Mathematical Problems of Computer Science 64, 56–65, 2025.

doi:10.51408/1963-0140

UDC 004

Development of the Automated Alert System for Detection and Handling of Compromised Email Accounts

Arthur S. Petrosyan, Gurgen S. Petrosyan and Robert N. Tadevosyan

Institute for Informatics and Automation Problems of NAS RA, Yerevan, Armenia
e-mail: arthur@sci.am, gurgen@sci.am, robert@sci.am

Abstract

With the increasing volume of email-based attacks and unauthorized access to mail servers, the need for automated monitoring and response mechanisms has become essential. This paper presents the development of an automated alert system designed to detect and handle compromised email accounts. The system monitors the mail server queue on a Linux server, detecting anomalies based on a significant surge in queued messages. Upon identifying suspicious activity, the system attempts to determine the username associated with the highest number of SASL authentications and triggers appropriate alerts or mitigation actions. Additionally, the system is integrated with a Telegram bot, allowing administrators to take immediate corrective actions remotely. This approach provides a lightweight, effective method for preventing email abuse and ensuring the integrity of email servers.

Keywords: Email, SPAM, Alert system, Telegram bot**.**
Article info: Received 25 April 2025; sent for review 2 May 2025; accepted 13 November 2025.

1. Introduction

Mail servers are common targets for malicious attackers seeking to exploit compromised accounts for spam and phishing campaigns. Once an email account is compromised, it can be used to send tens of thousands of spam emails before the breach is detected. Such activities can lead to unwanted load for the mail system, blacklisting of mail servers, reputational damage for the whole origin network, and security breaches. Traditional security mechanisms such as rate-limiting, IP-based restrictions, fail2ban [1], Rspamd [2] or Spamassassin [3] offer some level of protection, but they are not enough for immediate detection and mitigation of compromised email accounts.

This paper presents the implementation of an automated alert system that continuously monitors mail server queues and detects unusual spikes in queued messages. The system uses SASL authentication [4] logs to identify the account responsible for excessive email submissions. Additionally, it is integrated with a Telegram bot API [5], enabling administrators to receive alerts and take mitigation actions remotely. By automating the detection and response process, the proposed system minimizes response time by enabling immediate intervention, thus preventing further misuse of compromised accounts and reducing the impact of account compromises.

2. Related work

Several existing solutions aim to detect compromised email accounts, including:

- Intrusion detection systems (IDS) [6];
- Rate-limiting mechanisms [7];
- Reputation-based filtering using Domain Name System blocklists (DNSBL) [8];

However, these methods often rely on external data sources or predefined thresholds, which may not adapt to evolving attack patterns.

The system proposed in this paper offers a real-time, in-server detection mechanism that is independent of third-party services. It is not intended as a replacement for other solutions, but as an additional level of protection for the mail server.

3. Postfix considered as MTA

In this paper, we consider a mail server to run based on Postfix [9]. Postfix is one of the most widely used mail transfer agents (MTAs) for handling email delivery on Unix-like operating systems. Postfix was originally developed by Wietse Venema as a secure alternative to the standard UNIX MTA - Sendmail. Formerly known as Vmailer, Postfix was released by the end of 1998 as the IBM Secure Mailer and later renamed to Postfix [10, 11].

Postfix is known for its performance, flexibility, and ease of configuration. It supports various authentication mechanisms, filtering capabilities, and queue management features, making it a preferred choice for mail servers ranging from small deployments to enterprise-level environments. Due to its modular design, Postfix integrates well with security tools, spam filters, and monitoring systems, ensuring reliable and efficient email processing.

4. System Design and Implementation

The automated alert system is running regular checks for a sudden increase in the mail queue size. In case an unusual number of messages is detected in the queue, additional checks are performed to identify the cause of the anomaly. Most often, this indicates that the password of an email address has been hacked (usually by brute force) and spam has been sent from that address. If the assumption is confirmed by additional checks of log files, the system triggers an alert and takes emergency steps to prevent further damage. Details of the system are described below.

4.1 Mail Queue Monitoring

Postfix manages email delivery through a structured mail queue system, which consists of several directories where messages are temporarily stored during processing. The main queue directories include:

- **Incoming Queue** (*/var/spool/postfix/incoming*) – Stores new messages before they are processed by Postfix.
- **Active Queue** (*/var/spool/postfix/active*) – Contains messages that are actively being delivered. Postfix prioritizes these messages for immediate processing.
- **Deferred Queue** (*/var/spool/postfix/deferred*) – Holds messages that could not be delivered on the first attempt. Postfix retries delivery at scheduled intervals.
- **Maildrop Queue** (*/var/spool/postfix/maildrop*) – Used for messages submitted locally by Postfix-compatible senders.
- **Hold Queue** (*/var/spool/postfix/hold*) – Stores messages that require manual intervention before being processed further.

Each message in these directories is stored as a separate file with metadata that helps Postfix track its delivery status. The queue management system ensures efficient handling of emails, preventing congestion and optimizing delivery performance. Administrators can monitor and manage the queue using commands such as "*postqueue -p*" to list queued messages.

Since our goal is to detect spam outbreaks, which typically result in a sudden surge of emails being queued, monitoring the mail queue size is a crucial step. A basic way to achieve this is by using the "*postqueue -p*" command, which lists all queued messages and allows administrators to assess the queue status. Thus, the number of messages in the queue can be obtained at any time with a command sequence like "*postqueue -p | tail -n +2 | wc -l*".

However, this approach requires parsing command output, which can be inefficient, especially during a large-scale spam event when thousands (or even tens of thousands) of messages may be queued. To improve performance, we opt for a more efficient method - directly counting the number of files in the Postfix queue directories (Incoming Queue, Active Queue, Deferred Queue and Maildrop Queue). This approach eliminates the overhead of command execution and text processing, providing a faster and more reliable way to detect email surges in real-time.

It should be noted that we do not need to check the Hold Queue *(/var/spool/postfix/hold*) because messages in this directory are placed there manually or through policy-based filtering and are not actively processed for delivery. Unlike all other Postfix queue directories, which reflect real-time email flow, the hold queue contains messages that require administrative intervention before they can proceed. Since a spam outbreak results in a rapid increase in automatically queued messages, monitoring the hold queue would not provide meaningful insights into the outbreak's severity or impact. Instead, focusing on the other queues ensures that we detect and respond to excessive email generation more efficiently.

4.2 Periodic checks

To ensure continuous checks, the monitoring script should run periodically as either a Cron job [12] or a Systemd service timer [13], providing real-time detection of spam outbreaks without significant resource usage. Since it only performs lightweight operations - counting files in the mail queue and parsing logs - it runs efficiently without impacting system performance.

4.2.1 Monitoring with Cron Job

Generally, configuration with a Cron job is easier and sufficient. It only requires adding a line to the root user's crontab configuration, like the following example:

**/2 * * * * /path/to/script.sh*

As a result, the script will be continuously executed every 2 minutes.

But there is an important detail about Cron jobs that should be noted forimportant monitoring actions like the one described in this work. Running of any Cron job always depends on the cron service, which is a separate process (typically named cron or crond, depending on the Linux distribution), responsible for running scheduled tasks. So, if crond/cron accidentally stops or crashes, none of the scheduled jobs would run until the service is restarted. That is why the next systemd service timer-based solution is more preferable for our task.

4.2.2 Monitoring with Systemd Service

The key advantage of systemd timers over cron jobs is that they are managed by the systemd process. And since systemd is the process number one in the Linux system, it is always running as long as the system is up. This would ensure that any scheduled script will execute reliably. Even if a timer fails, the systemd process can restart it automatically.

However, configuring systemd timer requires much more efforts than just adding a line in Cron configuration. Below, we present an example of such a configuration.

Creating a systemd service file (*/etc/systemd/system/spam-check.service*):

[Unit]
Description=Spam Outbreak Detection Script
After=network.target
[Service]
ExecStart=/path/to/script.sh
Restart=always
User=root
[Install]
WantedBy=multi-user.target

Creating a timer *(/etc/systemd/system/spam-check.timer*):

[Unit]
Description=Run spam check script every minute
[Timer]
OnBootSec=2min
OnUnitActiveSec=2min
[Install]
WantedBy=timers.target

Enabling and starting the timer:

systemctl enable --now spam-check.timer

The above *systemd* timer configuration ensures the script runs at regular intervals, detecting anomalies promptly while maintaining system efficiency.

It should be noted that the checking frequency of every 2 minutes mentioned above was given as an example, but it can be adjusted according to need. At the same time, it is important to mention the fact that the log file check (which can take a considerable amount of time) is performed only at the second stage, in case of detection of exceeding the threshold of the mail queue size. This

means that in most cases, the check time will be very insignificant. We have checked this assumption by running the script (with the "time" prefix) to determine its execution time, and on average, we found it to be about 12-17 milliseconds. Thus, frequent script launching has virtually no effect on system load while at the same time ensuring timely response to anomalies.

4.3 Identifying the Compromised Account

Identifying the compromised account is completed at a second stage only if the mail queue size is detected to exceed the defined threshold. This is accomplished by analyzing the mail server log file. On each run, the script parses SASL authentication messages in the current mail server log file to determine which user has authenticated most frequently within a recent timeframe (on most Linux systems the default place for such messages is */var/log/mail.log*).

Log files are generally subject to rotation,which are most often performed either on a weekly or monthly basis. In any case, the checks described below reveal the most recent data for the period after the last file rotation. The command sequence used for the checks is presented below:

```
cat /var/log/mail.log | awk -F"sasl_username=" '{print $2}' | sort | uniq -c | sort -nr | head -6 | tail -5
```

The result is a list of sorted top 5 authentications from the last period. In case of a spam outbreak, the first line (or the first few lines) will have a comparatively very large number of successful authentications, which is a proof of a potentially compromised account.

An example of the alert message is presented below:

```
Mail server queue has exceeded the specified threshold.
Current queue size is: 12869
Top 5 SASL authentications from /var/log/mail.log:
 15628 [compromised-mail-address]@somedomain
   234  [other-mail-address1]
   122 [other-mail-address2]
   104 [other-mail-address3]
    86  [other-mail-address4]
```

In case such an anomaly is detected, the system immediately flags that email account as potentially compromised, notifies the administrator of the problem, and provides the opportunity to take preventive actions, which are described below.

4.4 Telegram Bot Integration

Integration with Telegram Bot is performed according to Telegram Bot API Documentation [5]. The script is integrated with the Telegram Bot to send alerts. Administrators receive messages with details such as the compromised account, suspicious activity, and recommended actions.

An example of a script to implement integration with a Telegram Bot is presented below:

```
#!/bin/bash
tgbot="**********************"
tgchatid="******************"
tgurl="https://api.telegram.org/bot"$tgbot"/sendMessage"
QUEUEDIR_ROOT="/var/spool/postfix"
MAX_QUEUE_LENGTH=***
# Get the number of messages sitting in each postfix queue directory
Q_ACTIVE=$(find ${QUEUEDIR_ROOT}/active -type f | wc -l)
Q_INCOMING=$(find ${QUEUEDIR_ROOT}/incoming -type f | wc -l)
Q_DEFERRED=$(find ${QUEUEDIR_ROOT}/deferred -type f | wc -l)
Q_MAILDROP=$(find ${QUEUEDIR_ROOT}/maildrop -type f | wc -l)
# If any of these queues contain more than $MAX_QUEUE_LENGTH issue an alert
if [ ${Q_ACTIVE} -gt ${MAX_QUEUE_LENGTH} -o ${Q_INCOMING} -gt
${MAX_QUEUE_LENGTH} -o ${Q_DEFERRED} -gt ${MAX_QUEUE_LENGTH} -o
${Q_MAILDROP} -$
Q_TOP5=$( cat /var/log/mail.log | awk -F"sasl_username=" '{print $2}' | sort | uniq -c | sort -nr | head -
6 | tail -5)
tgmessage="Queue on ${HOSTNAME} is: $((${Q_ACTIVE} + ${Q_INCOMING} + ${Q_DEFERRED}
+ ${Q_MAILDROP}))"$'\n'$Q_TOP5
curl -s -d
"chat_id="$tgchatid"&text=$tgmessage&parse_mode=markdown&disable_web_page_preview=1"
$tgurl > /dev/null 2>&1
exit 2
fi
exit 0
```

4.5 Mail Queue Clean up

Since Postfix does not provide a ready mechanism to remove messages from the queue based on the sender's email address, we use the following solution to clean up the mail queue, based on a compromised email address pattern:

```
#!/bin/bash
# Define the email pattern to search for
ADDRESS_PATTERN="$1"
if [[ -z "$ADDRESS_PATTERN" ]]; then
   echo "Usage: $0 '<email-pattern>'"
   echo "Example: $0 'spam@domain.com' or $0 '@domain.com'"
   exit 1
fi
# Get queue IDs for messages matching the pattern
QUEUE_IDS=$(postqueue -p | grep -B1 "$ADDRESS_PATTERN" | grep '^[A-F0-9]' | cut -d' ' -f1)
# Check if any queue IDs were found
if [[ -z "$QUEUE_IDS" ]]; then
   echo "No matching emails found in the queue."
   exit 0
fi
# Remove matching emails
echo "Removing emails matching pattern: $ADDRESS_PATTERN"
echo "$QUEUE_IDS" | xargs -r postsuper -d
echo "Done."
```

4.6 Mitigation Actions

Upon detecting a compromised account, the alert system allows administrators to take predefined mitigation actions via Telegram bot, including:

- stop/start mail service;
- mail queue is cleanup (remove queued mails form the compromised email address);
- lock/unlock/generate new password for compromised email account.

Besides notifications and managing with the Telegram bot, the system can also send alerts via email.

5. Results and Evaluation

The described alert system was tested on a Linux-based mail server running Postfix. The key findings include:

- Compromised accounts are successfully detected in real-time within several minutes after spam outbreak, significantly reducing response times compared to manual detection.
- The system prevented the mail server from being blacklisted by proactively blocking malicious email bursts.
- The Telegram bot significantly improved response times by allowing administrators to take action remotely.
- Low resource usage made it feasible the system to run constantly without impacting mail server performance.
- Solution to check Postfix mail queue directories is more efficient than using "postqueue -p" command, as it avoids parsing output and provides an immediate count of queued messages, which is crucial in case of large-scale spam outbreak, involving thousands or even tens of thousands of spam emails suddenly filling up the mail queue.
- Solution to configure systemd timers instead of Cron job ensures checks do not depend on the cron process existance.

6. Conclusion

The developed automated alert system provides an effective solution for detecting and handling compromised email accounts. It significantly improved detection accuracy and reduced administrative workload. At the same time, it is not a replacement for standard protection methods, but rather complements them perfectly.

By leveraging real-time mail queue monitoring and SASL authentication analysis, it enhances email security while minimizing the need for manual intervention. The integration of a Telegram bot further improves response efficiency, allowing administrators to act quickly during a spam outbreak.

The system described in this paper, has been successfully tested on the Academic Scientific Research Computer Network of Armenia (ASNET-AM) [14] and is currently being actively used for implementation of set goals.

References

[1] Fail2Ban: ban hosts that cause multiple authentication errors. [Online]. Available: https://github.com/fail2ban/fail2ban

[2] Rspamd,Fast, free and open-source spam filtering system. [Online]. Available: https://rspamd.com

[3] Apache SpamAssassin, Open Source anti-spam platform. [Online]. Available: https://spamassassin.apache.org/

[4] SASL Authentication Mechanism, [Online]. Available: https://www.cyrusimap.org/sasl

[5] Telegram Bot API Documentation, [Online]. Available: https://core.telegram.org/bots/api

[6] What is an intrusion detection system (IDS)? [Online]. Available: https://www.ibm.com/think/topics/intrusion-detection-system

[7] Postfix Performance Tuning. [Online]. Available: https://www.postfix.org/TUNING_README.html

[8] Domain Name System blocklist. [Online]. Available: https://en.wikipedia.org/wiki/Domain_Name_System_blocklist

[9] Postfix MTA. [Online]. Available: http://www.postfix.org

[10] Sharing Software, IBM to Release Mail Program Blueprint. [Online]. Available: https://archive.nytimes.com/www.nytimes.com/library/tech/98/12/biztech/articles/14blue.html

[11] Postfix free and open-source mail transfer agent (MTA). [Online]. Available: https://en.wikipedia.org/wiki/Postfix_(software)

[12] Cron job scheduler. [Online]. Available: https://en.wikipedia.org/wiki/Cron

[13] Systemd. [Online]. Available: https://en.wikipedia.org/wiki/Systemd

[14] The Academic Scientific Research Computer Network of Armenia. [Online]. Available: (ASNET-AM) https://asnet.am

Էլեկտրոնային փոստի կոտրած հաշվեհամարների հայտնաբերման և կառավարման ահազանգման ավտոմատացված համակարգի մշակում

Արթուր Ս. Պետրոսյան, Գուրգեն Ս. Պետրոսյան և Ռոբերտ Ն. Թադևոսյան

ՀՀ ԳԱԱ Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտ, Երևան, Հայաստան
e-mail: arthur@sci.am, gurgen@sci.am, robert@sci.am

Ամփոփում

Էլեկտրոնային փոստի վրա հիմնված հարձակումների և փոստային սերվերների նկատմամբ ոչ օրինական մուտքերի աճի ֆոնին կարևոր է ունենալ ավտոմատացված մոնիթորինգի և արձագանքման մեխանիզմներ: Այս հոդվածում ներկայացված է նման համակարգի մշակում, որն ուղղված է կոտրված էլ. փոստի հաշվեհամարների բացահայտման և վերահսկման խնդիրներին: Համակարգը մոնիթորինգ է իրականացնում Linux համակարգում գտնվող փոստային սերվերի հերթի չափի վրա՝ հերթի մեջ նամակների քանակի անսովոր կտրուկ աճ հայտնաբերելով: Նման դեպքում համակարգը փորձում է պարզել այն օգտատիրոջ անունը, որն ունի ամենաշատ SASL նույնականացումներ և կատարում է համապատասխան կանխարգելիչ գործողություններ: Բացի այդ, համակարգը ինտեգրված է Telegram բոթի հետ՝ հնարավորություն տալով ադմինիստրատորներին հեռավար կերպով կատարել շտկող գործողություններ: Այս մոտեցումը ապահովում է պարզ և արդյունավետ մեթոդ՝ կանխարգելելու էլ. փոստի չարաշահումը և ապահովելու փոստային սերվերների անվտանգությունը:

Բանալի բառեր՝ Email, SPAM, Alert system, Telegram bot.

Разработка автоматизированной системы оповещения для обнаружения и обработки взломанных учетных записей электронной почты

Артур С. Петросян, Гурген С. Петросян и Роберт Н Тадевосян

Институт проблем информатики и автоматизации НАН РА
e-mail: arthur@sci.am, gurgen@sci.am, robert@sci.am

Аннотация

С ростом объема атак на электронную почту и несанкционированного доступа к почтовым серверам потребность в автоматизированных механизмах мониторинга и реагирования становится все более существенной. В этой статье представлена разработка

автоматизированной системы оповещения, предназначенной для обнаружения и обработки взломанных учетных записей электронной почты. Система отслеживает очередь почтовой службы на сервере Linux, обнаруживая аномалии на основе резкого всплеска количества сообщений в очереди. При выявлении подобной активности система пытается определить имя пользователя, связанное с наибольшим количеством аутентификаций SASL, и запускает соответствующие оповещения или действия по устранению последствий. Кроме того, система интегрирована с ботом Telegram, что позволяет администраторам удаленно предпринимать немедленные корректирующие действия. Такой подход обеспечивает быстрый и эффективный метод предотвращения злоупотреблений электронной почтой и обеспечения целостности почтовых серверов.

Ключевые слова: Email, SPAM, Alert system, Telegram bot.