Detection of Callback Phishing Attacks With OCR

Author: Bayarmaa Khosbayar Havel, bamaa2000@yahoo.com

Advisor: Lenny Zeltser

Accepted: July 20, 2023

Abstract

Phishing attacks using fake invoice image attachments, commonly known as callback phishing attacks, have emerged as a prevalent and concerning threat (Long, 2023). This type of fraud relies on cybercriminals sending deceptive emails claiming recipients have subscribed to costly services, confusing the recipients who never signed up for such subscriptions. These attacks could escalate into serious ransomware incidents (Toulas, 2022). The broader implications of these attacks include eroding trust in online communications, transactions, and the economy.

This research investigates the feasibility of creating a localized script against this callback phishing fraud as a defense mechanism. The primary goal is to develop accessible, cost-effective solutions that cater to organizations with limited resources. This research aims to empower these organizations, enhancing their incident response capabilities and strengthening cybersecurity.

1. Introduction

Phishing attacks using fake invoice image attachments have grown in prevalence. (Long, 2023). Fraudsters impersonate well-known companies and send emails containing fraudulent order confirmations or invoices. These phishing emails commonly instruct recipients to call a phone number to dispute a charge or address a fabricated issue. However, the true intent behind these calls is to extract valuable information from the victims, lead them to take further actions that could compromise their systems, and/or deceive them into making unauthorized payments.

Industry reports, such as ProofPoint's State of Phish report, indicate that tactics such as impersonating popular brands and incorporating phishing emails are still common. Alarming statistics reveal that 44% of people consider an email safe if it contains familiar branding, and there has been a significant increase in financial losses from successful phishing attacks. (Proofpoint, 2023).

Reportedly, cybercriminals have also used phishing techniques involving fake invoice image attachments, often called callback phishing attacks, combined with social engineering tactics. These attacks could lead to social engineering attacks and ransomware attacks. (Toulas, 2022). Detecting and countering such attacks presents notable challenges, particularly without automated tools. Although some commercial anti-phishing and email security solutions exist, their high costs may be impractical for organizations with limited resources. As cyber criminals continuously adapt their methods to evade traditional security measures, exploring practical approaches to combat these evolving threats is crucial. The relevance of this problem extends beyond individual organizations, given that falling victim to such frauds can lead to financial losses, compromised data, and long-term damage to an organization's reputation. (Palmer, 2022) The attacks also undermine trust in communications, online transactions, and the economy. Organizations with limited resources need to explore proper solutions that are cost-effective and tailored to their specific needs. This effort to find effective solutions may include considering open-source tools, local automation, or other approaches that provide similar functionalities at a fraction of the cost.

Bayarmaa Havel, bamaa2000@yahoo.com

Local automation involves implementing automated processes and tools within an organization's infrastructure rather than relying on external solutions. This approach brings benefits such as customizing automation processes for smoother and more efficient operations. Moreover, sensitive data remains within the organization's network, reducing exposure to security risks from external providers. Organizations can avoid potential disruptions and vulnerabilities by reducing reliance on third-party services. Moreover, running processes in the organization's local network allows for faster network response times and quicker decision-making. Local automation also provides the flexibility to innovate and tailor automation tools to address specific goals and emerging threats. It provides cost efficiencies by minimizing recurring charges for external service subscriptions. Furthermore, organizations can ensure better regulatory compliance and control over data governance. The seamless integration of local automation with existing systems allows for streamlined workflows. This integration approach empowers organizations to optimize processes, enhance cybersecurity, and achieve greater operational efficiency in their environment.

This study explores a localized, automated approach using native and open-source tools to enhance incident response processes and strengthen defenses against callback phishing attacks that use fake invoice image attachments. The study evaluates the effectiveness and feasibility of automated techniques in reducing response times at various phases of the incident response lifecycle and compares them with traditional manual methods.

1.1. Incident Response Process

When dealing with phishing emails that contain fake invoice image attachments, it is crucial to adhere to industry-recommended guidelines. The National Institute of Standards and Technology (NIST) offers comprehensive recommendations on Cybersecurity Incident Management and Response in its 'Computer Security Incident Handling Guide' (SP 800-61 Rev. 2). (Cichonski, Millar, & Scarfone, 2012). According to NIST, the incident response process consists of five phases: preparation, detection and analysis, containment, eradication, recovery, and post-incident activities, as shown in Figure 1 below. This study will focus on detection & analysis and containment &

eradication phases within the NIST guidelines. These stages are essential for identifying, detecting, and deleting phishing emails with fake invoice image attachments.

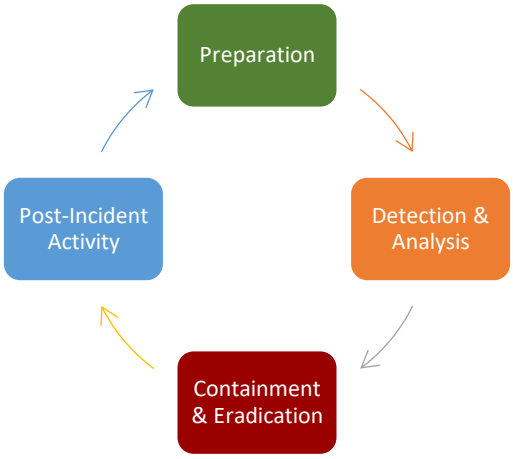

Figure 1. Incident Response Life Cycle

1.2. Unveiling the Power of OCR: Enhancing Email Security

Optical Character Recognition (OCR) is a technique for extracting text from images (Clements, 2023) to create editable documents from existing paper or image files. The OCR process involves identifying basic edge patterns using a simple edge detection technique and comparing them with predefined character templates. The technique is a specialized subfield within image recognition. It is widely used for data entry tasks when dealing with printed documents such as bank statements, invoices, resumes, business cards, and checks (Clements, 2023). OCR technology dates to the 1970s, when it was widely used in commercial applications. Notably, in the early 1970s, Recognition Equipment, Inc., based in Dallas, Texas, developed a high-speed system specifically for reading credit card receipts from gasoline purchases (Microscan, 2011).

In cybersecurity, OCR technology has also been effectively utilized. For example, InQuest Deep File Inspection (DFI) leverages machine vision and OCR to identify social engineering elements in phishing lures. (Long, 2023). After OCR extracts text from an image, it can use predefined or user-defined threat detection signatures to identify specific text patterns in the image

By leveraging OCR, organizations can strengthen their ability to identify and thwart phishing attempts, improving email security posture. OCR allows for the extraction and scrutiny of text, enabling further analysis and comparison with known patterns of fake invoices, aiding in identifying discrepancies and suspicious content that may indicate a phishing attempt.

2. Research Method

The research method employed in this study encompasses the critical components of an environment, tool creation, and attack detection.

2.1. Lab Environment

A controlled laboratory environment is set up to simulate real-world scenarios and facilitate the development and testing of the proposed solution. This controlled setting allows for systematic experimentation and evaluation of the tool's effectiveness in detecting phishing attacks using fake invoice image attachments. This research does not include detailed instructions for setting up the lab environment. However, readers can access the step-by-step instructions from the following GitHub repository: https://github.com/bamaa2000/Supplemental-Materials.

2.1.1. Computing Resources

The researcher chose the selected components for the lab environment in this research project to create a realistic and controlled simulation of a network environment. Each part serves a specific purpose to enable the study of phishing attacks with fake invoice image attachments effectively:

Host Windows 11 Machine with Hyper-V: This machine is the primary host for virtual machines, providing the necessary virtualization capabilities. Virtual machines allow researchers to create isolated and replicable environments for testing and analysis without affecting the underlying physical system.

Exchange Server 2016 (VM): Setting up a virtual machine running Exchange Server 2016 enables researchers to simulate an email server environment. The Exchange Server is essential as it is a typical email infrastructure commonly targeted in phishing attacks.

The server handles email communications in a lab environment, allowing researchers to analyze and gain insight into the delivery and handling of phishing emails.

Active Directory Domain Controller, DNS Server Windows Server 2016 (VM): Active Directory, Domain Controller, DNS Server Windows Server 2016 (VM): This virtual machine acts as an Active Directory domain controller and DNS server. Active Directory provides centralized user authentication and domain management, essential for managing user accounts, groups, and access rights in a lab environment. DNS resolution services are critical for routing email traffic and resolving domain names for emails and their attachments.

Image Analysis Host - Windows 10 Client Machine (VM): The image analysis host is a virtual machine running Windows 10, which serves as the main analysis host for this research project. Its primary purpose is to analyze incoming emails' content and attachments thoroughly. The host was equipped with an Outlook client and various analysis tools, including PowerShell scripts and Task Scheduler, allowing researchers to identify potentially malicious attachments. The dedicated nature of this host ensured that it was only used to run tools created for research purposes, allowing for focused and accurate analysis of phishing attacks with fake invoice image attachments.

Windows 10 Client Machine for Sending Fake Invoice Emails (VM): Another virtual machine running Windows 10 is solely responsible for sending fake invoice emails. This machine simulates the behavior of a compromised machine or an attacker sending fraudulent emails to identify potentially malicious attachments effectively. By using these components in the lab environment, researchers can conduct controlled experiments, analyze various phishing attack scenarios, and explore the effectiveness of the proposed localized automation approach in detecting and mitigating phishing attacks with fake invoice image attachments. This setup allows for targeted research and enables researchers to draw conclusions and develop practical solutions to strengthen cybersecurity defenses against such threats.

2.1.2. Lab Topology

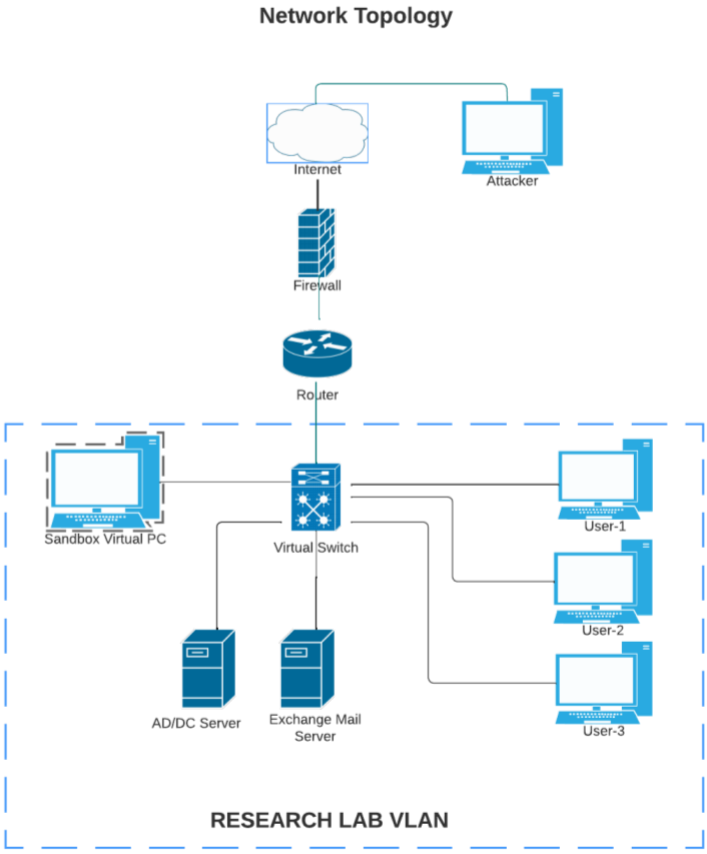

Figure 2 Diagram of the Lab Environment

The lab topology encompasses the virtualized environment, hosting and interconnecting virtual machines within the host machine. These virtual machines communicate with each other, and the external network is needed to simulate real-world scenarios and test the effectiveness of the detection and incident response mechanisms.

The lab topology allows for the simulation and analysis of various scenarios related to phishing attacks that exploit fake invoice image attachments. It provides a controlled environment where researchers can examine the behavior of these attacks, develop detection methods, and test the effectiveness of the incident response process.

2.1.3. Lab Configuration

The researcher took the following steps to set up the lab environment:

a. Active Directory Domain Controller: Installed and configured a virtual machine as an Active Directory Domain Controller to manage user accounts and domain-related services. (Flouds, Hardwood, & Ardolf, n.d.)

b. Active Directory User Accounts: Created user accounts in the Active Directory domain to simulate the presence of employees and users in the lab environment.

c. DNS Server: Configured a virtual machine as a DNS server to provide domain name resolution services in the lab environment.

d. Domain Name Registration: Registered a domain name, "[TestDomain.com]," through a domain registration service to enable communication in the lab environment.

e. Exchange Server: Installed and configured a virtual machine as an Exchange Server (Iyengar, Borsecnik, & Davis, Create a new exchange server self-signed certificate, n.d.) to handle email communication and mailbox management.

f. MX Server: Set up the lab environment to route incoming email traffic through the Exchange Server by configuring a Mail Exchange (MX) record for the registered domain. (AuthSMTP & GetOnline, n.d.), (NetDorm & DNSExit.com, n.d)

g. Email Address Assignment: Associated email addresses with the user accounts created in the Active Directory to enable email communication in the lab environment.

h. Mailbox Creation: Set up individual mailboxes for each user account to simulate the presence of email accounts associated with the registered domain. (Iyengar, Borsecnik, & Coulter, Manage user mailboxes, 2023)

i. Outlook Email on Image Analysis Host: Configure the Outlook email client on the image analysis machine to help analyze incoming emails and attachments. Also created a dedicated email account ORCAnalyst@TestDomain[.]com on the Outlook client, allowing the inspection of incoming emails and their associated attachments.

j. Gmail Account for simulated Phishing: Created a Gmail account to impersonate an attacker or a compromised computer to send fake invoice emails to a lab environment.

Following these steps and configuring the mentioned devices and hosts, the researcher set up a comprehensive lab environment to help research, detect, and analyze phishing attacks that exploit fake invoice image attachments.

2.2. Development of the Lightweight Script

The researcher created a customized script to detect and analyze phishing attacks that exploit fake invoice image attachments. This script automates the incident response process, which helps identify and mitigate fraudulent invoicing. During the tool's development, the researcher utilized existing frameworks, algorithms, and techniques to implement innovative approaches that enhance fraud detection capabilities. The developed tool undergoes rigorous testing in the lab environment to assess performance, accuracy, and reliability. The researcher simulates various phishing attack scenarios, including diverse types of fake invoice image attachments, to evaluate the tool's ability to detect and remove potential threats accurately. The testing phase involves an analysis of the tool's detection rates, false positive rates, and overall effectiveness in identifying phishing attacks. The small-scale automated workflow depicted in Figure 3 provides an

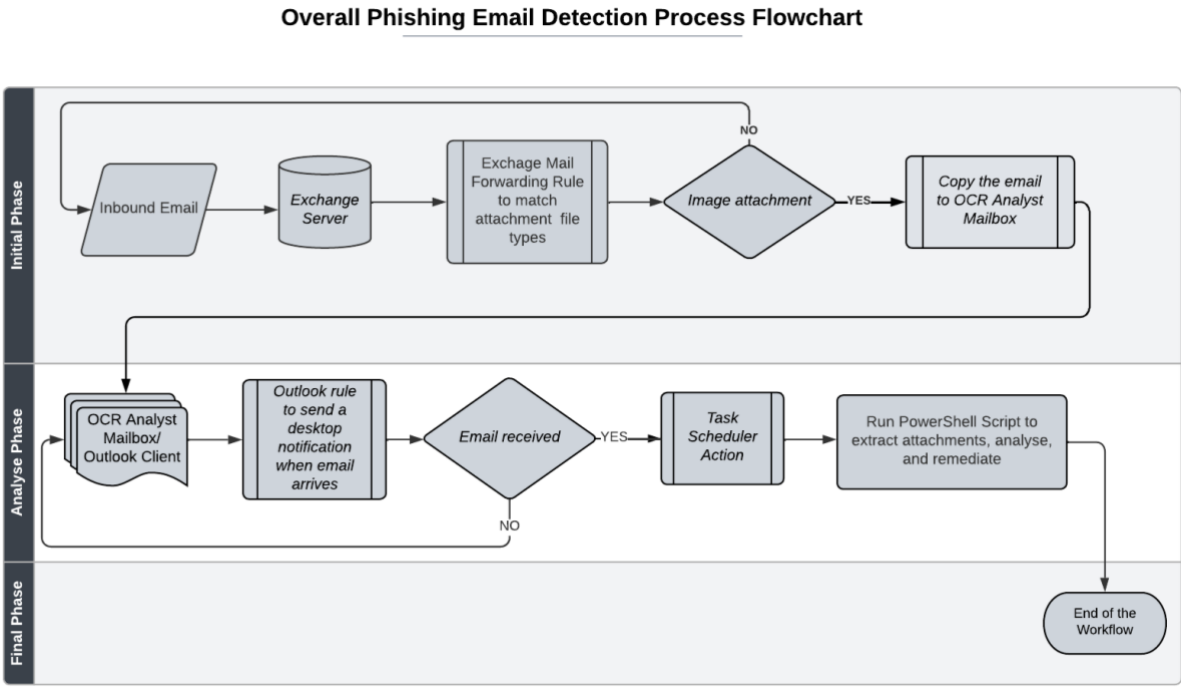

Figure 3 Detection and Eradication Flowchart

overview of the incident response process for phishing attacks with fake invoice image attachments.

This workflow incorporates various steps and components to streamline such threats' detection, analysis, and eradication. It highlights integrating automation tools, such as PowerShell scripts and OCR modules, to enhance detection, analysis, and response capabilities, ultimately contributing to efficient incident management. The detailed explanation of each step in the workflow shown in Figure 2 is provided below:

1. Phishing Email Detection: Emails suspected of phishing with fake invoice image attachments are identified using various detection mechanisms, such as email filters or user reports. These mechanisms help to identify potentially malicious emails that need further scrutiny.
2. Email Extraction: Relevant emails flagged as potential phishing attacks are extracted from the email repository for further analysis. Automation tools or scripts may be used to retrieve these emails based on predefined criteria, making the process more efficient and consistent.
3. Attachment Extraction: Attachments are extracted from the identified phishing emails. Automated techniques, such as PowerShell scripts, are employed to retrieve and save these attachments to a local directory for further processing.
4. Optical Character Recognition (OCR): The saved fake invoice images undergo OCR processing to convert into machine-readable text. OCR tools or modules, such as the PsOcr PowerShell module, extract text content from the images. This allows the script to analyze the content within the fake invoice images.
5. Text Analysis: The extracted text from the fake invoice images is analyzed for potential indicators of compromise (IOCs) or fraudulent content. Predefined IOCs or string patterns (Meskauskas, 2023), described in Appendix B, are compared against the extracted text to identify any matches or suspicious patterns that may indicate a phishing attack.
6. Threat Detection and Flagging: The email is flagged as a potential phishing attack if the extracted text contains matched IOCs or suspicious patterns. Flagging

mechanisms, such as alerts, notifications, or categorization, are employed to highlight these potential threats for further investigation by incident response teams.

7. Eradication and Reporting: Identified phishing emails are removed from user inboxes or quarantined to prevent further harm. Eradication operations can be automated through compliance search and removal processes, minimizing users' exposure to phishing attacks. Additionally, the tracking process captures relevant information about eradicated emails and helps post-incident analysis and reporting.

This workflow, involving a series of automated steps, allows for systematic and efficient incident response phases when dealing with phishing attacks that exploit fake invoice image attachments. The incident response team can detect, analyze, and mitigate potential threats using automation tools and advanced technologies such as OCR, contributing to a more secure and resilient environment.

2.2.1. Exchange Mail Flow Rule

Mail flow rules are integral to effective email management and security within organizations. These rules empower administrators to define specific conditions and actions for managing incoming and outgoing emails, offering automation, improved security, and streamlined workflow. In the context of developing an efficient workflow, the creation of a mail flow rule becomes a crucial step.

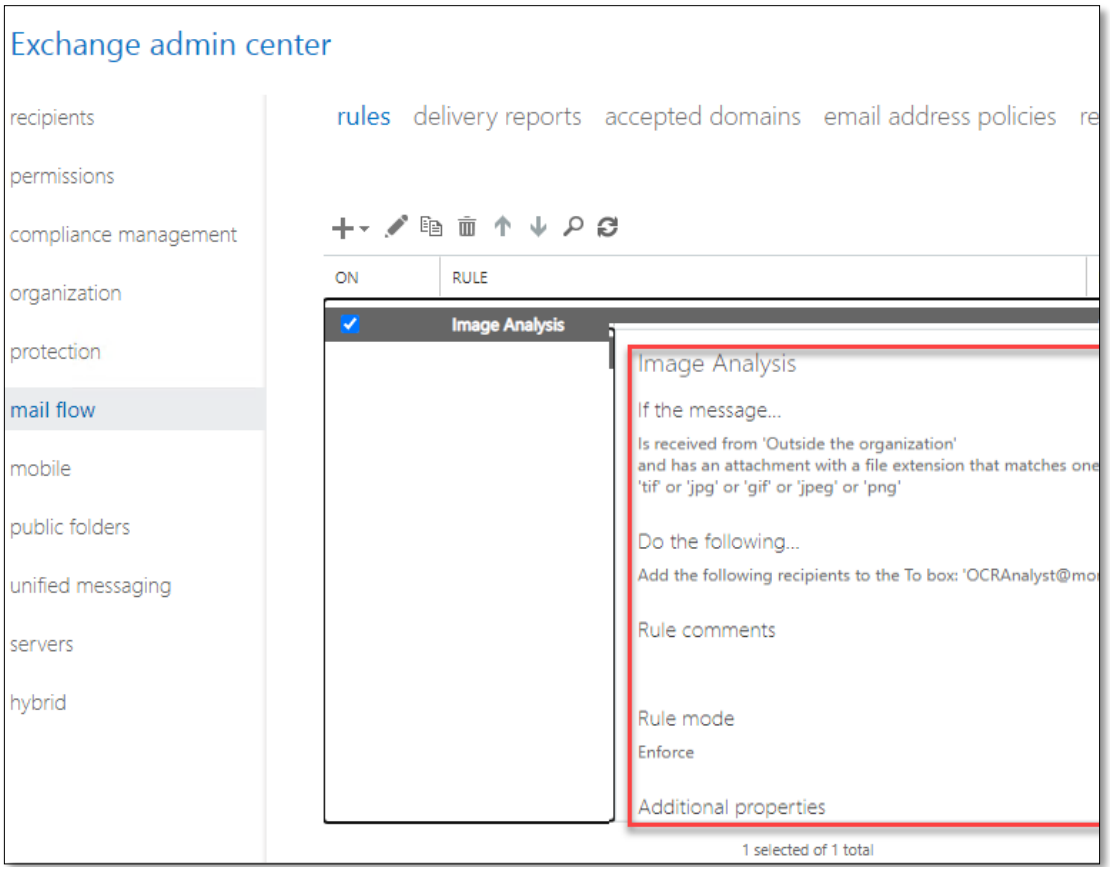

Figure 4 Exchange Mail Flow Rule

Copying emails with image attachments from external senders to the designated email account named OCR Analyst's (as mentioned in section 2.1.3) becomes possible by configuring a mail flow rule in the Exchange Server. This rule allows for the prompt capture of potentially malicious emails, ensuring a timely analysis and response. The process of creating a mail flow rule includes accessing the Exchange Control Panel (ECP), navigating to the "Mail Flow" and "Rules" sections, and configuring the rule with specific conditions and actions. The screenshot in Figure 4 illustrates the mail flow rule in the exchange admin center.

While the detailed steps to configure the rule are beyond the scope of this research paper, readers can find detailed step-by-step instructions in the researcher's GitHub repository https://github.com/bamaa2000/Supplemental-Materials/blob/master/Exchange_MailFlow_Rule.docx.

The mail flow rule detects and copies emails containing image attachments with specific file extensions, including TIF, JPG, GIF, JPEG, and PNG. This targeted

approach enables automated processing and analysis of image attachments, which is particularly relevant in detecting and analyzing phishing attacks that exploit fake invoice images. By focusing on these file types, the rule plays a pivotal role in efficiently managing and evaluating the content of image attachments, thereby aiding in identifying and mitigating fraudulent activities.

Organizations can tailor and fine-tune mail flow rules based on their requirements and security protocols. These rules can be adjusted by modifying conditions and actions to align with email management strategies and respond to each organization's distinct threat landscape. The research's core objective is to offer specialized insights and suggestions addressing the identification of fake invoice image phishing and associated fraudulent activities. While the study focuses on specific image types, organizations should embrace a holistic stance toward email security. Continual evaluation and updates to mail flow rules are vital to staying resilient against the ever-changing landscape of threats and attack vectors.

2.2.2. Extract email attachments

A PowerShell script was employed to automate the extraction of email attachments and address the challenges associated with manual retrieval. (Davis, 2019) Instructions on downloading attachments from Outlook using PowerShell guided the implementation of the script.

The PowerShell script successfully retrieves attachments from emails and saves them to a local directory. The script leverages the Outlook application and MAPI namespace to access email data, load emails from the designated subfolder, iterate through each email, and process the attachments. It modifies the file names of the attachments by inserting a timestamp, ensuring uniqueness, and providing a chronological order for the saved files.

Setting the Folder Path: The script includes a line that sets the $olFolderPath variable, specifying the target subfolder from which the emails are retrieved. This step allows for the selection of the desired email repository.

```
> $olFolderPath = "\\OCRAnalyst@[TestDomain.com]\Inbox\Quarantine"
$filePath = "//Sanbox/Users/ExchangeAdmin/Desktop/Attachments/"
```

Retrieving and Processing Emails: The script utilizes the Outlook application and MAPI namespace to access the emails. By creating an Outlook application object and accessing the MAPI namespace, the script sets up the interface to interact with email data.

```
$outlook = new-object -com outlook.application;
$mapi = $outlook.GetNameSpace("MAPI");
# set the Inbox folder id
$olDefaultFolderInbox = 6
$inbox = $mapi.GetDefaultFolder($olDefaultFolderInbox)
# access the target subfolder
$olTargetFolder = $inbox.Folders | Where-Object { $_.FolderPath -eq $olFolderPath }
```

Load Email: This script uses $olTargetFolder.olTargetFolder. The Items command loads emails from a specified subfolder. This step ensures that the subsequent processing focuses on the relevant email content.

```
# load emails
$emails = $olTargetFolder.Items
```

Implement a loop structure to iterate over each email in a subfolder. This loop allows operations to be performed on individual emails, such as extracting attachments.

Attachment Processing and Saving: This script contains code to modify the filename of the attachment and save the attachment to a local directory.

```
$emails = $olTargetFolder.Items
$count = $emails.count
$count = 0
foreach ($email in $emails) {
$email
$count = $count + 1
```

Modifying File Names: This script contains code to insert a timestamp into the attachment's filename. By using the received timestamp of each email, the modified file names ensure uniqueness and provide a chronological order for the saved attachments.

```
$timestamp = $email.ReceivedTime.ToString("yyyyMMddhhmmss")
```

Saving Attachments: The script uses the "saveasfile" method to save the attachments. By specifying the target directory with the $filePath variable, the script ensures that the attachment is saved in the desired location.

```
$email.Attachments | foreach {
       $fileName = $_.FileName
       $fileName = $fileName.Insert($fileName.IndexOf('.'),$timestamp)
       $_.saveasfile((Join-Path 'C:\Users\ExchangeAdmin\Desktop\Attachments\' $fileName))
}
```

The script retrieves the attachments from the specified subfolder, modifies the filenames, and saves them to the local directory. Implementation details can help readers replicate the process or adapt the script to their requirements.

2.2.3. Identify Fake Invoice with OCR

In the research, a PowerShell module developed by Dr. Tobias Weltner was used to access the OCR (Optical Character Recognition) functionality built into Windows 10. This module, called PsOcr, allows PowerShell to extract text from images. (Weltner, 2021) Install the module with the following command:

```
Install-Module -Name PsOcr -Scope CurrentUser
```

The following script extracts text from the saved images:

```
> foreach ($file in Get-ChildItem "C:\Users\ExchangeAdmin\Desktop\Attachments\") {
    $OCRText = convert-psoimagetotext -path (Join-Path -C:\Users\ExchangeAdmin\Desktop\Attachments\' $file)
    write-output $OCRText >> C:\Users\ExchangeAdmin\Desktop\Attachments\ImageToText.txt
    $IOC = get-content -path C:\Users\ExchangeAdmin\Desktop\Strings\StringsToMatch.txt
    # Search IOCs from the ImageToText file
    foreach ($string in $IOC) {
```

```
        # The code to process each IOC goes here
        # (It's provided in another snippet)
    }
}
```

The script iterates through each file in the specified directory and uses the convert-psoimagetotext function from the PsOcr module to extract text from the images. The extracted text appends to a text file named "ImageToText.txt." Moreover, the script reads a list of strings to match stored in the "StringsToMatch.txt" file. It then searches for these strings in the extracted text from the images. Upon finding a match, the script outputs a message indicating that the text contains the matched string.

This approach enables automated text extraction from images using the Windows 10 built-in OCR functionality and the PsOcr PowerShell module. A predefined Indicator of Compromise (IoC) list detects fake invoices.

2.2.4. Remove Fake Invoice Phishing Emails

Eradication constitutes an essential phase in the incident response process. To enable the primary tool's execution, setting up a remote connection to the Exchange Server using the provided credentials is necessary. This action allows the tool to access and utilize the required Exchange Server cmdlets in the current PowerShell session. The following script segment is a prerequisite for the primary tool and outlines the steps to set up this remote session using PowerShell.

To set up the remote session, the script includes the following steps:

1. Add Exchange Management PowerShell snap-in (Add-PSSnapin Microsoft.Exchange.Management.PowerShell.SnapIn;): This step ensures that the required cmdlets for managing the Exchange Server are available.
2. Set up a remote Session to Exchange-Server:
    a. Create a secure string object ($password=ConvertTo-SecureString 'SecretPassword' -AsPlainText -Force): This line converts the plain text password 'SecretPassword' into a secure string format, ensuring the confidentiality of the password.

b. Create a PowerShell credential object ($credential = New-Object System.Management.Automation.PSCredential ('exchangeadmin', $password)

   A credential object is created by providing the username ('exchangeadmin') and the secure password obtained from the previous step. This credential object will be used for authentication during the remote session.

c. Establishing the remote session (Davis & Raya, Connect to exchange servers using remote PowerShell, 2022) ($Session = New-PSSession -ConfigurationName Microsoft.Exchange -Connection Uri http://exchange-server/PowerShell/ -Authentication Kerberos -Credential $credential):
   This line creates a new PowerShell session ($Session) by specifying the configuration name, connection URI, authentication method, and credential object. The configuration name indicates the Exchange Server, the connection URI defines the PowerShell endpoint, Kerberos authentication is used, and the provided credentials are used for authentication.

d. Importing the session and enabling cmdlets (Import-pssession $session -DisableNameChecking -AllowClobber):
   This step imports the created PowerShell session ($Session) and enables the Exchange Server cmdlets in the current session. The -DisableNameChecking and -AllowClobber flags bypass name checking and allow overwriting of existing cmdlets if needed.

Following the steps outlined in the script sets up the necessary connectivity and permissions to interact with the Exchange Server and enable the execution of subsequent phases in the incident response workflow. Knowing the security implications of including user credentials in the script is essential. Since the research focuses on the feasibility of creating native automation tools, including credentials in the script may be acceptable. However, prioritizing security measures to protect sensitive information that includes limiting access to the script, encrypting its contents, implementing secure execution environments, and regularly reviewing and updating the script to address any security vulnerabilities that may arise. Adherence to these measures enhances the overall security

of the tool and minimizes the risk of unauthorized access or exposure, even within the research context.

The eradication of the identified fake invoice phishing emails involved utilizing the following PowerShell script:

```
       $subject = $email.Subject
       $from = $email.SenderEmailAddress
       $SenderName = $email.sendername
       $attachment = $email.Attachments  | select-object -expandproperty DisplayName
        $Search=New-ComplianceSearch -Name "Bait-Phishing" -ExchangeLocation All -
ContentMatchQuery "(attachment:$attachment)(from:$from)"
        start-ComplianceSearch -Identity $Search.Identity
        New-ComplianceSearchAction  -SearchName "Bait-Phishing"  -Purge
        remove-compliancesearch -identity "Bait-Phishing" -Confirm:$false
        #Track Removed Face-Invoice emails
        echo -------- $email.ReceivedTime "From: "$from "To: "$email.to $subject
$attachment >> "C:\Users\ExchangeAdmin\Desktop\Removed-Fake-Invoices.txt"
        exit
        remove-item C:\Users\ExchangeAdmin\Desktop\Attachments\*.*
        Echo "Removed a Fake Invoice"
```

The following list explains each step of the presented script.

1. Extraction of Email Details:
    - Get email subject with $email.Subject.
    - Obtain the sender's email address using $email.SenderEmailAddress.
    - Retrieve the sender's name with $email.sendername.
2. Identification of Attachments:
    - Select and expand email attachment display names using $email.Attachments | select-object -expandproperty DisplayName.
    - Store attachment display names in the $attachment variable.
3. Initiating Compliance Search (Davis C. , Remove-compliancesearch (exchangepowershell), 2023):

- Create a compliance search "[New Compliance Search Name]" with New-ComplianceSearch cmdlet, matching attachment, and sender info.
- Start the compliance search using start-ComplianceSearch -Identity $Search.Identity.
- Introduce a one-second delay using Start-Sleep -Seconds 1.

4. Purging Search Results:
    - Purge search results with New-ComplianceSearchAction -SearchName "[New Compliance Search Name]" -Purge.
5. Clean Up Compliance Search:
    - Remove the compliance search with remove-compliancesearch -identity "[New Compliance Search Name]" -Confirm:$false.
6. Tracking Removed Emails:
    - Append information about the removed email to "Removed-Fake-Invoices.txt" using the echo command. Include received time, sender, recipients, subject, and attachment.
7. Exiting the Script:
    - Use the exit command to end the script.
8. Removing Attachments and Providing Feedback:
    - Delete all files in the "Attachments" directory with remove-item C:\Users\ExchangeAdmin\Desktop\Attachments*.*.
    - Output the message "Removed a Fake Invoice" using echo.

The inclusion of this script illustrates its role in the detection and analysis, containment, and eradication phases of the incident response process. It effectively identifies and deletes fake invoice phishing emails, mitigating threats, and preventing further damage or unauthorized access. The entire script is available in Appendix A, or readers can download it from the researcher's GitHub repository at the following repository: https://github.com/bamaa2000/Supplemental-Materials/blob/master/Fake_Invoice_Phishing_Email_Detection_Removal.ps1.

Bayarmaa Havel, bamaa2000@yahoo.com

2.3. Automation

Automation is critical in streamlining the incident response process and reducing manual effort. In this study, a three-step automation technique was used to improve the effectiveness of the vital incident response phases:

2.3.1. Outlook Rule to Invoke a Display Notification

The researcher enabled an Outlook rule to generate display notifications when the account receives a new email to enhance the immediate visibility of incoming emails. This notification triggers the Task Scheduler to execute the PowerShell script discussed in Sections 2.2.2 – 2.2.4. Figure 5 shows an example of a display notification generated when an Outlook account receives a new email.

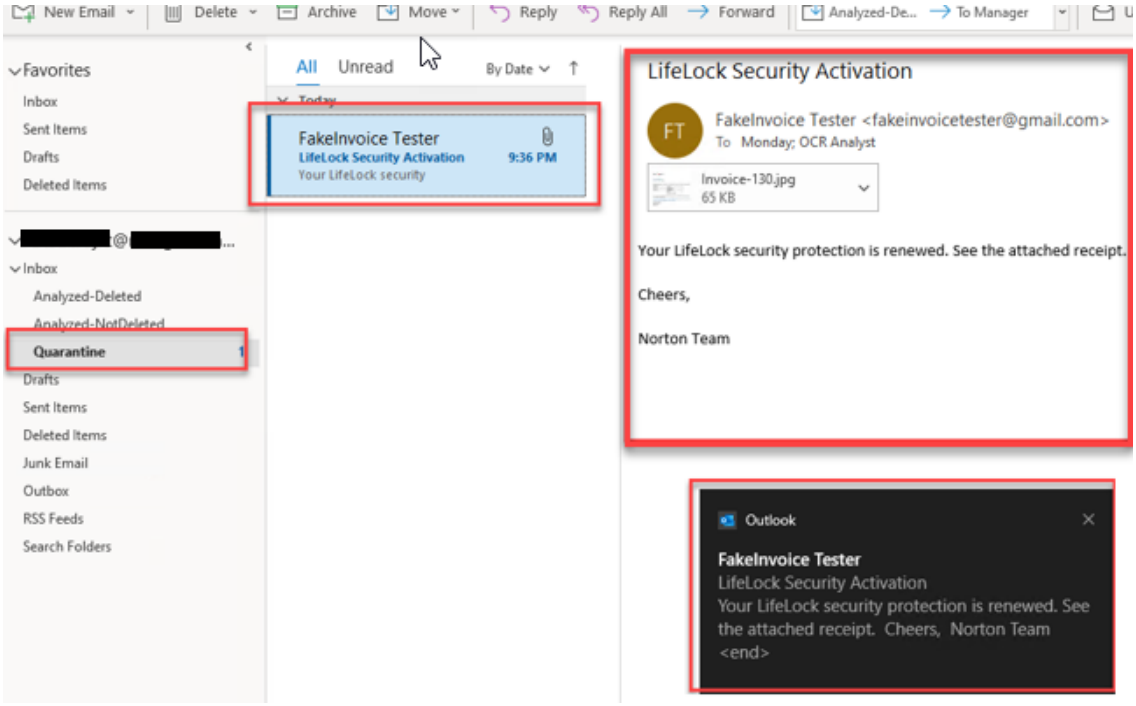

Figure 5 Sample Display Noticifation

2.3.2. Automating the PowerShell Script Execution with Task Scheduler

To automate the execution of PowerShell scripts at predefined intervals or in response to specific events, the researcher used Task Scheduler. Task Scheduler is a built-in Windows feature (Steven, Simpson, Radich, & Satran, 2023) that allows for scheduling

and automating various tasks on a system. In the case of the study, the researcher scheduled a task to run based on Event ID 10016, which is related to Distributed Component Object Model (DCOM) events (Deland-Han, Anna-Li, & Zou, 2021). This event triggers the execution of the PowerShell script, ensuring its automatic run whenever the specified event occurs. By leveraging the Task Scheduler, the incident response team can automate critical script execution, ensuring timely responses to security events or system conditions.

Figures 5-1, 5-2, and 5-3 illustrate examples of Task Scheduler configuration steps in a Windows 10 environment.

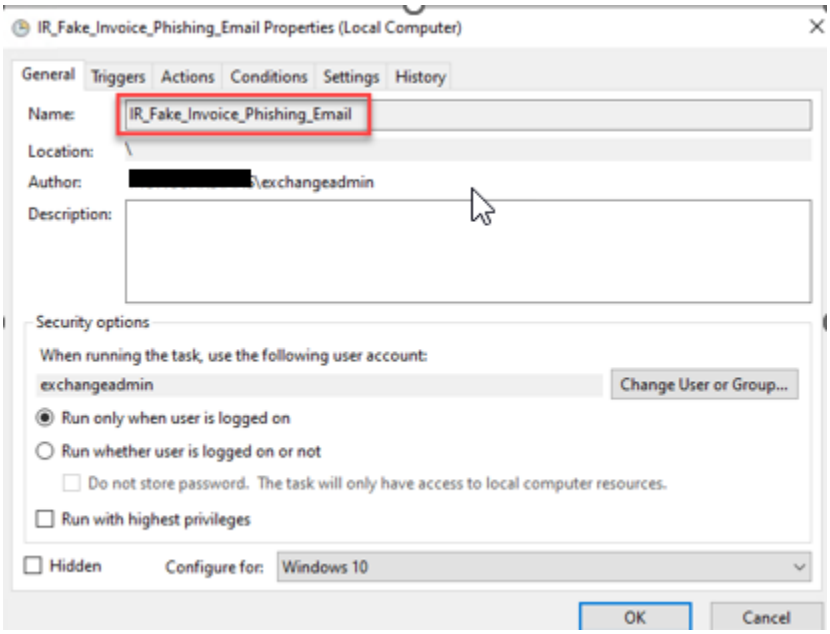

Figure 5-1 Sample Task Schedule Setting

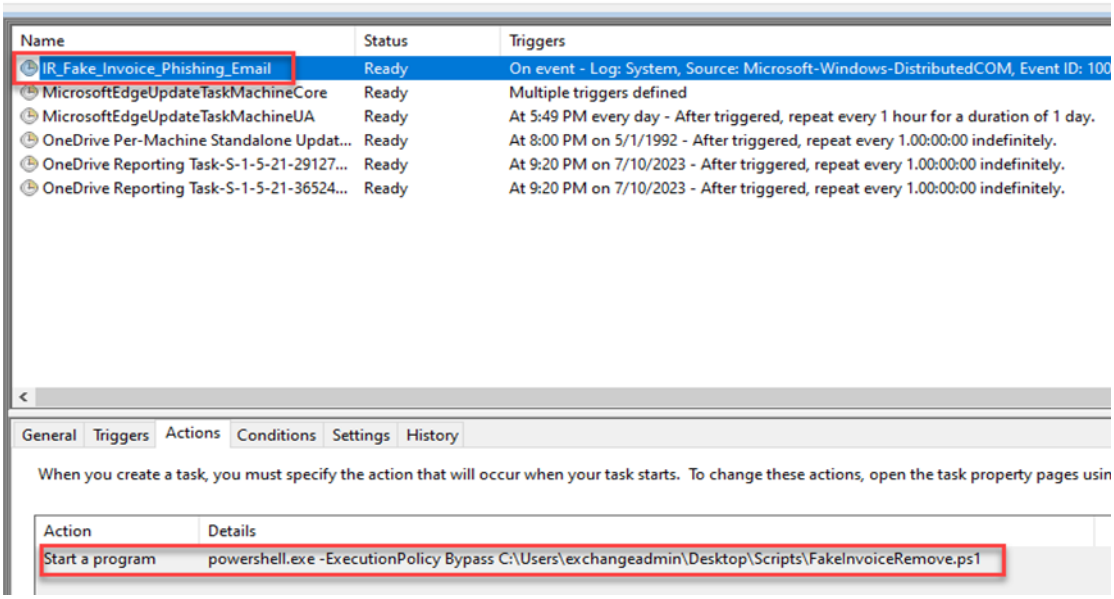

Figure 5-2 *Sample Task Schedule Setting*

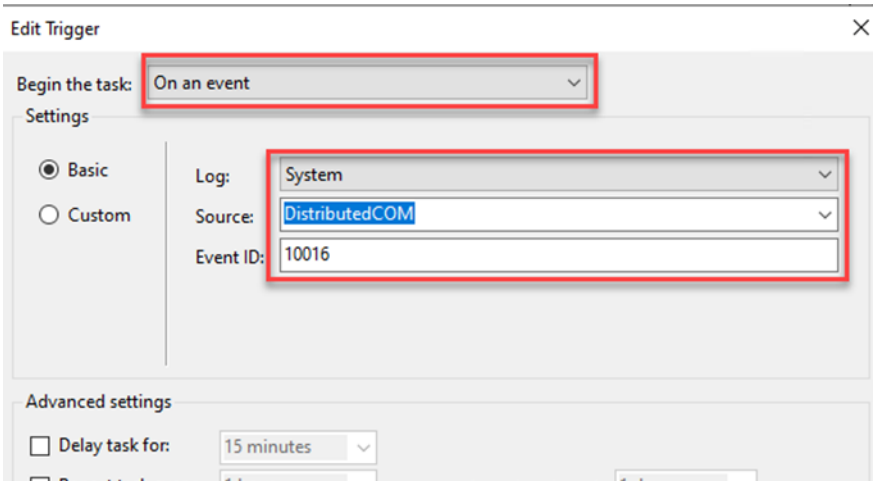

Figure 5-3 Sample Task Schedule Setting

These figures highlight how to create a task, specify the trigger conditions (such as Event ID 10016), and set up the PowerShell script as the action to execute.

Organizations can customize these configurations based on their specific needs and goals, allowing scheduling flexibility and automating tasks according to their incident response requirements.

2.3.3. PowerShell Script for Image Extraction, Analysis, and Purging

The researcher designed the PowerShell script to extract images from incoming emails, analyze the extracted images, and initiate purging if necessary. Task Scheduler triggers this script to perform these actions automatically. The script uses various PowerShell modules and functions to perform image extraction, analysis, and email purging tasks. Automating these processes empowers the incident response team to efficiently manage incoming emails, analyze potential threats, and take proper actions promptly.

By implementing this three-step automation technique, the research aims to streamline the incident response workflow, reduce manual efforts, and help faster and more efficient responses to security incidents.

3. Testing, Result, and Evaluation

3.1. Datasets and results

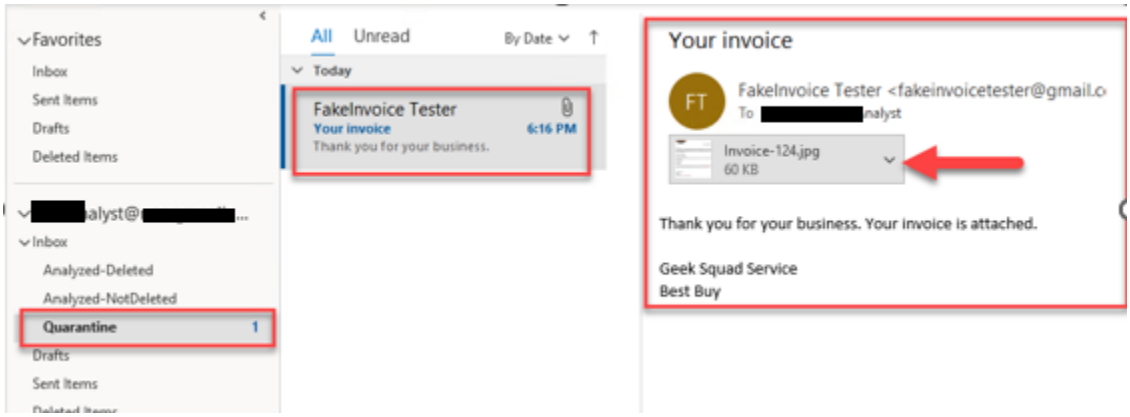

Figure 6 Sample incoming email message

Figure 6 shows an example of an incoming email message that an Exchange mail flow rule has replicated to an analyst account.

Figure 7 below (University, 2022) shows one of the samples of fake invoices used to evaluate the functionality of the developed tool. This image is a typical fake invoice often used in callback phishing attacks. The researcher used multiple samples to simulate real-world scenarios. Using these samples aimed to assess the tool's effectiveness in detecting and analyzing fraudulent invoices in realistic settings. Through testing with representative samples like this fake invoice, the research aimed to validate the tool's functionality and evaluate its capabilities in identifying and mitigating phishing attacks involving fake invoice images.

Figure 7 Samples Fake Invoice

Figure 7-1 displays the result of the OCR (Optical Character Recognition) analysis script that is executed to extract text from the image file attachment of an email. The script uses OCR techniques to convert the image's content into machine-readable text.

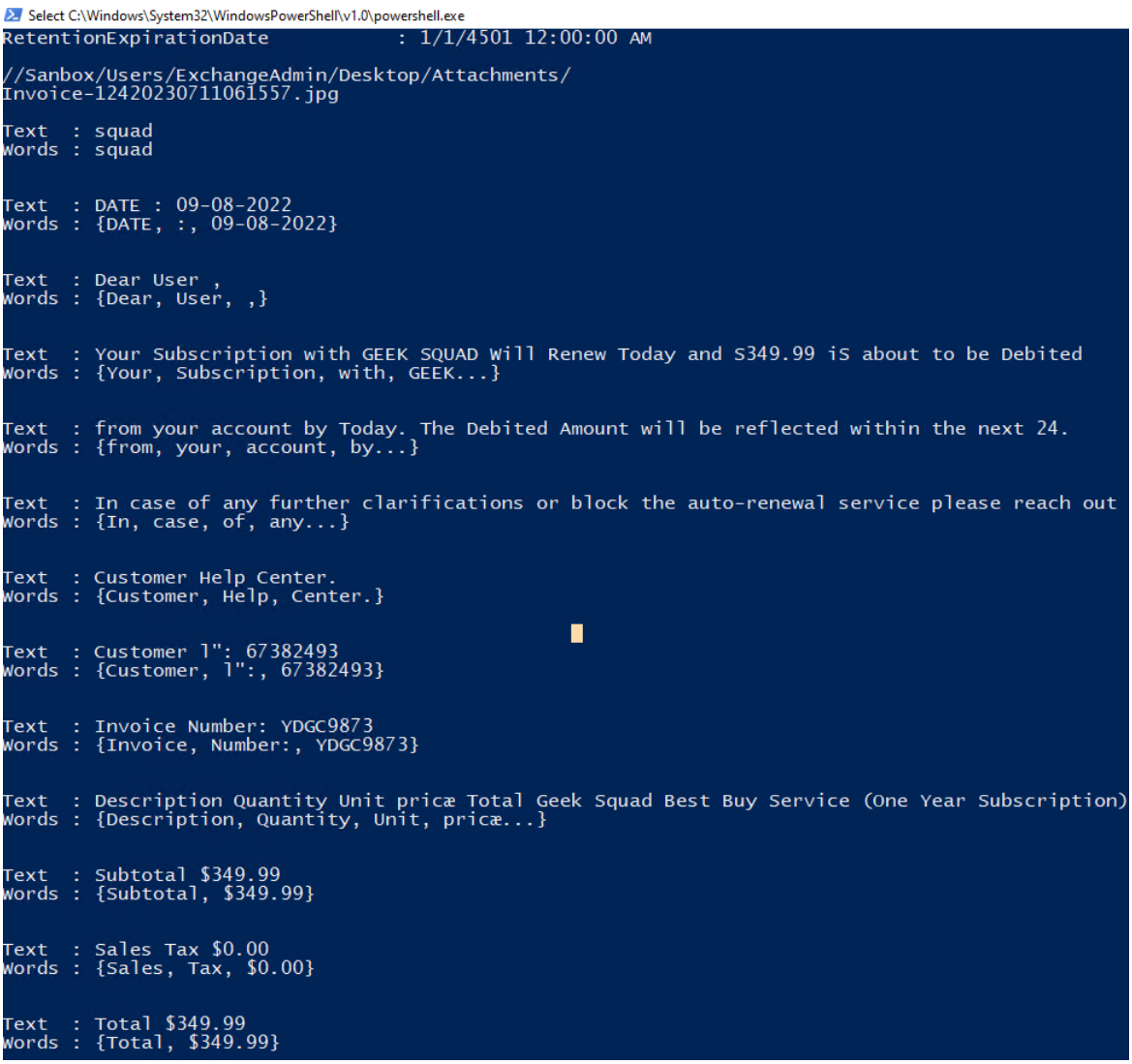

Figure 7-1: Result of OCR Analysis Script Extracting Text from Image File Attachment

Figure 7-2 below shows the results of an automated cleanup script that removed identified phishing emails from user inboxes. The script promptly removes potential threats, preventing further damage or unauthorized access. By automating the purging process, organizations can efficiently remove malicious emails and reduce the risk of

falling victim to phishing attacks. Automated cleanup scripts provide a proactive approach to email security that responds to and mitigates threats promptly.

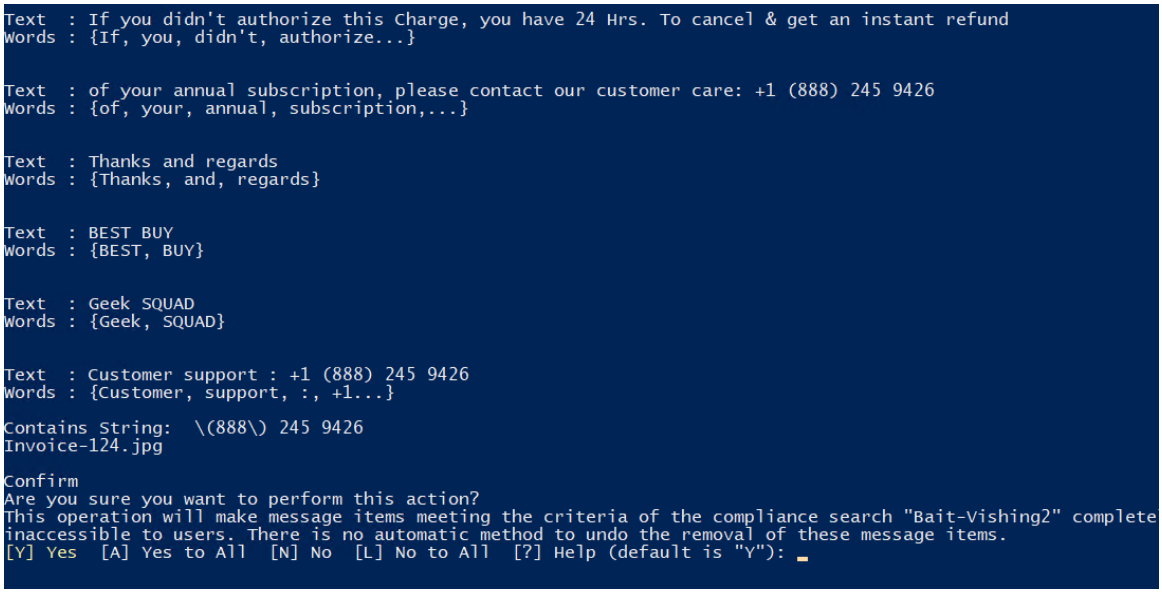

Figure 7-2: Result of OCR Analysis Script Extracting Text from Image File Attachment and Automated Purging

3.2. Comparative analysis and assessment of phishing email detection techniques.

Table 1 presents a comparative analysis and evaluation of manual and automated assessments using the OCR PowerShell Tool for phishing email detection techniques. Analysis time refers to an organization's duration to detect, evaluate, and respond. The goal is to minimize the analysis time, reducing false positives and false negative detections.

#	Dataset Used	Method proposed	Accuracy count	Analysis time (Min)
1.	Ten phishing emails with fake invoice Image attachments	Manual assessment	10	30
2.	Ten phishing emails with fake invoice Image attachments	Automated assessment with OCR PowerShell Tool	10	3
3.	Ten non-phishing with image attachments	Manual assessment	10	10
4.	Ten non-phishing with image attachments	Automated assessment with OCR PowerShell Tool	10	2

Table 1 C*omparative Analysis of Phishing and Non-Phishing Email Detection Methods*

These methods classify phishing emails with fake invoice image attachments and non-phishing emails with image attachments. The researcher reports the classification accuracy and time required for each method on different datasets:

In Dataset 1, the researcher manually assessed ten phishing emails with fake invoice image attachments. The classification accuracy was 100%, meaning all phishing emails were correctly identified. This manual assessment took 30 minutes.

Dataset 2 classified the same ten phishing emails using an automated tool with OCR PowerShell. The classification accuracy was 100%, correctly identifying ten out of ten phishing emails. The analysis time was reduced to 3 minutes compared to the manual assessment.

Dataset 3 consisted of ten non-phishing emails with image attachments, manually assessed, without automation tools. The accuracy was 100%, correctly identifying all non-phishing emails. The analysis time for this dataset was 10 minutes.

Finally, in Dataset 4, the same ten non-phishing emails were assessed using the automated tool with OCR PowerShell. The classification accuracy was 100%, correctly identifying 10 out of 10 non-phishing emails. The analysis time was reduced to 2 minutes compared to the manual assessment. Readers can access the sample datasets used in the simulated testing from the researcher's GitHub repository: https://github.com/bamaa2000/Supplemental-Materials/.

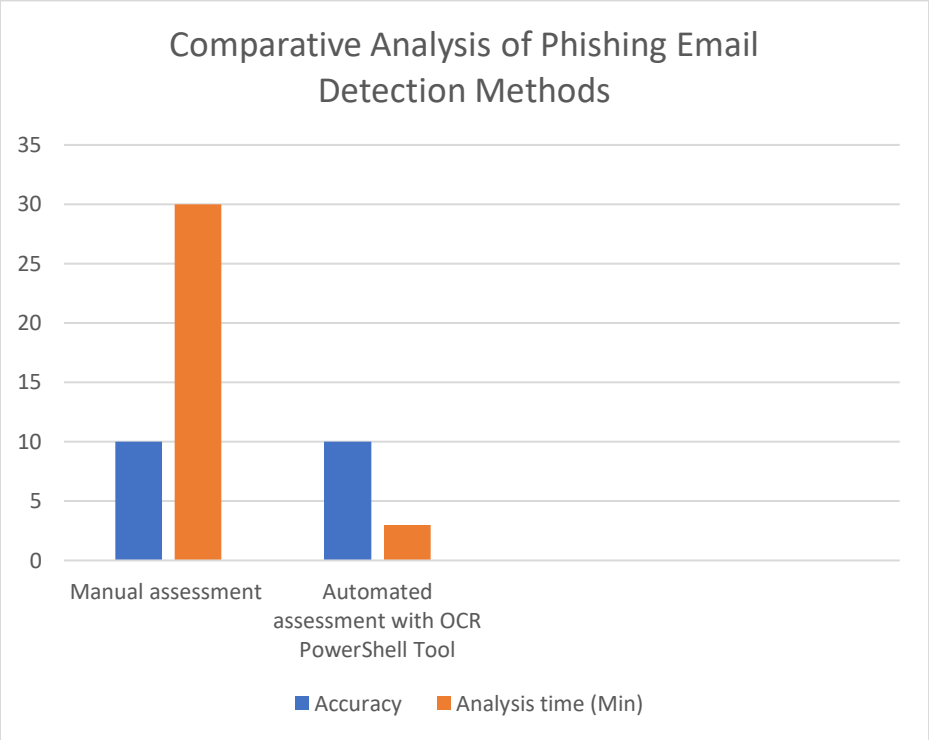

Figure 8 Comparative Analysis Chart

In Figure 8, the findings show that the automated assessment tool utilizing OCR PowerShell effectively reduces assessment time compared to the manual method.

4. Recommendations and Implications for Future Research

The subsequent section provides recommendations to enhance phishing email detection techniques and incident response strategies based on the research findings. Furthermore, it explores the implications of future discoveries to advance the research. These suggestions enable organizations to proactively combat phishing attacks that exploit fake invoice image attachments.

4.1. Key Recommendations Based on Research Findings

Organizations should consider developing and deploying local automation tools tailored to their specific needs and resources. These tools can enhance incident response processes by automating, detecting, and mitigating phishing attacks that exploit fake invoice image attachments. Phishing tactics constantly evolve, and attackers often modify their techniques to bypass existing security measures. By regularly updating the signatures used for phishing detection, cybersecurity systems can stay updated with the latest trends and patterns employed by phishing campaigns.

Optical Character Recognition (OCR) to extract text from the phishing email's images allows for identifying specific text strings commonly found in fraudulent invoices or social engineering attempts. These extracted text strings can serve as valuable indicators to create or refine signatures for detecting phishing attempts. By integrating OCR technology and maintaining a dynamic signature database, organizations can enhance their ability to recognize and block phishing emails that utilize fake invoice images as bait. This proactive approach is essential to avoid cyber threats and safeguard users from falling victim to phishing scams. (Smith, 2020).

Organizations can augment their incident response capabilities by integrating locally automated tools with third-party solutions. This integration allows enhanced threat intelligence, advanced analytics, and collaboration with external security providers.

4.2. Implications for Future Research

The research method employed in this study encompasses the key components of setting up a lab environment, creating a custom tool, and detecting phishing attacks. The

samples used in the research were obtained from publicly available sources on the internet, as there was a lack of access to real-life fake invoice emails. Therefore, the researcher recommends conducting further research using real-life datasets that organizations receive to conduct more comprehensive studies.

The PowerShell OCR module used in this research is based on the Windows native tool but has certain limitations. These limitations include limited language support, reliance on local computing power, and accuracy issues. Additionally, organizations may need to work on integrating the tool with custom applications and address the absence of regular updates.

The research utilized ten fake invoices due to limited resources. It is important to note that the accuracy results of the testing may be affected by the small dataset. Therefore, the researcher recommends conducting tests on broader datasets to obtain more reliable results.

While the Windows native OCR tool offers basic functionality, it may not fully meet the organization's requirements seeking advanced capabilities, higher accuracy, or broader integration support. To overcome these limitations, researchers can explore cloud-based OCR solutions such as Azure Cognitive Services, that offer more extensive features and flexibility. Moreover, researchers should investigate automating incident response processes beyond the detection, analysis, and containment phases. A key focus should be applying automation to the Post-Incident Activity phase to optimize response times and enhance incident handling. This idea may include integrating email security gateways to block malicious sender email addresses and malicious file hashes, improving the overall incident response.

5. Conclusion

Phishing attacks with fake invoice image attachments present significant risks to individuals and organizations. This research focuses on developing a localized script to bolster a part of incident response lifecycles and defend against such attacks. Organizations can enhance their incident response capabilities by employing automation techniques like extracting email attachments, using OCR technology to identify fake invoices, and removing phishing emails. This solution offers free and accessible defense mechanisms, making them particularly valuable for resource-constrained organizations with privacy concerns about exposing data to cloud environments. The research findings and recommendations provide useful insights for organizations looking to strengthen their cybersecurity resilience and combat fraudulent invoicing scams.

Bayarmaa Havel, bamaa2000@yahoo.com

References

AuthSMTP, & GetOnline. (n.d., n.d. n.d.). *Microsoft Exchange 2016 - alternative port*. Retrieved May 2, 2023, from Exchange 2016 Smarthost Connector - Change SMTP port: https://www.authsmtp.com/exchange-2016/exchange2016_alternative_port.html

Cichonski, P., Millar, T., & Scarfone, K. (2012, August). *NIST Special Publication 800-61 Revision 2*. Retrieved from NIST: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf

Clements, J. (2023, July 18). *Managed Outsource Solutions*. Retrieved from ttps://www.managedoutsource.com/blog/how-ocr-and-machine-learning-features-changing-banking-industry/

Davis, C. (2023, May 2). *Remove-compliancesearch (exchangepowershell)*. Retrieved from (ExchangePowerShell) | Microsoft Learn.

Davis, C., & Raya, T. (2022, March 30). *Connect to exchange servers using remote PowerShell*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/powershell/exchange/connect-to-exchange-servers-using-remote-powershell?view=exchange-ps

Davis, D. (2019, September 27). *M365ROCKS*. Retrieved from M365ROCKS: https://www.danny-davis.com/blog/2019/9/27/download-attachements-from-outlook-with-powershell

Deland-Han, Anna-Li, & Zou, L. (2021, November 30). *Event ID 10016 is logged in windows - windows client*. Retrieved from https://learn.microsoft.com/en-us/troubleshoot/windows-client/application-management/event-10016-logged-when-accessing-dcom

Flouds, I., Hardwood, R., & Ardolf, D. (n.d., n.d. n.d.). *Install Active Directory Domain Services (level 100)*. Retrieved from Microsoft Learn:

https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/deploy/install-active-directory-domain-services--level-100-

Iyengar, A., Borsecnik, J., & Coulter, D. (2023, February 21). *Manage user mailboxes*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/exchange/architecture/mailbox-servers/mailbox-servers?view=exchserver-2019

Iyengar, A., Borsecnik, J., & Davis, C. (n.d., n.d. n.d.). *Create a new exchange server self-signed certificate*. Retrieved February 21, 2023, from Microsoft Learn: https://learn.microsoft.com/en-us/exchange/architecture/client-access/create-self-signed-certificates?view=exchserver-2019

Long, J. (2023, June 1). *Intego.com*. Retrieved from Fake invoice scams: Norton, McAfee, PayPal, and more: https://www.intego.com/mac-security-blog/fake-invoice-scams-norton-mcafee-paypal-and-more/

Meskauskas, T. (2023). *Geek Squad email scam*. Retrieved from Geek Squad Email Scam - Removal and recovery steps (updated): https://www.pcrisk.com/removal-guides/23907-geek-squad-email-scam

Microscan. (2011, February 11). *microscan.com*. Retrieved from Understanding optical character recognition - files.microscan.com: https://files./industrysolutions/ocr_whitepaper.pdf

NetDorm, I., & DNSExit.com. (n.d, n.d n.d). *Mail redirect service - GO AROUND INBOUND SMTP port 25 blocked problem*. Retrieved May 4, 2023, from SMTP Port 25 blocked workaround - Email Redirection Service: https://dnsexit.com/services/email-redirect-smtp-port-25-blocked/

OpenAI, O. (n.d.). *Chatgpt. GPT-3.5 (Version 3.5)*. https://openai.com/chatGPT

Palmer, D. (2022, November 22). *Warning: This scam starts with a fake invoice. it could end with crooks stealing your data*. Retrieved from ZDNET:

https://www.zdnet.com/article/warning-this-scam-starts-with-a-fake-invoice-it-could-end-with-crooks-stealing-your-data/

Proofpoint. (2023, February 28). *Proofpoint*. Retrieved from 2023 state of the Phish Report - Phishing Stats & Trends: Proofpoint us: https://www.proofpoint.com/us/resources/threat-reports/state-of-phish

Smith, J. (2020, May 12). *Detecting coercive lures with OCR*. Retrieved from InQuest.com: https://inquest.net/blog/2020/05/13/detecting-coercive-lures-ocr

Steven, W., Simpson, D., Radich, Q., & Satran, M. (2023, February 8). *Task Scheduler for developers*. Retrieved from Win32 apps | Microsoft Learn: https://learn.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page

Toulas, B. (2022, October 07). *BleepingComputer*. Retrieved from Callback phishing attacks evolve their social engineering tactics: https://www.bleepingcomputer.com/news/security/callback-phishing-attacks-evolve-their-social-engineering-tactics/

University, M. (2022, September 9). *Latest phishing scam impersonates Geek Squad.* Retrieved from Miami University: https://miamioh.edu/_files/images/it-services/news-articles/2022/09/geek-squad-scam.jpg

Weltner, T. (2021, January 15). *TOBIASPSP/PSOCR: Home of the PowerShell module "Psocr" which uses the native Windows 10 OCR engine to convert image files to text*. Retrieved from GitHub: https://github.com/TobiasPSP/PsOcr

Appendix A

```
# Add-PSSnapin Microsoft.Exchange.Management.PowerShell.SnapIn;
# Open a remote Session to Exchange-Server

$password = ConvertTo-SecureString '[Password]' -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential
('[exchangeadminusername]', $password)
$Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri
http://exchange-server/PowerShell/ -Authentication Kerberos -Credential $credential
Import-PSSession $session -DisableNameChecking -AllowClobber

# Phase-1: Copy all emails with image attachments to a designated mailbox
# This Phase needs to be done in Exchange Server

# Phase-2: Extract attachments from the designated mailbox
# link to the folder

$olFolderPath = "\\[EmailAddress]\Inbox\[Folder Name]"

# Set the location to a temporary file

$filePath = "//[FilePath]/"

# Use MAPI namespace

$outlook = New-Object -ComObject outlook.application;
$mapi = $outlook.GetNameSpace("MAPI");

# set the Inbox folder id

$olDefaultFolderInbox = 6
$inbox = $mapi.GetDefaultFolder($olDefaultFolderInbox)

# access the target subfolder

$olTargetFolder = $inbox.Folders | Where-Object { $_.FolderPath -eq $olFolderPath }

# load emails

$emails = $olTargetFolder.Items
$count = $emails.Count

# process the emails

$count = 0
foreach ($email in $emails) {
    $email
    $count = $count + 1

    # Format the timestamp
```

```
    $timestamp = $email.ReceivedTime.ToString("yyyyMMddhhmmss")

    # Filter out the attachments

    $email.Attachments | ForEach-Object {

        # Insert the timestamp into the file name
        $fileName = $_.FileName
        $fileName = $fileName.Insert($fileName.IndexOf('.'), $timestamp)

        # Save the attachment

        $_.SaveAsFile((Join-Path '[File Path]' $fileName))
    }

    # Phase-3: Convert Text from an Image file using the PSOImageToText module and save it to a file

    foreach ($file in Get-ChildItem "[File Path]") {
        $OCRText = Convert-PsoImageToText -Path (Join-Path '[File Path]' $file)

        # Read the text file containing the text from the image file

        Write-Output $OCRText >> "[File Path]\ImageToText.txt"

        # Read Indicator of Compromise file (Make sure this file is updated frequently from threat
intelligence feeds)

        $IOC = Get-Content -Path [File Path for the String Match File]\StringsToMatch.txt

        # Search IOCs from the ImageToText file

        foreach ($string in $IOC) {
            if ($OCRText -match $string) {
                Write-Host 'Contains String: ' $string

                # Phase-4: Delete emails

                $subject = $email.Subject
                $from = $email.SenderEmailAddress
                $SenderName = $email.SenderName
                $attachment = $email.Attachments | Select-Object -ExpandProperty DisplayName
                $Search = New-ComplianceSearch -Name "[NewComplianceSearchName]" -
ExchangeLocation All -ContentMatchQuery "(attachment:$attachment)(from:$from)"
                Start-ComplianceSearch -Identity $Search.Identity
                New-ComplianceSearchAction -SearchName "[NewComplianceSearchName]" -Purge

                # Clean up ComplianceSearch

                Remove-ComplianceSearch -Identity "[NewComplianceSearchName]" -Confirm:$false

                # Track Removed Fake-Invoice emails
```

```
            Echo ----- $email.ReceivedTime "From: " $from "To: " $email.To $subject $attachment >>
"[Path]\Trackers.txt"

            Exit
          }
        }

        Remove-Item [File Path]\*.*
    }

    Echo "Purge Succeeded"
}
```

Appendix B

Geek Squad	888 808-8226	\(833\)-382-5611
GeekSquad	888 808-8228	888-949-3492
PayPal	\(808\) 755-4476	866-563-2808
Pay-Pal	\(844\) 397 7452	807-770-9563
Norton	1-808-666-6112	989-884-0201
LifeLock	888-297-0415	855-204-1993
(888) 808-8225	1-\(800\)–306–2981	888-805-3464
\(888\) 808-8225	844-211-2097	888-660-5657
\(888\)808-8225	855 926 3414	888-988-2738
888 808-8227	855 538 1691	\(808\) 272-9479
888-949-9432	888-758-1025	+1 888 228-8236
+1 \(888\) 489-2060	\(888\) 990-6803	855-857-2244
818 963 9046	888-384-0056	+1\(888\) 697 5085
\(806\) 839-6096	818-797-0937	888-986-0145
\(888\) 404-4624	904-650-0950	888-243-3183
833 382 8887	1-\(888\)-738-8146	+1 801 833 0348
\(808\) 720-4622	1 \(803\) 263-6654	888-872-0883
\(808\) 658-8805	1 866 681 0802	\(808\) 800-8785
855-654-2777	\(845\) 385-5565	888-392-3179
218-262-9639	1 888-338-7751	877-653-3728
888-395-9408	1 888-436-0814	866-748-0439
1-805-206-2624	1 \(818\) 527-4140	888-727-0427
1-808-318-8005	1 \(888\) 273-3449	808-646-8594
888-505-0949	1\(844\) 480-3111	1\(844\)594-0268
+1 \(808\) 229-2338	+1 \(808\) 437-8454	1-845-317-7313
\(808\) 229-3137	1-888-616-8191	888-988-7286
888-795-8522	+1 808-444-5401	888-682-2711
808 272 7580	+1\(805\) 386-6133	\(808\) 493-1933
808-229-3680	+1 \(888\) 354-1387	+1 808 515 4814
\(808\) 229-3034	1 877-363-5566	888-616-4196

888-365-8944

\(855\) 459-7988

\(808\) 646-5251

877-762-0736

+1 \(888\) 634-8657

+1 \(888\) 296-4945

\(818\) 533-9751

818 435 8423

Bayarmaa Havel, bamaa2000@yahoo.com