Dynamic pattern based virus Identification using dynamic program invocation pattern model

Peram Manoj Kumar[1], M. Dileep Kumar Reddy[2],S.Sai Surya Narayana Raju[3], Prem Priya[4]
[1,2,3,4] *M.Tech.,(Ph.D) Assistant Professor ,Department of CSE,R.M.K College of Engineering and Technology*

***Abstract*- Research focus on the dynamic invocation of code and patterns based on the existing antivirus model.This project emphasizes an artificial intelligence model in developing the code which will do the following process. Invoke automatically and started executing in case of any virus files available The virus files can be identified through the signatures Can be identified via the type / name of files which already exists. In case, the user will train the model for the type of virus. An intelligent system will take care of future detection with those patterns. The operations involve the 3 step process Validating the behavior of the files An automatic event detection happens in case of any actions needs to be done. Event will be handled to process in case of virus detection. The scope of this project is to prevent the customized viruses which can be analyzed via the name and also the signature of the virus.**

1. INTRODUCTION

These days one cannot open their email without seeing countless spam messages in their inbox. For the email-recipient, spam is easily recognized. However, the receiver of spam loses countless hours manually deleting the intrusive messages from their inbox. Spam filter mechanism can help mitigate this overwhelming chore. Spam filter mechanism can reduce the amount of junk mail delivered to a user's inbox.

The problem of spam or Unsolicited Bulk Email (UBE) is becoming a pressing issue. In spite of the development of many anti-spam techniques, the war against spam is far from being successful. This is partly due to several characteristics of spam that make it a difficult problem. E-MAIL communication is prevalent and indispensable nowadays. However, the threat of unsolicited junk emails, also known as spams, becomes more and more serious. The primary challenge of spam detection problem lies in the fact that spammers will always find new ways to attack spam filters owing to the economic benefits of sending spams. Note that existing filters generally perform well when dealing with clumsy spams, which have duplicate content with suspicious keywords or are sent from an identical notorious server. Therefore, the next stage of spam detection research should focus on coping with cunning spams which evolve naturally and continuously. we present an open digest technique, that we have adapted to take into account disguising attacks and illustrate its resiliency

Previous researchers have developed various methods on near-duplicate spam detection; these works are still subject to some drawbacks. To achieve the objectives of small storage size and efficient matching, prior works mainly represent each e-mail by a succinct abstraction derived from e-mail content text. Moreover, hash-based text representation is applied extensively. One major problem of these abstractions is that they may be too brief and thus may not be robust enough to withstand intentional attacks.

In this paper, we explore to devise a more sophisticated email abstraction, which can more effectively capture the near duplicate phenomenon of spams. Motivated by the fact that email users are capable of easily recognizing similar spams by observing the layouts of e-mails, we attempt to represent each e-mail based on the e-mail layout structure. Fortunately, almost all e-mails nowadays are in Multipurpose Internet Mail Extensions (MIME) format with the text/html content type. That is, HTML content is available in an e-mail and provides sufficient information about e-mail layout structure. In view of this observation, we propose the specific procedure Structure Abstraction Generation (SAG), which generates an HTML tag sequence to represent each e-mail. Different from previous works,

SAG focuses on the e-mail layout structure instead of detailed content text. In this regard, each paragraph of text without any HTMLtag embedded will be transformed do a newly defined tag <my text=>.
In the field of collaborative spam filtering by near-duplicate detection, a superior e-mail abstraction scheme is required to more certainly catch the evolving nature of spams. Compared to the existing methods in prior research, in this paper, we explore a more sophisticated and robust e-mail abstraction scheme, which considers e-mail layout structure to represent e-mails.

2. EXISTING SYSTEM

Existing System focused on the event driven actions.
The three main diagnostic phases that emerges are

Phase 1: Behavior modeling.
Phase 2: Event detection.
Phase 3: Event handling.

Existing System provides an automated process in handling the viruses such as,
Detect the virus signature and delete it.
Detect the standard formats of the viruses such as exe and vbs files.
Disadvantages:
No proper reporting of the viruses happening in the system
Custom signature viruses cannot be detected.
Custom formats were not identified.

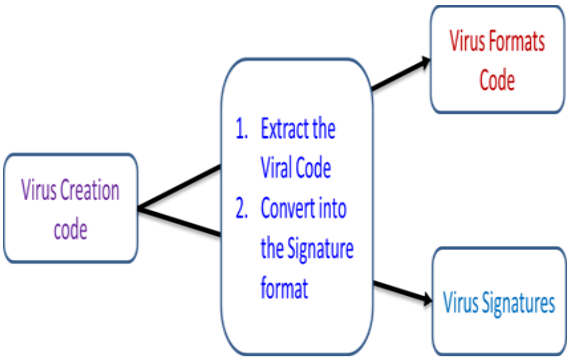
3. PROPOSED SYSTEM

Proposed system mainly focusses on the Artificial Intelligence system.In this project, Custom file format files will be deleted based on the user feedback provided. Custom virus signature can be added and it will be evaluated on the files and based on the outcome the files will be deleted.Automated email system in case of file detection.
Advantages:
Proper reporting of viruses were done in the proposed format through emails
Custom signature viruses can be detected. User feed signatures were considered in detecting the viruses.
Custom formats will be added and the actions were marked for future detections and actions.

4.MODULE DESCRIPTION

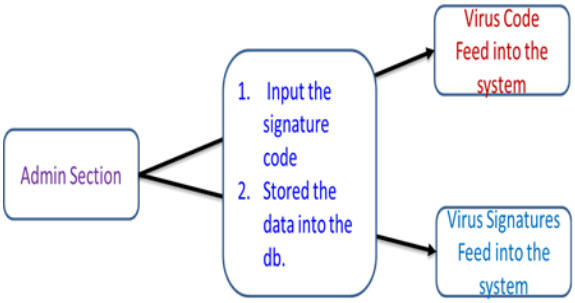
4.1 Sample Virus Creation

In this module, we are trying to create the possible sample viruses.Virus indicates, it wont exist in its format. In turn, it will exists in the format of subordinating with the existing code – Exists as the subroutine program. This module will make us to create the virus in its own format of signatures, The signature can be,
An ASCII format Can be a AlphaNumeric format,An encrypted formatted string,This module enables you to create the viruses along with the signature format. The signaturized formats will be left off from the source to the destination which is really a challenging process for the recent IT trends. Let's see how to overcome the same.



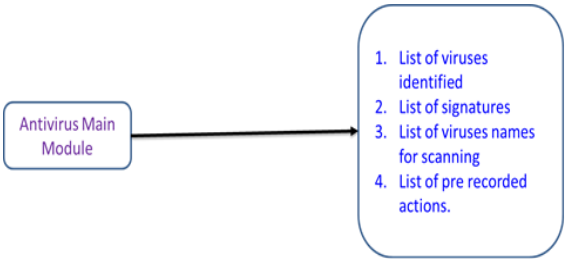
4.2 Virus Pattern Define Module – Admin Section

In this module, we are trying to create the pattern for the virus formats and functionalities.The patternized virus codes were feed as signature for future reference.
Future references indicates,
Creating the signature code for the input code.Feed it into the system.Looping through the searching methodologies to search for the possible viruses in the input file injected into the system.This module utilizes the Signature match algorithm in matching the input format with the stored formats



4.3 Antivirus Main Module

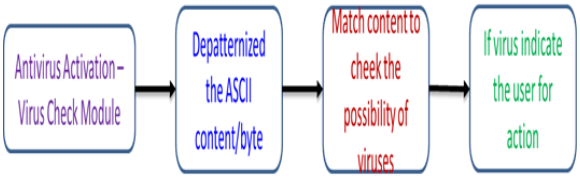
This module indicates the following sections,List of viruses identified with the dates and everything.List of signatures added.List of names of the viruses added In case of executable files, the only option is to scan through the names.List of pre-recorded actions taken on the signature and viruses.



4.4 Antivirus Activation – Virus Check Module

This module matches the virus signature and the input pattern.It involves,

De-patternized the ASCII content / bytes in to the original data.Match content to cheek the possibility of viruses.If virus indicate the user for action



4.5 Dynamic Pattern Feed/Name Feed Module

In this module, the name and the pattern of the viruses will be fed dynamically. This module provides the features of,Editing the name of the viruses, If wrongly feed into the system. Editing the patterns of the virus formats.View all the virus patterns Edit the actions taken on the files and viruses

4.6 Virus Alert Module / Virus Action Module

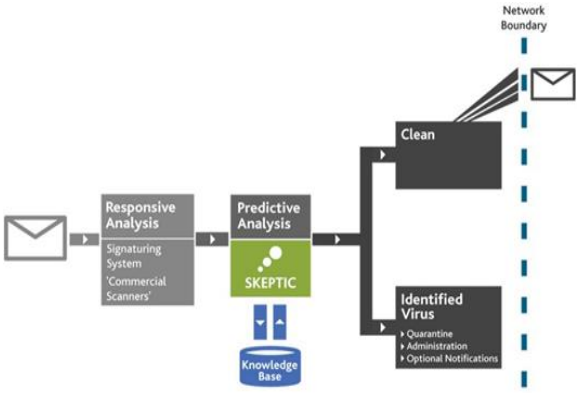
Final module includes the type of actions taken on the viruses.

User will be given an option to finalize the future actions to be taken on the viruses. In case, if it comes from different sources.

The actions specified will be reconfirmed to the user before taking an action.

The viruses will be moved to a very safe location in the machine. In case, it is needed for future references.

5. ARCHITECTURE DIAGRAM



6 SYSTEM TESTING

6.1 TESTING OBJECTIVES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTS

6.2.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually

run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is entered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.3 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.3.1 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.3.2 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

6.4 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Features to be tested

Verify that the entries are of the correct format

No duplicate entries should be allowed

All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.
Test Results: All the test cases mentioned above passed successfully. No defects encountered

7 LITERATURE REVIEW

TITLE: Automatic Inference and Enforcement of Kernel Data Structure Invariants
We introduce a narrative method for Fast SVM Training detection, called Fast SVM Training .
It uses a Bayesian network to resolve the chance of two Fast SVM Training elements being duplicates, making an allowance for not only the information within the elements, except the way that information is ordered.
A novel pruning strategy, capable of significant gains over the un-optimized version of the algorithm, is used to improve the efficiency of the network evaluation.
Here we are able to do better than another state-of-the-art duplicate detection solution, equally in conditions of efficiency and of effectiveness.

MERITS:
Using our algorithm a high precision and recall scores in several data sets is achieved.
We propose a lossless pruning strategy, improve the BN evaluation time.

DEMERITS:
Fast SVM Training needs petite user intervention, in view of the fact that the user only needs to offer the attributes to be considered, their individual default probability parameter, and a similarity threshold.
We aim to enlarge the BN model construction algorithm to balance Fast SVM Training L objects with diverse structures.

TITLE: Control-Flow Integrity Principles, Implementations, and Application
Here we are majorly focusing on computation acceleration Sequential Minimal Optimization.
We present a revision of Sequential Minimal Optimization and decide that memory-side data loading in the parsing phase incurs a major performance overhead, as much as the computation does.
We suggest memory-side acceleration which incorporates of data prefetching techniques, and can be practical on top of computation-side quickening to speed up the Sequential Minimal Optimization data parsing.
We put into practice a prefetcher on an display place in an effort to appraise its execution feasibility in conditions of area and energy overhead.

Merits:
Memory-side accelerators carry substantial effectiveness athwart existing parsing models.
Applying a two-layer prefetcher may guide to up to 4 percent additional energy consumption.

Demerits:
Sequential Minimal Optimization performance is hurt by the latency, due to bandwidth.
Only 20 percent of performance is improved.

TITLE: Detecting kernel-level rootkits Through Binary Analysis
In this paper, we take a unusual approach—deploying interoperable Simple Object Access Protocol (SOAP)-based web services straight on the nodes and not by means of gateways.
It enables standard based and straight application-layer integration flanked by web service- enabled IT systems and resource-constrained sensor nodes.
We establish a novel pruning strategy, competent of significant gains over the un-optimized version of the algorithm, is used to perk up the efficiency of the network evaluation.
Here we are able to do better than another state-of-the-art duplicate detection solution, equally in conditions of efficiency and of effectiveness.

Merits:
We combine lightweight TCP/IP stack implementation and SOAP-based web service implementation.

It provides for simple integration with legacy IT systems and chains heterogeneity at the lowest level.

Demerits:
The transparency related to SOAP message processing is very small compared to message transmission.
Dissimilar ways to lower the related visual projection should be investigated.

TITLE: A TAXONOMY OF EMAIL SPAM FILTERS
This paper proposed a taxonomy of existing SPAM email countermeasures, and a brief description of the taxa we have proposed, and ascribe a number of existing SPAM filters to the various taxa. We have presented a wide range of the techniques that have been used or proposed for use to fight SPAM, and attempted to indicate which SPAM filters use which techniques. These filters make assessments about the reputation of one or more of the participants (sender, recipient and intermediaries) in the email transaction.

MERITS:
Based upon the type of spam we are going to use the technique to filter the spam.
The taxonomy presented here is clearly preliminary in nature, and non-exhaustive.

DEMERITS:
There is no information about additional SPAM filters and techniques, and to address any refinements that become apparent during that process

TITLE: ON EXTENDABLE SOFTWARE ARCHITECTURE FOR SPAM EMAIL FILTERING
In this paper, we propose to use dynamic multi normalizers as the preprocessors for spam filters. The normalizers convert an email to its plain text format. The most important feature of this paper is architecture is the flexibility in easily adapting techniques to fight new spamming inventions. Normalizers can be easily introduced into or removed from the system.

MERITS
- This type of filtering is flexible to adapt the new development of spam techniques, such as HTML tagging, image based spam, and keyword obfuscating etc.

DEMERITS
- The obfuscated images, although still humanly readable, won't bring the expected return to the spammers, as human readers are less likely to respond to this type of images.

TITLE:A NOVEL METHOD OF SPAM MAIL DETECTION USING TEXT BASED CLUSTERING APPROACH
In this paper, we can extract spam/non-spam email and detect the spam email efficiently. Representation of data is done using a vector space model. Clustering is the technique used for data reduction.

MERITS
In this paper an email clustering method is proposed and implemented to efficient detect the spam mails.
The proposed technique includes the distance between all of the attributes of an email.

DEMERITS
- In paper the BIRCH clustering, decisions made without scanning the whole data & BIRCH utilizes local information (each clustering decision is made without scanning all data points).

BIRCH is a better clustering algorithm requiring a single scan of the entire data set thus saving time. So, it cannot work in any other algorithm.

TITLE: A LEARNING APPROACH TO SPAM DETECTION BASED ON SOCIAL NETWORKS
This paper consists different approach to spam detection is based on the behavior of email senders. we propose a learning approach to spam sender detection based on features extracted from social networks constructed from email exchange logs. Legitimacy scores are assigned to senders based on their likelihood of being a legitimate sender.

MERITS
- This system have various spam filtering and resisting possibilities.
- The proposed technique does not require the content of the mail.

DEMERITS

The system have to be cautious about the actual performance of the proposed scheme considering that we are scoring senders instead of individual emails.

REFERENCES

[1] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features ofescape panic," Nature, vol. 407, pp. 487–490, Aug. 2000.

[2] R. M. Bond et al., "A 61-million-person experiment in social influenceand political mobilization," Nature, vol. 13, no. 489, pp. 295–298, 2012.

[3] E. Cator and P. Van Mieghem, "Susceptible-infected-susceptible epi-demics on the complete graph and the star graph: Exact analysis," Phys.Rev. E, vol. 87, no. 1, 2013, Art. no. 012811.

[4] F. Bonchi, C. Castillo, A. Gionis, and A. Jaimes, "Social network analysis and mining for business applications," ACM Trans. Intell. Syst.Technol., vol. 2, no. 3, Apr. 2011, Art. no. 2