified the URL. They extracted 133 features and used only a subset of it which they considered as prominent. They have not specified the reason for choosing the parameters using which they declare a website as malicious or no. They used different parameters and different algorithms to just test the efficiency obtained [7]. We plan to select a single algorithm and use all the parameters and features as given in the dataset we selected. In “A Comparison of Machine Learning Techniques for Phishing Detection”, Abu-Nimeh et al have done a comparative study of six different classifiers to find which classifier works the best. They have showed that Random Forest out performs other classifiers by having the lowest error rate [8]. Instead of using a single classification algorithm, we will use a hybrid a two algorithms to provide better efficiency of classification.

In the paper “James, Joby, L. Sandhya, and Ciza Thomas. "Detection of phishing URLs using machine learning techniques", James et al discuss about the rise of phishing websites and give techniques to extract features and implement machine learning algorithms to classify the same. They have extracted features like traffic rank details, lexical features, page rank etc. They have presented a study of different machine learning algorithms. A fixed result showing the best algorithm is not done in the paper, we will give statistical analysis of all the algorithms and even the value of the hybrid algorithm to prove the result [9].

As the technology advances, the number of possible malicious attacks would also increase. It would be impossible to prevent all the new malicious and phishing websites using the traditional methodology of storing the list of malicious URLs and checking directly from the database. So “Malicious Web Content Detection by Machine learning” aims at improvising the traditional methodology by adding machine learning for the same. The paper uses four different classification algorithms to detect Dynamic HTML malicious codes and states that Boosted Decision Tree gives the best output. The prototype they made is resilient to code obfuscations and can determine whether webpage is malicious or not, but cannot block or prevent the malicious content [10]. Now, there can be an instance that the URL might be benign but it may contain certain JavaScript code or iframe which will lead to a download of malicious content. So taking this possibility in consideration we will scrape the entire webpage to detect malicious contents even if the website is rendered safe by the algorithm.

**5. Hardware and software requirements**

**5.1 Hardware Requirements:**

* Laptops or Personal Computers (PC) will be needed with the users. Mobile phones will not do as there is no support for Chrome extensions on mobile phone devices.

**5.2 Software Requirements:**

* Google Chrome should be installed on the user’s laptop or PC and the extension should be added to their Chrome browser. Also, the extension should not be turned off at any point, to ensure continuous support.
* Python needs to be installed on server PC to handle requests from users. Whenever a user accesses a website with the extension turned on, the URL will be sent to the server, features will be extracted and testing will be done with the existing dataset. If the current URL resembles exactly one of the URLs already present in the dataset, then the URL won’t be stored for future training of the classifier. On the other hand, if it is different, then the extracted features will be added to the dataset with the result and after a duration of approximately 7 days, the machine will be trained again with the new URLs that have been collected till that time.

**6. Malicious Web Content Detection Design Diagrams**

**6.1 Use-Case Diagram**

Use-Case-Diagram.png

Fig 1. Use Case diagram for ‘Machine Learning based approach for malicious web page detection’

**6.2 Sequence Diagram**

Sequence.png

Fig 2. Sequence Diagram for ‘Machine Learning based approach for malicious web page detection’

**6.3 Activity Diagram**

**6.3.1 Activity Diagram for Training Part**

Activity-Diagram-1.png

Fig 3. Activity Diagram for training part of ‘Machine Learning based approach for malicious web page detection’

**6.3.2 Activity Diagram for Testing Part**

Fig 4. Activity diagram for testing part of ‘Machine Learning based approach for malicious web page detection’Activity-Diagram-2.png

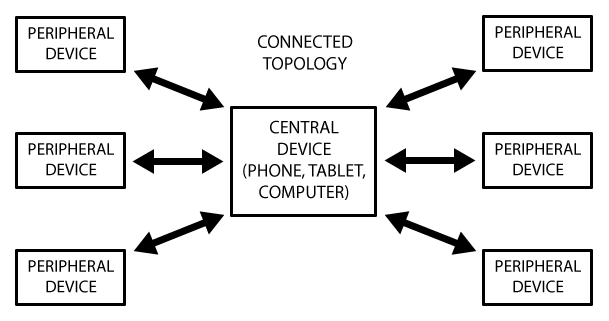
**7. Implementation**

**7.1 Detection of BLE devices through mobile phone:**

The app we created shows a list of available Bluetooth LE devices and provides an interface to connect and display data. It creates a Service for managing connection and data communication with a GATT server hosted on a given Bluetooth LE device.

GATT is an acronym for the Generic Attribute Profile, and it defines the way that two Bluetooth Low Energy devices transfer data back and forth using concepts called Services and Characteristics. It makes use of a generic data protocol called the Attribute Protocol (ATT), which is used to store Services, Characteristics and related data in a simple lookup table using 16-bit IDs for each entry in the table

The most important thing to keep in mind with GATT and connections is that connections are exclusive. What is meant by that is that a BLE peripheral can only be connected to one central device (a mobile phone, etc.) at a time! As soon as a peripheral connects to a central device, it will stop advertising itself and other devices will no longer be able to see it or connect to it until the existing connection is broken.

. 

The peripheral is known as the GATT Server, which holds the ATT lookup data and service and characteristic definitions, and the GATT Client (the phone/tablet), which sends requests to this server. All transactions are started by the master device, the GATT Client, which receives response from the slave device, the GATT Server.

Screenshot of working app:

* 1. **Simulation of Raspberry Pi:**

As we await the possession of our hardware requirements, simulation of Raspberry Pi can be done for initial implementation. QEMU (Quick Emulator) is a tool which can be used for simulation of Raspberry Pi. QEMU is a free and open-source hosted hypervisor that performs hardware virtualization and emulates CPUs through dynamic binary translation. QEMU provides a set of device models, enabling it to run a variety of unmodified guest operating systems (Raspbian OS in our case).

QEMU is used purely for CPU emulation for user-level processes, allowing applications compiled for one architecture to be run on another.

**Comparison of QEMU with Contiki- Cooja**

For a comprehensive overview of available simulators, we compared QEMU with another simulator, Contiki-Cooja. Contiki is an operating system for networked, memory-constrained systems with a focus on low-power wireless Internet of Things devices.

Cooja - Contiki Network simulator allows simulation of Contiki motes (hardware emulations). A simulated Contiki Mote in COOJA is an actual compiled and executing Contiki system.

However, we found a few drawbacks of the Cooja simulator which could be overcome by using QEMU instead. Here are the drawbacks we found:

1. Contiki OS is not for full-blown computer systems like the Raspberry Pi
2. Cooja simulator for network sensors does not have a Raspberry Pi ‘mote’ (sensor node)

Whereas, the following are the advantages of using QEMU over Cooja

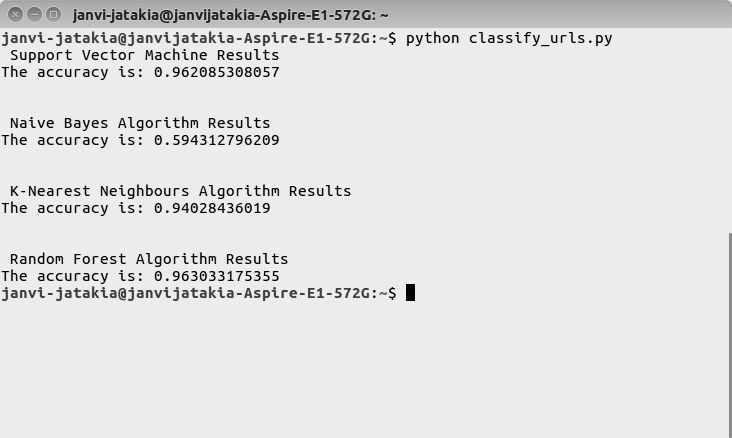
1. QEMU is a complete and standalone software on it's own
2. Used to emulate machines, it's very flexible and portable
3. Supports emulation for raspberry Pi

**7.3 Operations on the Dataset and Choosing the best algorithm for classifying data:**

Reading and understanding different machine learning algorithms, four of the algorithms were shortlisted. The following are the algorithms:

1. Support Vector Machines (SVM)
2. Naive Bayes Algorithm
3. K- Nearest Neighbours (k-NN) Algorithm
4. Random Forests Algorithm

The dataset obtained from the repository was first converted into the form of arrays which are necessary for training the data. This was done using basic python functions. Now, this dataset is trained using the above algorithms in python. Of the 11055 entries, we kept 1055 entries for testing the trained classifier formed from 10000 entries. The accuracy of the 4 classifiers were calculated and we found the following output:

Fig 5. Accuracy when classifier is tested after extracting all the 30 features.

Now, the dataset was changed by removing the columns corresponding to the features we didn’t consider. Again using the basic python functions this was done. Similar procedure as earlier was followed on this new dataset and the following results were obtained:

Fig 6. Accuracy when classifier is tested after extracting the 24 decided features.

Analysing the above results, we can see that two of the algorithms have over 95% accuracy. So, we plan to study more on Support Vector Machines(SVM) and Random Forest Algorithm for better efficiency of the system. Also, it was noted that even after removing some of the features from the dataset, the accuracy was found to change just by 1%. Hence, it is proved that the features we removed were not contributing more to the result. So, it is safe to remove those features and keeping the results same as it was earlier.

**7.4 Feature Extraction Plans:**

We have to extract the features of any new website provided to the system. In the dataset we considered, there are different types of features. The documentation of the dataset gives a clear idea about the features used. For extracting such features, we have to use different methods. The future plan for extracting these features is as follows:

1. Address Bar Based Features:

These features include the features which can be easily obtained from the URL of the webpage. So for getting these features, rules have been defined feature wise. We will be implementing these rules in the python language. This code developed will be used for getting some of the features from the given website.

1. Abnormal Based Features:

These features are the ones which include checking the request urls of objects in web pages or some functions used in the script which directs to some site or mails to some personal account. For extracting such features, we plan to use scraping techniques which gets us the source code of the page and we can check for such features.

1. HTML and JavaScript Based Features:

These features includes the elements of HTML which can be used for attacking as well as some malicious javascript functions. These features can be obtained using a similar way as the above one.

1. Domain Based Features:

These features are the ones which find the popularity of the websites and how many times it is being visited using the domain name. These can be extracted from readily available data from Google, etc. by writing a python code for the same.

In this way we plan to extract all the features and used the trained classifier from the above step to detect if the site is a phishing site or no.

**8. Timeline**

July-September:

* Finalization of Topic.
* Research about the topic.
* Identification of the Hardware and Software requirements.

October-January:

* Finding the possible solution.
* Simulation of Raspberry Pi.
* Detection of BLE devices through mobile phone.

January-March:

* Evaluation and Refining the solution
* Setup the Raspberry Pi.
* Connection of Raspberry Pi with the lock
* Finalizing the Solution
* Presenting the paper.

**9. References**

[1] http://www.w3schools.com/browsers/ Month-by-month Browser Usage Statistics

[2] D. R. Patil, J. B. Patil, “Survey on Malicious Web Pages Detection Techniques,” *Science and Technology, 2015 International Journal of u- and e- Service.*

[3] B. Wardman, G. Shukla and G. Warner, "Identifying vulnerable websites by analysis of common strings in phishing URLs," *2009 eCrime Researchers Summit*, Tacoma, WA, 2009, pp. 1-13.

[4] A. B. Sayambar, A. M. Dixit, “On URL Classification,” *International Journal of Computer Trends and Technology 2014.*

[5] Xiang et al., “A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites,” *ACM Transactions on Information and System Security 2011.*

[6] Abunadi, Ahmad, Oluwatobi Akanbi, and Anazida Zainal. "Feature extraction process: A phishing detection approach." 2013 13th International Conference on Intellient Systems Design and Applications. IEEE, 2013.

[7] M. Aydin and N. Baykal, "Feature extraction and classification phishing websites based on URL," *Communications and Network Security (CNS), 2015 IEEE Conference on*, Florence, 2015, pp. 769-770.

[8] Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A Comparison of Machine Learning Techniques for Phishing Detection. APWG eCrimes Researchers Summit, (pp. 60-69). Pittsburgh, PA.

[9] James, Joby, L. Sandhya, and Ciza Thomas. "Detection of phishing URLs using machine learning techniques." Control Communication and Computing (ICCC), 2013 International Conference on. IEEE, 2013.

[10] Hou et al., “Malicious Web Content Detection by Machine learning,” *Expert Systems with Applications, International Journal 2010.*

[11] G. Venkataraman and A. Ravichandran, "Adaptive Semantic Search: Re-Ranking of Search Results Based on Webpage Feature Extraction and Implicitly Learned Knowledge of User Interests," *Semantics, Knowledge and Grids (SKG), 2014 10th International Conference on*, Beijing, 2014, pp. 75-78.

[12] Khamis et al, “Characterizing A Malicious Web Page,” *Australian Journal of Basic and Applied Sciences 2014.*

[13] Curtsinger et al., “Zozzle: Fast and Precise In-Browser JavaScript Malware Detection,” *SEC'11 Proceedings of the 20th USENIX conference on Security 2011.*

[14] G. Lu and S. Debray, "Automatic Simplification of Obfuscated JavaScript Code: A Semantics-Based Approach," *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference*, Gaithersburg, MD, 2012, pp. 31-40.

[15] M. Aldwairi, R. Alsalman “MALURLS: A Lightweight Malicious Website Classification based on URL features,” *Web Intelligence, 2012 Journal of Emerging Technologies.*

[16] Eshete et al., “Malicious Website Detection: Effectiveness and Efficiency Issues,” *SysSec Workshop (SysSec) 2011.*

[17] http://archive.ics.uci.edu/ml/datasets/Phishing+Websites UCI Phishing Websites Data Set

[18] Ma et al., “Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs,” *Knowledge discovery and data mining, 2009 15th ACM SIGKDD international conference.*

[19] Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Xiaotie Deng and Zhang Min, "Phishing Web page detection," *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, 2005, pp. 560-564 Vol. 2.

[20] Canali et al., “Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages,” *Web Security, 2011 WWW 2011.*

[21] Cova et al., “Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code,” *WWW '10 Proceedings of the 19th international conference on World wide web, 2010.*

[22] Ma et al., “Identifying Suspicious URLs: An Application of Large-Scale Online Learning,” *Machine Learning, 2009 ICML '09 Proceedings of the 26th Annual International Conference.*

[23] Y. Huang and L. Li, "Naive Bayes classification algorithm based on small sample set," 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, Beijing, 2011, pp. 34-39.

[24] Zhou et al., “Malicious Websites Detection and Search Engine Protection,” *Journal of Advances in Computer Network 2013.*

[25] Choi et al., “Detecting Malicious Web Links and Identifying their attack types,” *Web application development*, 2011 *2nd USENIX conference.*