Forensic Analysis and Detection of Spoofing Based Email Attack Using Memory Forensics and Machine Learning

Sanjeev Shukla[1(✉)], Manoj Misra[1], and Gaurav Varshney[2]

[1] Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee, India
{sanjufcc,manojfec}@iitr.ac.in

[2] Department of Computer Science and Engineering, Indian Institute of Technology, Jammu, India
gaurav.varshney@iitjammu.ac.in

Abstract. Emails encounter many types of cyber-attacks and email spoofing is one of the most common and challenging investigation problems. This paper identifies spoofing-based email attacks in an organization by analyzing received and replied emails. The detection works by capturing the email traces via memory forensics. Unlike the traditional approaches of capturing the entire physical memory, we only capture the memory of relevant processes for email header extraction. It significantly reduces the size of the memory dump and makes detection faster. We suggest a novel mechanism called URL extractor, which uses seven novel features from URL to identify the live running email message process by applying ML that traces received emails and captures their header fields for analysis. The authentication header fields of *SPF, DKIM, DMARC,* and *ARC* are examined closely to develop a detection algorithm for received emails. Similarly, novel header fields of *Reference* along with *MX record* are applied for the detection of replied emails. The MX record is fetched to verify the domain name by sending a forward ns-lookup query to DNS. It also includes an email attack alert mechanism for intimating IT admins of an organization regarding suspected attacks. The results thus obtained show that email detection takes 35 secs (apprx.) to complete with high accuracy and low false positives.

Keywords: Email forensics · Email spoofing · Memory forensics · Cyber security · Email attacks

1 Introduction

In the modern era of an IT-enabled society, consumers increasingly use digital communication as a preferred mode of interaction for their business and personal needs. Email is still a reliable, safe, and most widely used mode of data communication on the Internet. As email services have gained popularity, attackers are

F. Li et al. (Eds.): SecureComm 2022, LNICST 462, pp. 491–509, 2023.
https://doi.org/10.1007/978-3-031-25538-0_26

using email as a platform to launch cyber attacks [1]. *Email spoofing* is one of the prevalent attacks on the email platform. Most Business Email Compromises (BEC) and phishing attacks use email spoofing as the first step. In email spoofing, the attacker finds a way to deceive the receiver into trusting a false identity of the sender, eventually gaining a higher level of trust. Email spoofing is generally performed by finding out an open relay or open *simple mail transfer protocol (SMTP)* server which can relay the emails without any sender authentication.

1.1 Motivation

CSO Online article "Top cybersecurity facts, figures and statistics" and a survey by International Data Group (IDG) in 2020 report that email application is still mostly preferred for malicious propagation as found in 94% of cases. It also states that phishing accounts for 80% of reported security incidents [2]. It brings a unique challenge for organizations as it caters to both external and internal attack threats. The inbound emails (received emails) attacks pose a challenge of early detection so that necessary mitigation steps can be applied. Also, a malicious insider can use the organization's precious resources to launch an email spoofing attack. This outbound emails (sent emails) attack is more difficult to detect and needs urgent attention as it blacklists the IP addresses of the organization and brings a bad reputation. Therefore, the motivation of this work is to provide a mechanism for detection and early warning for admins and security experts to address email spoofing attacks precisely and timely.

1.2 Email Forensics

Email forensics is a sub-branch of network forensics that performs forensic investigation to extract digital evidence regarding an email for further analysis. Emails are subjected to threats and are vulnerable to spoofing, spamming [3], and phishing attacks [4]. Email spoofing [5] is a critical step to a successful phishing attack as the hacker impersonates someone whom the victim trusts [6]. This is due to the SMTP's inherent weakness, lacking source authentication or verification mechanisms. The email header analysis is crucial to email forensics investigation because the critical information related to the email sender along the path from sending *mail transfer agent (MTA)* to receiving MTA can be obtained through the email header metadata [7].

1.3 Memory Forensics

Memory forensics is a crucial element and upcoming sub-branch of digital forensics, where the current state of physical memory (RAM) of a compromised device or application is captured and dumped on the hard disk as a snapshot file for forensic investigation [8,9]. This snapshot file is referred to as a memory dump. The most significant advantage of using memory forensics is that it guarantees non-repudiation and can retrieve data even when end-to-end encryption is

applied at the application or transport layer. Though memory forensics is quite effective in some instances, its practical application has some critical challenges. The biggest challenge is the size of memory dump file. Every snapshot of the entire physical memory is enormous in size and is proportional to the size of the RAM installed on a computing device. Periodically storing or processing memory for any analysis is expensive and time-consuming, and hence only near real-time solutions can be built using such analysis.

1.4 Contribution

This paper proposes a detection scheme that identifies spoofed emails in received and replied emails via live memory forensics. The traditional approach in memory forensics is to capture live memory dump of complete RAM resulting in a large file to be stored, which further requires a significant amount of time for processing and extracting the email header [10]. This approach was improved by capturing all the live processes associated with the browser having multiple tabs [11]. The above method can further be enhanced by addressing the research gap by identifying only the process (amongst all live browser running processes having multiple tabs and web pages opened for browsing) associated with email inbox messages and only capturing it. Therefore if user is running a chrome browser and is browsing ten websites (10 processes), then the objective is to find the one process out of 10 which is associated with the email inbox message. The significant contributions of this paper are, thus, as follows:

- We propose a resource-effective email header extraction process from live memory by identifying 7 novel features from URL to identify the live running email message process by using ML.
- We develop an email spoofing detection algorithm using novel header fields of *ARC* and *References* for received and replied emails.

2 Literature Survey

A literature survey for email spoofing detection techniques and approaches is carried out where research articles after 2013 are considered, as shown in Table 1.

P. Mishra et al. [12] proposed email date and time as measures to detect email spoofing. The algorithm checks the semantics of the date and time fields and matches the sending date and the last date of the received email. Finally, it calculates the threshold or margin of standard time taken by receiving email, which indicates it is a spoofed or legitimate email. S. Gupta et al. [13] proposed spoofed email detection of received emails by examining the header fields of email authentication standards like SPF, DKIM, DKIM-Signature, and DMARC. The proposed algorithm checks the header field values and decides the authenticity of an email. Small dataset and not finding accuracy are its limitation. R. P Iyer et al. [10] proposed spoofed email detection that uses the volatile memory of a system. They captured a host machine's complete volatile memory (RAM)

Table 1. Comparison of similar email spoofing detection schemes

Author(s)	Novelty	Spoofed email detection uses			Email header used for		Database	Dataset Size (Emails)
		Memory Forensic	Received Email	Replied Email	Received Email	Replied Email		
P. Mishra et al. [12]	To detect email spoofing using date and time fields	No	Yes	No	Date, Time	-	Self Generated	Total =3 (All S)
S. Gupta et al. [13]	Developed algorithm based on Authentication header field values	No	Yes	No	SPF, DKIM, D-Sig, DMARC	-	Self Generated	Total =10 (Mix)
R.P Iyer et al. [10]	Applied the concept of memory Forensic for the first time.	Yes	Yes	Yes	Message-ID	InReplyTo	Self Generated	Total =70 (All S)
S. Shukla et al. [11]	Reduced computational complexity By using process forensics	Yes	Yes	No	Message-ID MX record	–	Self Generated	G=50, S=50, Mix =100
O. Oduni bosi [14]	Classification of email header Using ML to detect spoofing	No	Yes	No	From, Message-ID	-	Self Generated	1000
K. Konno et al. [15]	Detect false positive email deliveries In sender domain authentication	No	Yes	No	DMARC	- –	Dmarc Report	1 week D-marc Report
S. Maroof et al. [16]	Large-scale analysis of the adoption of email anti-spoofing schemes	No	Yes	No	SPF, DMARC	–	Open Source	236 mil+ 32k (approx)

G-Genuine, S-Spoofed, FP - False Positive, Mix - both Spoofed and Genuine emails

and extracted email header features from it. The detection algorithm used only one email header called message-ID to check whether the email was legitimate or spoofed. The limitation of this approach was the generation of a large size memory snapshot file resulting in significant time required for the detection algorithm. S. Shukla et al. [11] proposed an improved method where instead of capturing the entire RAM, only live running processes related to the browser were captured. It reduced the size of the captured file significantly and improved the capturing speed. The detection algorithm used message-ID and DNS lookup in real-time to verify the domain IP address. The limitation of this technique was that it captured all the live running processes since it could not identify the exact browser process of all. O. Odunibosi [14], in his work, proposed machine learning to perform email spoofing detection. He extracted the emails from the user inbox using python script, saved the headers in CSV format, and classified the user inbox message as spoofed or legitimate using the RF algorithm. The limitations of such an approach is its dependence on an email server for open-port or protocol for fetching emails, the necessity to write a new script for each email server, and the usage of only one header field (message-ID) to determine spoofed emails. K. Konno et al. [15] suggested an approach to identify legitimate IP addresses by using DMARC report. K-means clustering is applied to find false positives in sender domain authentication. The limitation is that it only works where DMARC is implemented, and it does not check false negatives. S. Maroofi et al. [16] studied and evaluated the adoption rate of SPF and DMARC across a vast set of domains. He proposed an algorithm to detect defensively registered domains and enumerated misconfigured SPF, deployment in sub-domains, and the possibility of sending spoofed emails in a non-existent subdomain by an end-to-end subdomain. The limitation of its approach is that it does not provide any detection algorithm and results in sharing recommendations.

3 Proposed Approach

The proposed scheme of *spoofed email attack detection* is designed to identify spoofed email attacks by analyzing the email header of both received and replied emails. This is achieved by capturing the memory of the live running process associated with an email inbox opened over the browser on the host. The email header information is then extracted from the live memory dump and passed to the detection algorithm to flag spoofed emails.

3.1 System Architecture

The architecture of the proposed system consists of two modules - *The email header capturing module* and *The detection module* - as shown in Fig. 1.

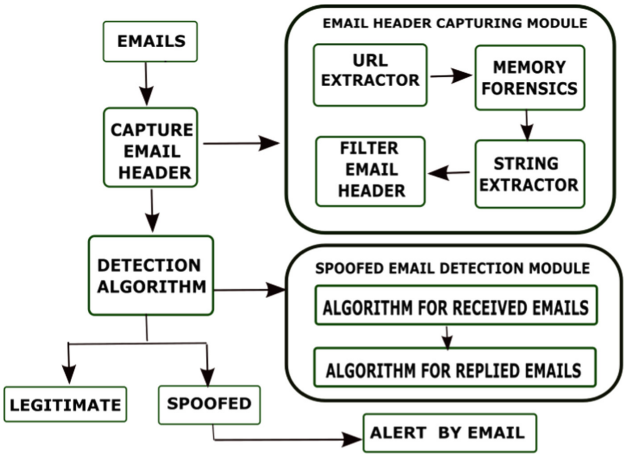

Fig. 1. Architecture diagram of the proposed work

Email header capturing module consists of four sub-modules:

URL Extractor: The task of this sub-module is to extract the URL of a process with an open email inbox and extract features from the URL representing an email transaction. In order to extract features, we have studied 35 most popular email servers currently used to provide email services [17]. Our experiment included extracting the open email URL. For this, we had to create email accounts on these servers. That gave us access to the email inbox. We even sent a few test emails and extracted open email URLs. After obtaining all the URLs from these email servers, we closely examined the structure of the complete URL to extract common features that can be attributed as distinguishable (i.e., features which can reliably be used to identify a URL as an email transaction URL).

Feature Extraction: Any webpage opened is referred to by its Uniform Resource Locator (URL). We have selected eight features (F1-F8) to identify an email based URL. While F1-F6 are Text-based Features that are searched to find the occurrence's of the keyword in URL, F7-F8 are other features. Out of eight features, F1-F7 are novel features and F8 is taken from literature. These features in the structure of opened email URL are discussed below:

- Mail: The 'mail' keyword is very commonly used in URLs, prefixing the domain name of an email server. This is a strong feature that is found in most email servers. E.g., ; is an email message opened URL. Here 'mail' prefixes the google.com domain. In some cases, if the mail keyword is not present as a prefix of a domain, then it is found in the structure of the email URL. Therefore we check the keyword 'mail' in the URL structure and its presence is considered as relevant URL.

$$F1 = \begin{cases} 1, & \text{if 'mail' keyword exists} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

- Message-ID: This is a long alphanumeric string of random characters that are very common in all email URLs. It represents the unique reference given to each message. E.g., https://mail.protonmail.com/inbox/bdb1RXwqXwqR8J-gUhUfMxVb; is an email-based URL where message-ID is mentioned at the end of the URL structure.

$$F2 = \begin{cases} 1, & \text{if the last alphanumeric string exists} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

- Inbox: 'Inbox' keyword is another significant feature found in an email opened URLs. E.g., https://www.fastmail.com/-mail/Inbox/ff47cec8a17710ad.M6eb-41729aIn; some places, the 'inbox' keyword is represented by 'folder/1'. Its presence in URL is marked as relevant.

$$F3 = \begin{cases} 1, & \text{if 'inbox' keyword exists or 'folder/1' exist} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

- Messages: Some email servers use 'message' as a keyword to represent the email message in the URL. Its presence in URLs by some service providers is observed. E.g., https://in.mail.yahoo.com/d/folders/1/message/277?greferrer-a0cHM6-Ly9sb2dpanclBJ55Acd0YZH8t.

$$F4 = \begin{cases} 1, & \text{if 'message' keyword exists} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

- Home directory: This is represented by '/0'in the URL to mark the home directory. E.g., https://outlook.live.com/mail/0/inbox/id/ADAwATMwMtM2NmOC.

$$F5 = \begin{cases} 1, & \text{if '/0' exists} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

– User ID: The 'uid' or 'id' keyword is present in most URLs representing user ID. This feature is commonly cited in most email URLs. E.g., https://mail.yan-dex.com/?uid=1426701416#message/175921860444160001.

$$F6 = \begin{cases} 1, & \text{if 'uid' or 'id' keyword exists} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

– Special Character: The presence of a special character in a URL like '?' is acceptable. This is commonly found in email URLs. E.g., https://mail.tutanota.com/mail/id=?-MaDCBSW-7-2MaDCBXBUZ-2; Any other special symbol found like '@' can be malicious, phished, or irrelevant.

$$F7 = \begin{cases} 1, & \text{if '?' or no special character} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

– HTTPS/HTTP count: We count the number of occurrences of this feature (protocol) in the URL. More than one count is malicious while a single count of HTTPS/HTTP is genuine.

$$F8 = \begin{cases} 1, & \text{if https/http count is } > 1 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

In order to further experiment and find the significance of features, we use ML-based algorithms. Before applying ML, we need to prepare our dataset.

Dataset - Our dataset thus used comprises a total of 33080 URLs, of which 14350 URLs are downloaded from the 35 popular email servers having email features [17] and the rest are non-email URLs that are used from Alexa top websites [18]. We then prepare a CSV file have eight columns representing the eight features (discussed above) and the tuples represent the URL. We take the first tuple, check the keywords and mark them as 1 or 0 based on their presence in the URL. E.g., If a feature (named "mail") is present in the URL, it is marked as 1 (in the "mail" column) and if it is not, then it is represented as 0. In this way, each URL is searched to find the presence of the corresponding feature and they are marked as 1 or 0 in the csv file. The class is characterized as 1, if features are present and 0, if features are not present.

Feature Selection - Feature selection is the most critical part of any data model. Here, our aim is to select the most optimal features that should be independent of any bias in order to generate a highly efficient data model. To achieve this, we used Filtering Method, where we tried Chi-Square(CS) and Information gain (IG) techniques and we found that CS results were not appropriate for our problem due to their dependence on the significance level. Therefore, we chose IG. Further, a brief comparison was also made to understand the discrepancies between Gini Index (GI) and IG based on the Entropy quantifier. In contrast, GI facilitates the bigger distributions and is easy to implement, whereas the IG favors lesser distributions having small counts with multiple specific values. GI

is predominantly used in CART algorithms, while IG is used in ID3, C4.5, and J48 algorithms. Also, GI operates on the categorical target variables, whereas IG computes the difference between entropy before and after the split and indicates the impurity in classes of elements. IG is thus applied to calculate the normalized average impurity using Eq. 9.

$$Gain(S, A) = Ent(S) - \sum_{v \in Values(A)} \frac{|Sv|}{S} Ent|Sv| \tag{9}$$

where, Values(A) is the all possible values of attribute A, and Sv is the subset of S for which attribute A has value v. Ent(Entropy) is calculated using Eq. 10.

$$Entropy(S) = -(P \oplus log_2 P \oplus + P \ominus log_2 P \ominus) \tag{10}$$

Where, P$\oplus$ is the portion of positive examples and P$\ominus$ is the portion of negative examples in S.

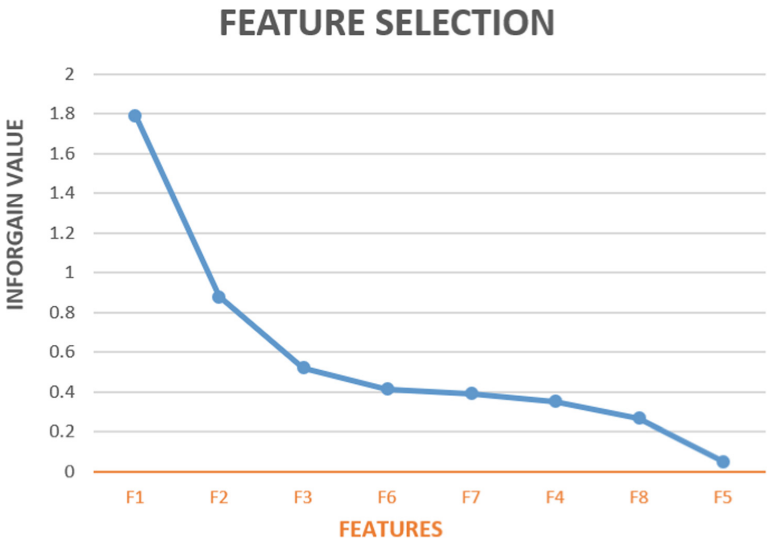

Fig. 2. Feature extraction

Figure 2 shows the ranking of all the features based on their information gain values, where the x-axis shows the features and the y-axis shows the Info Gain value. As we can observe from Fig 2, the gain value stagnates at F6, F7 and F4 and then takes a dip after F8. Thus we select the top 7 features which have the highest significance value and impact on the model.

Classifier Selection: The next step is to select an appropriate classifier based on its performance to predict the model correctly. We use binary classifiers to test our proposed method based on the features described above and classify the

URLs as relevant (for opened emails) or irrelevant (for non-email URLs). Next, 6 binary ML classifiers, Naive Bayes (NB), Support Vector Machine (SVM), Logical Regression (LR), Decision Tree (DT), Random Forest (RF) and K Nearest Neighbour (KNN), are compared with 7 features set (as obtained from feature extraction) where k-fold cross validation (k = 10) is applied.

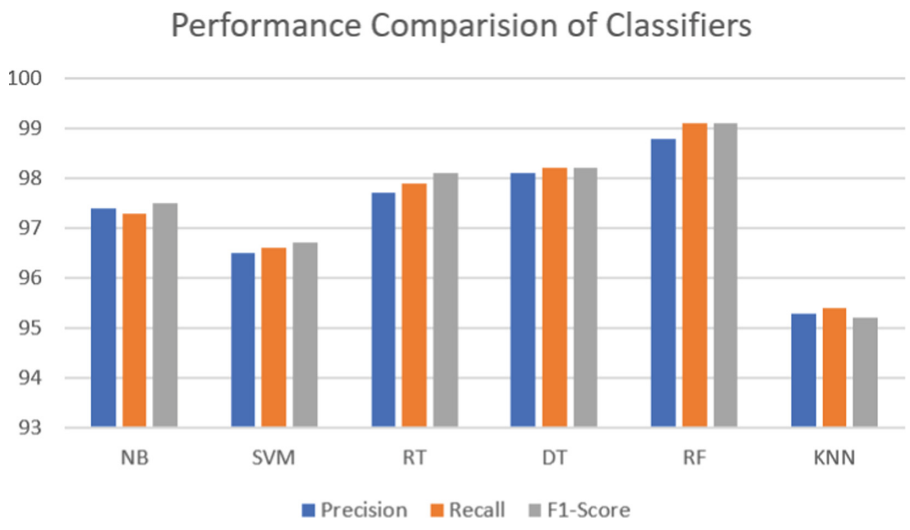

Fig. 3. Comparison of different ML classifier

All the chosen classifier performance is evaluated with precision, recall, and F1-score values, as shown in Fig. 3, respectively. We chose the RF classifier based on its performance (which is highest amongst all), avoids overfitting while model buildup time is marginally higher (1–2%). After the selection of the classifier, the dataset is used to train and test the model. The dataset is divided with various random data splits using 10-fold cross-validation.

Our program works as follows. Each instance of browser execution triggers the event listener of the URL extraction module to obtain the URL and Process ID(PID) of the browser tab and verify whether the URL is of relevance (email is opened in the browser). The relevance of URL is found by applying ML, which uses the 7 email URL features as explained above. If the URL is not relevant, it is immediately rejected, and the program (developed in python) stops.

Memory Forensics: The live active processes from memory associated with email are captured using PID (PID obtained from URL extraction module) by using Magnet Process Capturing Tool [19].

Extract Strings: From captured memory dump, we extract ASCII and UNICODE strings using Microsoft's Sysinternal Strings64 [20].

Filter Email Header: Finally, the email header fields that the Detection Algorithm will use are extracted from the Strings file.

3.2 Detection Algorithm for Received Emails

Email header analysis is essential to identify spoofing attacks in emails. Some of the header fields used in the detection algorithm are explained in Table 2:

Table 2. Header Field Description

Header field	Definition/Description
`From`	E-mail message sender's email address
`To`	The first or main recipient's email address
`Message-ID`	A unique reference to email message created by originating email server
`SPF`	A message value that specifies a valid host that can send emails for that domain
`DKIM`	It validates the origin of an email from a domain through cryptography authentication
`DMARC`	Authenticates emails by checking the alignment of SPF and DKIM checks
`ARC`	Authenticates original email when it is modified between senders and receivers
`References`	It indicates threaded mail reading and message-ID added with each reply
`MX Record`	A DNS record that specifies a domain's valid mail server

To detect spoofed emails received, we extract `SPF`, `DMARC`, `DKIM`, and `ARC` headers, analyze the values of each field as per Algorithm 1, and determine whether an email is spoofed or legitimate. In the case of legacy email servers or service providers that do not use authentication methods, we may not get header field values. In such cases, we rely on message-ID-based spoofing detection.

Algorithm 1: Spoofed Email Detection for Received Emails

```
Input: Received Email Headers
       Result: Genuine Email, Spoofed Email
while read header do
    if Header value != none then
        if DMARC = PASS then
            Genuine Email;
        else
            Check DKIM;
            if DKIM = PASS AND DKIM Signature domain value = domain value of message-ID field then
                Genuine Email;
            else
                Check SPF;
                if SPF = PASS AND ARC = PASS then
                    Genuine Email;
                else
                    Spoofed Email;
                end
            end
        end
    else
        compare (message-ID Domain = From email ID Domain);
        if matched then
            Genuine Email;
        else
            Spoofed Email;
        end
    end
    ReceivedLog = write (From, To, DMARC, DKIM, SPF, ARC)
end
```

3.3 Detection Algorithm for Replied Emails

Algorithm 2 detects spoofed emails replied by an employee by matching the header values of `Reference` and `To` fields. If they match, we declare it as a legitimate email else, we again check it by querying the DNS through forward ns-lookup to fetch `MX records`. The domain name thus obtained is matched with `Reference` field. In case of a match, we consider it as a genuine email else, we declare it as a spoofed email.

Algorithm 2: Spoofed Email Detection for Replied Emails

```
Input: Replied Email Headers
       Result: Genuine Email, Spoofed Email
while read header do
    From = From Field of Header;
    To = To Field of Header;
    References = References Field of Header;
    ToDName = GetDomainName (To);
    RefDName = GetDomainName (Reference);
    comp = Compare (ToDName, RefDName)
    if comp = TRUE then
        Genuine Email;
    else
        Check MX Record;
        nslkupMX = nslookup (ToDName);
        comp1 = Compare (nslkupMX, RefDName);
        if comp1 = TRUE then
            Genuine Email;
        else
            Spoofed Email;
        end
    end
    RepliedLog = write (From, To, References)
end
```

A local database of `MX records` is maintained to increase algorithm speed and reduce the bottleneck and dependency on internet connectivity. The algorithm first queries the local database and then queries the DNS and also saves

this value to keep the database updated. The `MX record` matching is applied when a genuine email fails to match `Reference` with `To` field, as shown in Fig. 4.

Fig. 4. Genuine email fails in `To` and `reference` matching

It has been experimentally tested and identified that `Reference` matching fails when enterprises purchase email-related services from email vendors. For instance, if an enterprise (here, *iimraipur*) uses G-suite from Google, the emails will retain the original domain (*iimraipur.ac.in*), but the `Reference` generated by MTA of originating email-server (here, *Gmail*) has the Google domain. Comparing this with the `To` field domain will always show a legitimate email as spoofed. This shortcoming is resolved with `MX record` matching, as shown by our program in Fig. 5.

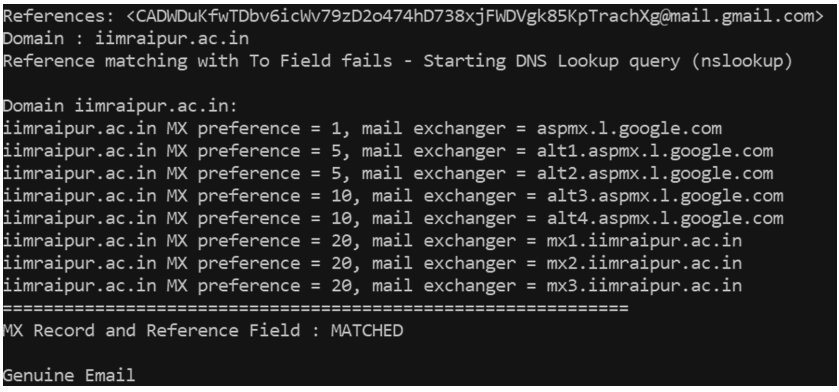

Fig. 5. `MX record` matching

4 Experimental Setup and Testing

4.1 Assumption

The laptop used for testing is free from any infection from malicious programs. Hence, it is assumed that the memory dump used for analysis contains an unmodified/authentic version of the live memory.

4.2 Experimental Setup

The proposed method is tested on a laptop having Intel(R) Core(TM) i5-7200U CPU @ 2.7 GHz as processor, x64-based processor with 64-bit Operating System (Windows 10), 16 GB and 1 TB HDD. To test legitimate and spoofed emails, both types of emails are sent. Fake emails were sent from anonymous or fake email services like anonymailer.net, sendanonymousemail.net, emkei.cz, and spoofbox.com, whereas genuine emails are sent from Gmail and Yahoo. The proposed method is deployed as a client-side solution with alert messages sent to the admin over HTTP. Every PC or laptop provided to the working employees in any organization has pre-loaded software installed, and this tool is expected to be one of them in commercial deployment whose primary task is to detect and report spoofed emails being received or replied by an employee in an organization to administrators.

Our program is developed in python, which runs as a listener event to check the browser instance to call back and activate the URL extractor function. The output of the URL extractor passes the PID to the process capturing tool, which generates memory snapshot file of the process associated with PID. This file is searched using String64 to finally extract the email headers. Further, the header field values are then extracted and saved to create a CSV file. This CSV file is used in our python code by using pandas (an open-source data analysis package) to implement a spoofing detection algorithm.

5 Results and Discussion

5.1 Results of URL Extractor

The performance of the URL extractor module to find relevant URLs is evaluated using accuracy-98.8, precision-98.9, recall-97.3, and F1 score-99.1. The results thus obtained show the model has performed better with an average metric score of 98%.

5.2 Detection Algorithm

In order to test the detection algorithm accuracy, a real case scenario consisting of both genuine and spoofed emails is taken. The dataset is collected via an automated browser extension program that can open the full path of email header metadata of emails in inboxes over the browser and dump them as a file to be used for header extraction. The browser extension uses a content script to read the contents of the *document object module (DOM)* and hence the raw text required for header extraction. The extension, however, needs to follow the email inbox DOM that has the necessary information to construct URLs that can lead us to the individual emails' original or raw headers information. The active tab permission is enough in the manifest file to allow browser extension to access the DOM of a page. The dataset thus obtained consists of 17200 emails (a mix of both spoofed and genuine emails where spoofed email = 2967 and genuine

email = 14233). Emails from the 3 most popular public email servers of Gmail, Yahoo and Rediffmail are used for testing the detection algorithm. The dataset is further anonymized, and no privacy issues have been violated in preparing the dataset. A comparison of our proposed approach results with other standard state-of-art email spoofing detection methods is shown below in Table 3.

Table 3. Comparison of the proposed approach with other standard methods

Approach	Accuracy	Precision	Recall	F1 score
P. Mishra [12]	Not given			
S. Gupta [13]	Not given			
R. P Iyer [10]	96	–	–	–
S. Shukla [11]	98	–	–	99
O.Odunibosi [14]	1	99	1	1
K. Konno [15]	SPF+DKIM FP=7%, DMARC FP=50%			
S. Maroof [16]	Recommendations			
Proposed	Received=96.15	96.9	97.89	97.39
Method	Replied=95.09	95.68	97.51	96.59

5.3 Resource Utilization

Performance analysis is carried out by executing the proposed scheme on the user machine while considering various system parameters. Average values of *CPU* utilization, *disk* usage, *memory* consumption, and *time* required for the execution of the detection algorithm were observed as shown in Fig. 6.

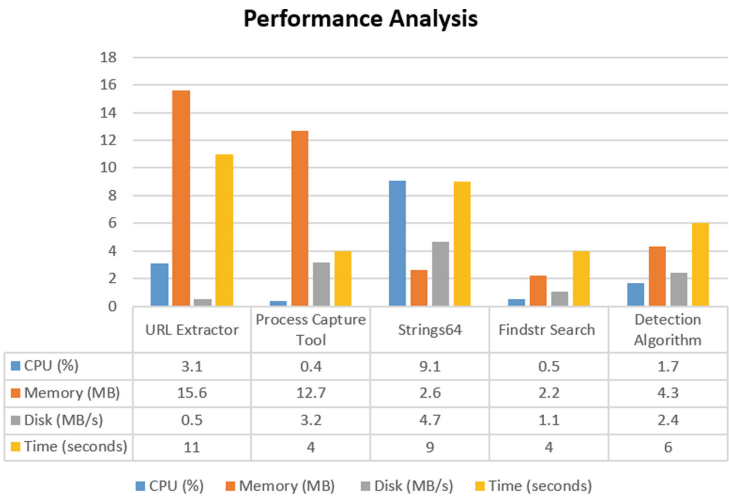

Fig. 6. Performance analysis

Figure 7 shows the detection time required for varying sizes of captured data files. These are the memory dump (dmp file) captured from live memory. The average detection time is 34.57 s for an average data file of size 161.7 MB.

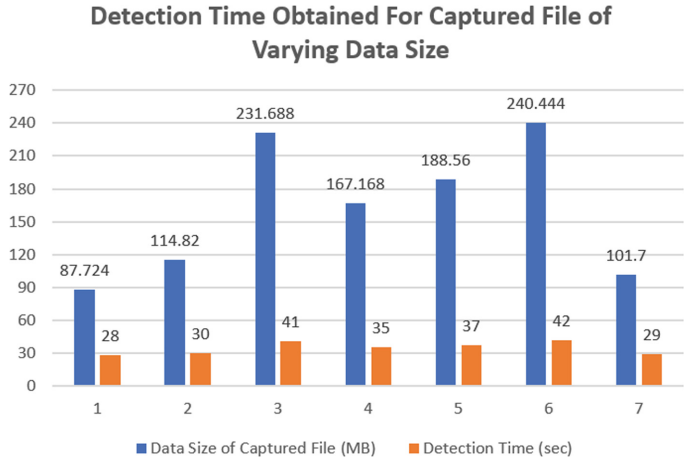

Fig. 7. Detection time with varying sizes of captured data file

Figure 8 contains *PM* as the proposed method and [10] and [11] as similar state-of-the-art references that are used for comparison.

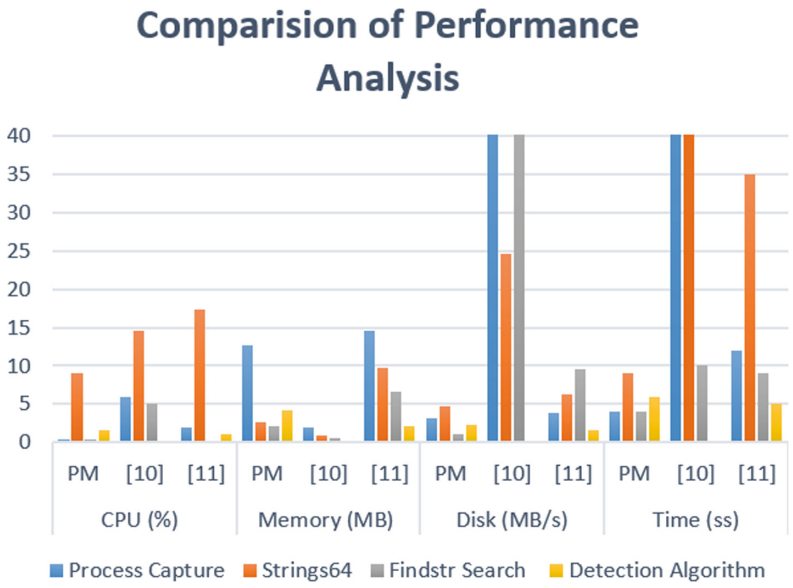

Fig. 8. Comparison of resource utilization

The total time, thus, required by the proposed method from capturing live process dump to detecting spoofed email is 35 sec approx. This is better than the 12 min time taken for performance analysis by a similar method [10] or the 1 min time taken by the process-based approach [11], as shown in Fig. 8.

5.4 Comparison Points in the Benchmarks and Proposed Framework

After recognizing and defining the comparative checklist issues, our proposed method was compared with those from other relevant studies with the help of a benchmarking checklist that is shown in Table 4. Comparison results show that most of the benchmark studies obtained scores between 16.66% to 83.33%, covering 1 to 5 benchmark points, whereas our proposed method covered all the points and obtained a score of 100%. The comparison score also validates our comparison analysis with [10,11], as both of them are the top two highest scores. The study of previous benchmarks was mainly focused on the inbound attack (received emails) and did not consider outbound attack by an inside user (relied email). In contrast, our proposed method focuses on both types of attacks. Though some studies saved detection results in log files, any kind of alert mechanism was not used in the past. Also, the key benchmark points missed by previous research studies were related to the number of header fields used for detection, dataset size, detection accuracy and detection time.

Table 4. Benchmarking checklist

Comparison points	P. Mishra et al. [12]	S. Gupta et al. [13]	R.P. Iyer et al. [10]	S. Shukla et al. [11]	O. Oduni et al. [14]	K. Konno et al. [15]	S. Maroof et al. [16]	Proposed work
Handling received and replied emails	x	x	✓	✓	x	x	x	✓
Email alert mechanism	x	x	x	x	x	x	x	✓
Save the results as log files	x	x	✓	✓	x	x	x	✓
No of email headers used (more than 1)	✓	✓	✓	✓	✓	x	✓	✓
Large dataset greater then 1000	x	x	x	x	✓	✓	✓	✓
Detection time (less than 1 min)	x	x	x	✓	x	x	x	✓
Accuracy greater than 95%	x	x	✓	✓	✓	x	x	✓
Score	16.66%	16.66%	66.66%	83.33%	50%	16.66%	33.33%	100%
Difference	83.33%	83.33%	33.33%	16.66%	50%	83.33%	66.66%	–

5.5 Commercial Applications and Limitations

The proposed tool can be deployed as a client-side tool that can detect email spoofing. The novelty of memory forensics is that the organization can even prevent email spoofing in scenarios where it does not own the email server used by the employee in the organization. The tool can detect email spoofing even when the email is received on a personal email service used by the employee on the host provided by the organization. The limitation of the proposed scheme is that it is currently tested on webmail-based email services.

6 Conclusion and Future Work

The advantage of using memory forensics in spoofed email detection is that it guarantees non-repudiation of a digital trace of the host in physical memory [8]. Further, by identifying the exact processes and only capturing them addresses the disadvantage of capturing the entire physical memory. In our work, we examine the current URLs to identify the ones related to opened emails and then capture this live process to perform header extraction and further apply detection algorithm to identify and store the results in respective log files. The alert mechanism of emailing the log file, thus, gives a threat profile and early warning to the IT admins and security team of the organization to initiate further forensic investigation [21] and adopt a suitable mitigation strategy for such threat scenarios [16]. Our performance analysis shows that the proposed work takes approximately 35 s to complete the email detection process with minimum false positives. This is achieved with minimal consumption of system resources, least overheads, and without interference with the normal functioning of the user's system. Also, the earlier practice of periodic program scheduling with a fixed time interval had challenges in determining the time interval and storage of irrelevant memory dumps. By replacing scheduling with the callback function to auto-trigger, any instance of browser execution saved system resources and overhead significantly. It is also better than the browser extension method of spoofed email detection due to the ease with which users can switch off the extension [22]. Also, the browser extension method is not browser independent and one has to write different extension codes for different browsers used. Future work can be extended to include other email client-side applications such as Outlook, Postbox, Apple Mail, Mozilla Thunderbird, etc. Similarly, the proposed scheme can be extended to mobile phones to test its performance and efficacy on android and apple OS used in mobiles.

Compliance with Ethical Standards

Conflict of Interest. The authors declare that they have no conflict of interest.

Ethical Approval. This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Lutui, R.: A multidisciplinary digital forensic investigation process model. Bus. Horiz. **59**(6), 593–604 (2016)
2. Fruhlinger, J.: Top cybersecurity facts, figures and statistics. CSO Online, IDG (2020)
3. Sheikhalishahi, M., Saracino, A., Martinelli, F., Marra, A., Mejri, M., Mejri, N.: Digital waste disposal: an automated framework for analysis of spam emails. Int. J. Inf. Secur. **19**, 499–522 (2020)
4. Gupta, B.B., Arachchilage, N.A.G., Psannis, K.: Defending against phishing attacks: taxonomy of methods, current issues and future directions. Telecommun. Syst. J. **67**, 247–267 (2018)
5. Mooloo, D., Fowdu, T.P.: An ssl-based client-oriented anti-spoofing email application. Africon, pp. 1–5 (2013)
6. Hu, H., Peng, P., Wang, G.: Towards understanding the adoption of anti-spoofing protocols in email systems. In: IEEE Cybersecurity Development (SecDev 2018) (2018)
7. Hunt, R., Zeadally, R.: Network forensics: an analysis of techniques, tools, and trends. IEEE Comput. **45**(12), 36–43 (2012)
8. Pagani, F., Fedorov, S., Balzarotti, D.: Introducing the temporal dimension to memory forensics. ACM Trans. Priv. Sec. **22**(9), 1–21 (2019)
9. Parida, T., Das, S.: Pagedumper: a mechanism to collect page table manipulation information at run-time. Int. J. Inf. Secur. **20**, 603–619 (2021)
10. Iyer, R., Atrey, P.K., Varshney, G., Misra, M.: Email spoofing detection using volatile memory forensics. In: IEEE Conference on Communications and Network Security (CNS), Las Vegas, NV, pp. 619–625 (2017)
11. Shukla, S., Misra, M., Varshney, G.: Identification of spoofed emails by applying email forensics and memory forensics. In: Published in Proceeding of ACM Digital Online, 10th International Conference (ICCNS 2020), pp. 109–114 (2020)
12. Mishra, P., Pilli, E., Joshi, R.: Forensic analysis of e-mail date and time spoofing. In: Third International Conference on Computer and Communication Technology (2013)
13. Gupta, S., Pilli, E.S., Mishra, P., Pilli, S., Joshi, R.C.: Forensic analysis of e-mail address spoofing. In: 5th IEEE International Conference on the Next Generation Information Technology Summit, pp. 898–904 (2014)
14. Odunibosi, O.: The classification of email headers using random forest algorithm to detect email spoofing. School of Computing, National College, Ireland (2019)
15. Konno, K., Kitagawa, N., Yamai, N.: False positive detection in sender domain authentication by dmarc by dmarc report analysis. In: 3rd International Conference on Information Science and System ICISS, pp. 38–41 (2020)
16. Maroofi, S., Korczynski, M., Hölzel, A., Duda, A.: Adoption of email anti-spoofing schemes: A large scale analysis. IEEE Trans. Netw. Service Manage. 1–1 (2021)
17. Hu, H., Wang, G.: End-to-end measurements of email spoofing attacks. In: Proceedings of 27th USENIX Security Symposium (2018)
18. Alexa: Alexa most popular website. http://www.alexa.com/topsites
19. Magnet: Magnet process capture tool. https://www.magnetforensics.com/resources/magnet-process-capture/
20. Microsoft: Windows sysinternals strings v2.53. https://docs.microsoft.com/en-us/sysinternals/downloads/strings

21. Banday, M.T.: Analysing e-mail headers for forensic investigation. J. Digital Foren. Sec. Law **6**, 49–64 (2011)
22. Sanchez, P., Tapiador, J., Schneider, G.: After you, please: browser extensions order attacks and countermeasures. Int. J. Inf. Secur. **19**, 623–638 (2020)