Global Information Assurance Certification Paper

Automating Response to Phish Reporting

GIAC (GSLC) Gold Certification and MSIM 5501

Author: Geoffrey Parker, obiwan324@gmail.com
Advisor: Tanya Baccam

Accepted: May 23, 2019

Abstract

Phish Reporting buttons have become *easy buttons*. They are used universally for reporting spam, real phishing attacks when detected, and legitimate emails. Phish Reporting buttons automate the reporting process for users; however, they have become a catch-all to dispose of unwanted messages and are now overwhelming Response Teams and overflowing Help Desk ticket queues. The excessive reporting leads to a problem of managing timely responses to real phishing attacks. Response times to false positives, spam, and legitimate messages incorrectly reported are also significant factors. Vendors sold phish alert buttons with phishing simulation systems which then became part of more in-depth training systems and later threat management systems. Because of this organic growth, many companies implemented a phish reporting system but did not know that they needed an automation system to manage the resulting influx of tickets. Triage systems can automate a high percentage of these phish alerts, freeing the incident response teams to deal with the genuine threats to the enterprise on a prioritized basis.

© 2019 The SANS Institute Author retains full rights.

1. Introduction

From humble origins on January 2, 1996, in a Usenet group called AOHELL (phishing.org, 2018) to the modern-day occurrences and extreme financial computing Infrastructure and Intellectual Property (IP) damage caused by phishing, organizations, enterprises, and institutions have been on the defensive trying to fend off criminal attacks. Over time, the attacks have become more and more sophisticated and harder and harder to detect (Proofpoint 2019).

Phish Reporting buttons first came into commercial use during 2015 (Wombat Security, 2015). They were initially marketed and sold as part of training programs aimed at creating behavioral modifications in users. The companies that manufactured and sold these buttons were trying, as part of their growing IT Security Awareness Training suites, to give users a way to identify and easily report phishing.

In the early days of phish reporting, which was only a few years ago, the focus was on the reporting process to educate users on how to eschew falling victim to an email-based phishing attempt. The focus during this time was on teaching users what phishing is, how to be alert to phishing, the different kinds of phishing, how to detect phishing and how to report phishing (SANS, 2016).

Vendors at the time focused on including a "report phishing" button as a feature and a benefit to help sell their particular training solution. Over time, the focus changed from reporting phishing to protecting and defending against phishing. Security Information and Event Management (SIEM) has evolved from discrete, separate systems. Threat response systems are far more integrated, automated, and mature in many ways than they were just a few years ago. The focus of phish reporting has now changed to a certain degree. Companies are now no longer focused on the simple reporting of the potential phish, but instead, are looking at how to respond to the phish. Response teams – however large or small – must deal with those phish reports coming in and creating response team tickets.

To a point, even smaller organizations with anywhere from 500 to 2500 employees are receiving a high volume of messages. The researcher's company is receiving

approximately 1.6 million messages a month from the external gateways, which translates to more than 50,000 per day (Cheniere Energy Inc., 2019). From these messages, typically 93% are logged as threats or spam. Although automation resolves most of that, users are still confronted with many emails each day, and many of these could be phishing.

The problem then becomes how a Security Operations Center (SOC) team deals with the influx of messages. Users are now using the [Report Phishing] Button almost like an *easy* button for whatever they do not want, do not need, or even do not like, let alone the messages that are actual threats to the organization (Cofense, 2018).

A best practice to help mitigate the inevitable false positives of phish reporting is to create a culture of awareness. The prudent company will ask users in a new employee orientation to become a human sensor network. Users will see the [Report Phishing] tool in the email client. The users learn the difference between spam and phishing so that they can spot red flags that signal one or the other. Finally, users can be educated to mark potential spam as junk, rather than reporting it. Reiteration of these aspects of reporting phishing in the mandatory annual IT Security awareness training will also be helpful.

Because of the potentially high volume of phishing reports from the Reporting Phishing button that IT Security Awareness officers have taught users how to use, organizations now need an automated solution to deal with the output alerts that it generates. Options now exist for triage solutions, either as standalone software tools or as part of larger packages or platforms. This research examines three such commercial solutions: Cofense (PhishMe) Triage, KnowBe4 PhishER, and Proofpoint TRAP. The research will also analyze an open source solution, StackStorm, and finally, determine whether PowerShell could be used to automate responses via scripting.

2. Identifying the Problem and Determining a Solution

At the start of the research, blurred expectations existed. The options for viable solutions were not known. Colleagues in the IT Security Awareness community and the systems admins and technical services colleagues, as well as incident response team members, were all observing similar issues (SANS, 2019). These issues stemmed from a flood of messages into the response queues from the armies of Happy Clickers out there who

are using the [Report Phishing] button as an Easy Button. One imagines that in some cases they merely did not realize that the message they reported was going somewhere and that someone would have to do something about it. What they saw was a message leave their inbox when they clicked the [Report Phishing] button.

These users could have benefitted from education on right clicking on an unwanted message and clicking on block sender to resolve these issues. In other cases, they thought they were doing the right thing. In still other cases the users seemed to be legitimately reporting what they thought were obvious phishing attempts. A manual response system by incident responders is not feasible in this environment for all but the very smallest enterprises. Response Team ticket queues quickly grow out of control. Response Teams start experiencing *alert fatigue*. Non-critical messages stop being responded to either within their Service Level Agreement (SLA) or at all as resources dwindle and priorities to deal with real threats increase.

This problem is rampant as the industry struggles to shift from an IT Security awareness training and threat reporting perspective to a threat response and automated management environment. Email in corporate environments is a cornerstone of how organizations communicate in the modern era. A single solution that can block or stop phishing does not exist. Nation-state actors or other nefarious players will continue landing problematic phishing messages in mailboxes that users must deal with in some way.

Having armed users with a [Phish Report] button, it now becomes necessary to manage this new information stream using automation. Thus, emerges the need for available solutions to deal with the false positives, the nuisances, and the actual attacks. Incident Response teams must be available to deal with real compromised threats that face organizations. See Figure 1 below.

There are many potential automated triage solutions and tools available. In researching the top products in use across the globe (SANS IT Security Community Discussion Board, 2019) by respected IT Security professionals, some tools emerged as most widely used. The companies surveyed included many different industries and verticals, and organizations of various sizes. This study looks at the top three commercial solutions, one open source solution, and one potential Windows PowerShell scripting -based solution to see

which one is the most effective at triaging and automatically responding to [Phish Report] button messages…

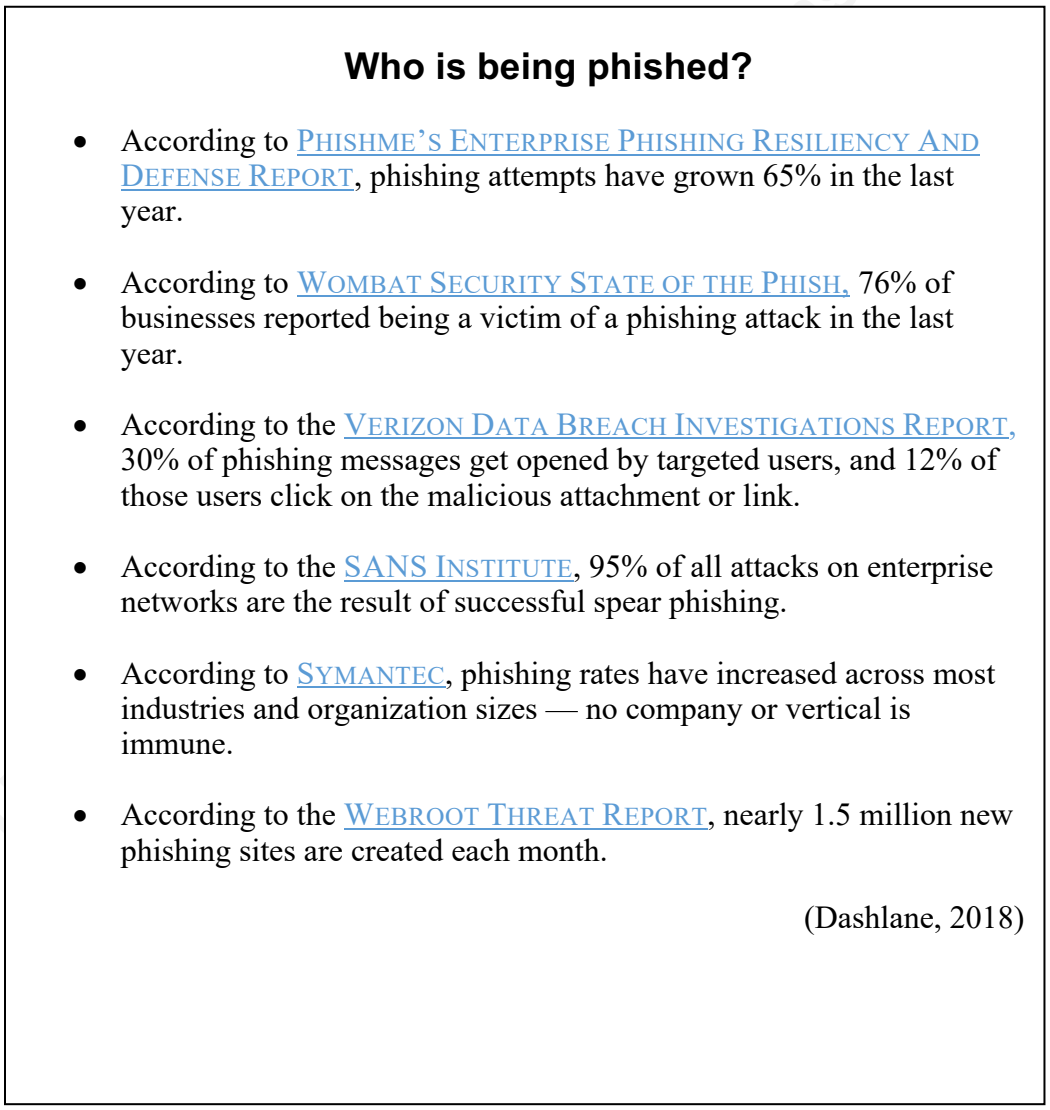

Figure 1. Key Phishing Statistics

3. A Typical Phish Report workflow

While all companies have unique infrastructures and workflows, there are specific standards which must be adhered to for IT Support to function correctly. ITIL V 4 (Axelos, 2019) is one such standard which dictates best practices in IT service and support. In a traditional environment, a user will receive a message in their inbox. If they deem the

message to be suspicious and believe it to be phishing, they will use the [Phish Report] button to report the email.

The [Phish Report] button, when activated, will typically use X-header programming that does three things:

- Sends the reported email as an attachment with origin headers intact to a designated mailbox
- Takes the reported message from the user's inbox and puts it in the deleted folder
- Displays a message to the end user thanking them for submitting the report

Two other functions of a [Phish Report] button come into play when a company launches an internal phishing assessment (campaign). In this case, an added feature of the [Phish Report] button is to congratulate the user on accurately detecting a phishing campaign and reporting it. A further function is to include flag data in the X-header that reports back to a campaign console. This data includes date and time stamps for the user's button activation (reporting the phish) for inclusion in campaign data.

The ingesting mailbox location may be a discrete inbox which ingests phish reports from users for processing by analysts. In larger or more sophisticated organizations, there may be rules and scripting, which takes the reported message, and based on commands, carries out specific actions. A typical example of this would be to convey the reported phish to a centralized help desk (support) ticketing system. In many cases, based on rules created within the help desk ticketing system, with the presence of the tag [PHISH ALERT] the ticketing system will create a ticket and shunt it to a particular queue, such as the SOC (Security Operations Center) or Response Team queue. At this point, a human analyst from the team would examine and analyze the reported message and manually respond to the user. This workflow appears in Figure 2 below.

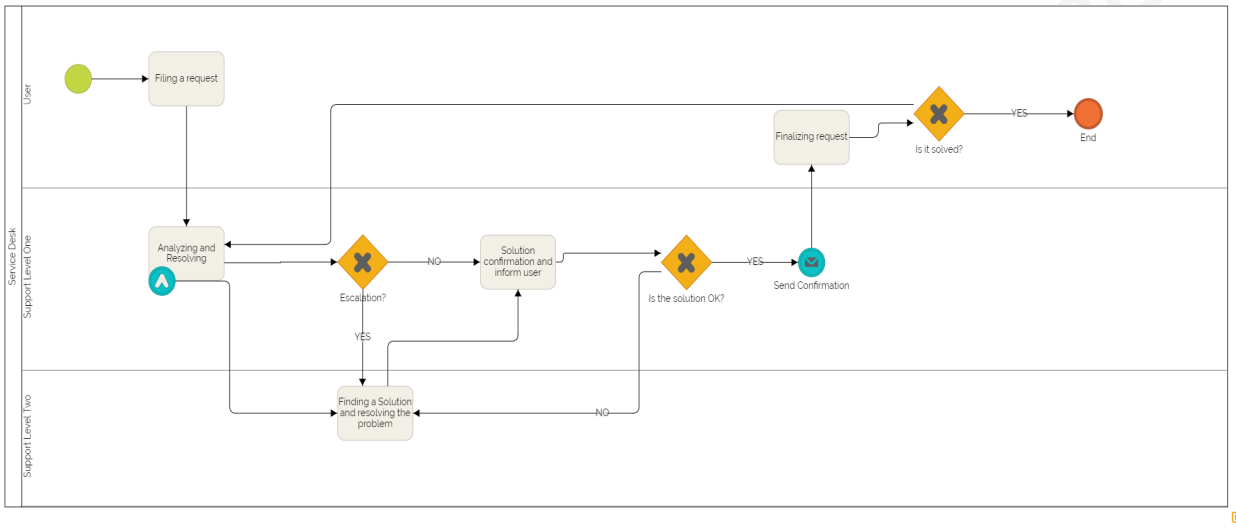

Figure 2. ITIL V 3 Service Desk Process (Heflo, 2018)

3.1. How a Triage System Works

A triage system is designed to ingest a [PHISH REPORT] and take the following actions:

- Analyze it for threats, spam, or legitimacy
- Compare it against known threats (e.g., malware, ransomware) from its current databases
- Match it against Virustotal or a virus signature database for pattern matching
- Respond to the user based on its analysis using automated rules and templates
- If analyzed route the message to a processed mailbox
- If unable to identify the message, send it to the analyst's / Threat Response queue for further processing.

By taking these actions, the triage system can process more than 90% of the reported messages it ingests. Figure 3 shows an example of the ticketing system alert that comes through the system.

Fwd: The following has been added to your group Ticket ID: ##RE-184865## [[Phish Alert] FW: Your Recent Zelle® Transfer(s)]

Figure 3. Sample [Phish Alert] subject line

4. Research Method

Because many different companies, institutions, and organizations exist at many different levels, from smaller organizations with just a few hundred employees to global

companies with many thousands of employees, there is no such thing as a one- size- fits- all scenario for phish alert triage systems. The enterprise scope and suitability are all factors to consider when considering the purchase of a package. A test scenario of 100 messages comprised of different types of spam, phishing, and legitimate messages will be used to test the software packages. See Table 1 below.

For scale, the reference modeled company is in the oil and gas industry. The firm has 2500 users, a headquarters based in Texas, has multiple plant locations and offices with many subsidiary companies, and offices in several other countries. The company utilizes a Microsoft Exchange ADFS system and is running Office 365. This mid-size enterprise is very highly automated and resembles the structure of a large global enterprise due to the complexities of its business model.

For each of the products, the findings below will discuss their applicability to larger and smaller companies. Regardless of the size of an enterprise, from the least to the largest, phishing emails and how they are addressed is a potentially critical threat. Therefore, the suitability of a given solution to a larger or smaller company will also be analyzed.

To account for differences in scale of organization, cost analysis, and how much staffing resources are required will be examined. These packages run the gamut from comprehensive global enterprise-class software to open source (freeware) and a simplistic scripting shell. Thus, even if there is no available budget for a triage system and minimal staffing resources, analysis can be made of solutions that will work as triage systems.

The study makes comparisons as to how the products performed and variances in the way the products handled the suspect messages. The research will note any failures to process messages. The following scenario was devised to test the different systems:

The tests used a baseline of 100 e-mails for each test. Of these emails:

Percentage of messages	Type of message
20 %	Spam
20 %	Known virus payloads
20 %	Domain spoofing
20 %	Links to known bad sites
10%	Business Email Compromise (BEC)
10%	Legitimate e-mails

Table 1. Email message composition

4.1. Triage Systems tested

A selection of three commercial systems, one open source system, and one Windows-based scripting language (PowerShell) was used for testing and evaluation. These systems are:

Software name	Software Type
Proofpoint with Threat Response Auto Pull (TRAP)	Commercial
Cofense Triage	Commercial
KnowBe4 PhishER	Commercial
StackStorm	Open Source
PowerShell 6 Core	Windows / Open

Table 2. Software Titles in Study

The study selected the non-commercial product StackStorm 2 for its ability to interact with mail systems, including O365 and Exchange as well as help desk systems (Polley, 2017). Finally, an attempt was made to create a triage system using Windows PowerShell 6.x scripting features, a PowerShell cmdlet named "Get-MailDetailSpamReport" which interacts with Exchange Online and the Office 365 administrator role (Doron, 2017). Other cmdlet and functions such as "if…elseif… else" and "send_message with attachment" were also used.

4.2. Test Procedures and environments

The study tested Proofpoint on Demand and Knowbe4 PhishER in a development environment that closely modeled the company's commercial configuration. The systems and

environment, including the Exchange environment, and the mailbox structure, were duplicated. However, only testing mailboxes were configured. This configuration did not influence the behavior of the triage systems being tested since their installations are hosted and because they are configured to ingest messages containing a subject of [Phish Alert] and then follow programmed rules as to disposition.

For Cofense Triage, a hosted demonstration system was used. Emails were also received using a [Phish Alert] trigger, and the messages were delivered to a *help@mailbox* inbox. In all three cases, a test mailbox was created as the destination address that simulated a user inbox for delivery of the test messages.

By using a phishing simulation platform, test messages with the appropriate test type messages were created. The system cloned, renamed campaigns, and the message type was changed to simulate the desired type of message, as indicated in Table 1 above. For the legitimate messages, all indicators of compromise, signs of spoofing, spam, malware, or viruses were removed. The sender hostname was set to a user at the test company domain from another internal address or a valid external email address.

4.3. Test Procedures and environments for StackStorm

StackStorm was installed to a clean 64 bit Linux virtual installation. Docker was used as required (StackStorm, 2019). A basic configuration was defined, and a simplistic level of triage automation folders was created and deployed (Polley, 2017). Automated responses based on the folder template were created at an elementary level, for example, *"Thank you for reporting the suspicious email message to the IT Security Dept. This message is spam."* Three colors indicated Green for a legitimate message, orange for a spam message, and red for an actual threat reported by the user. In creating the configuration, the GIAC Gold paper ComBAT Phishing with Email Automation, written by Seth Polley in 2017 on using StackStorm for automating responses to reported email threat messages was an essential blueprint (Polley, 2017). Once the sample system was up and running and functioning correctly, the same testing scenario was used as previously referenced in Table 1.

4.4. Test Procedures and environments for PowerShell

PowerShell was run in a Windows virtual development environment with an instance of Exchange Online, Active Directory and Office 365. The goal here was to determine what, if any scripts could be automated to perform a form of triage with an autoresponder for

generated [Phish Alert] messages. Part of the rationale for testing this configuration was to determine if a viable non-commercial solution could be achieved with native tools.

5. Test Results Overview

All three commercial products performed as to their design specifications and correctly identified the type of message that had been sent. Differences in results were based on the configuration of the individual product, the logic, and priority employed with the decision-making process, and the rules that were set for the product. All three systems were able to identify spam emails, legitimate emails, and emails with virus or malware payloads. The systems correctly identified and quarantined messages with spoofed domains. Finally, the systems correctly processed Business Email Compromise (BEC) messages.

StackStorm also performed well. However, it must be noted that StackStorm was only configured as a triage response automation system, and has no other functions or features in the context of this study. It does not generate reports, trends, or graphics. It processes ingested messages and responds to users. It should also be noted that StackStorm has the most flexibility in the context of the size of the organization of all of the products tested. StackStorm could accurately function in a 1 user to 10,000 (or more) user environment. Likewise, PhishER has tremendous flexibility as it is a fully- hosted solution except for the Phish Reporting Button itself. Its triage processing speed depends on the speed of the Internet Connection, and the volume of mail received rather than the actual size of organization or number of messages it ingests.

PowerShell did not function as intended. PowerShell was not viable on its own as an automated triage response tool, and likely cannot be used as a response tool on its own. Extensive research was done into IF This Then That (IFTTT) methods for PowerShell and the investigation was unable to come up with suitable scripting or techniques to accomplish this task. This work is further detailed in Appendix 1.

5.1. KnowBe4 PhishER

Knowbe4 PhishER is a simple, full-featured, flexible and robust tool at a lower price than its commercial competitors. It appears to be geared around a smaller firm that may not have dedicated IT staff. While it can be used by larger enterprises, PhishER has the quickest

and easiest setup of any of the tools tested. In the test environment, a technician from KnowBe4 had the PhishER instance set up, configured and fully functional in less than 10 minutes. For the initial test in a development environment, the PhishER sent [Phish Report] messages to the usual help@mailbox. The help@mailbox would then have a rule triggered which, based on the tagged subject of [Phish Alert], would then forward the message to a response team queue. At the same time, PhishER ingested one hundred test messages as quickly as they were tagged and sent to the PhishER system and processed them correctly. This activity was exciting to watch as it was happening: On the one hand, a help queue rapidly filled up with one hundred messages. At the same time, one hundred responses came back to the test mailbox which had fired off the [Phish Report] emails. PhishER has canned generic messages that it comes with as its initial configuration default folders; these can easily be modified and personalized to include a more human-focused approach when a user makes a [Phish Report]. A company with one dedicated Full-Time Employee (FTE) to work on configuring PhishER could probably have it totally configured and fully deployed in one or two full days.

PhishER, Cofense Triage, and Proofpoint are all similar to StackStorm in that they all give enterprises a virtually unlimited amount of customizable Emergency Rooms (triage mailboxes are called ER's) for triage. Thus, a company can get as detail-oriented as they want – for example, one could have a triage mailbox for Business Email Compromise from typical users, and a dedicated mailbox with separate messaging for C suite users and above could be created who receive a BEC and report it.

PhishER, as with Triage, has a YARA rules editor and tool for on the fly and emerging threats. However, unlike with Triage from Cofense, PhishER does not have an intelligence feature from the Cofense Reporter module (an add-on to Triage). This Reporter module establishes the reputation of the user making the phish report. The higher the user's reputation, the more reliable their reports will be.

In terms of simplicity and ease of use, configuration, and setup, PhishER was by far the most straightforward, uncomplicated, and most user-friendly to set up. PhishER appears to be geared more towards a smaller company that does not have a dedicated IT staff or help desk. It can, however, readily scale to the larger enterprise with several thousand or more employees. The test environment had approximately 3000 Active Directory (AD) users in it

at the time of testing and a volume of roughly 2,000,000 external messages a month. PhishER performed exceptionally well in this environment. Particularly for smaller companies with limited staff, or perhaps that have only one person who is responsible for response management, PhishER would work well. PhishER is the least expensive of the commercial products tested, making it affordable for a small company as well.

5.2. Cofense Triage

Triage is a very robust system, clearly designed to go head to head with larger, more robust software packages such as Proofpoint. It is designed to scale effectively and to integrate well with larger enterprises. With its costs and its add-on modules and options for Software as a Service (SaaS) via a managed platform, it attempted to scale from the SMB enterprise to the larger global enterprise. Triage is likely not suitable for the smaller enterprise. Companies with less than 1500-2000 users may not be a good fit for the product due to its pricing and complexity and support requirements.

At the heart of the Cofense platform is the Triage module itself. This module is the feature which will automate the process of analyzing reported phishing emails and using what it calls a Noise Reduction platform to shunt spam to appropriate folders, auto respond to users, and get the real enterprise threats to the response team. The Noise Reduction module frees the Response Team to respond to threats, as they are not processing the dozens of messages in a ticket queue and attempting to determine which tickets are real phishing attacks and which are spurious.

While it costs slightly less than Proofpoint, Triage still costs for a minimal installation about two to three times what PhishER runs. With comprehensive capabilities comes higher costs, and as the organization bolts on more modules, the installation gets more expensive.

Cofense Triage is not easy to install. It is a very robust and complex piece of software which requires a dedicated support team to implement. Implementation for an enterprise of from 2500 to 5000 persons can be expected to take up to 2 weeks to install, depending on the type and complexity of installation the company chooses (Cofense, 2019).

Working with a small, basic, test setup, assisted by an experienced sales engineer and a support team representative got the demo installation up and running. It took approximately

12 working hours to get it installed and configured as the researcher was learning the system as it was being configured. Some of this was due to the complexities of setting up in a virtual development environment and the rigors of creating message handling and transport rules. It was also dependent on ensuring that the test simulation parameters specified could actually send and receive correctly in the demo test environment.

One of the most helpful features of Triage is its ability to render a visual representation in numbers and with color coding of the status of the triage reporting queues. A dashboard view shows the exact system state with regards to incoming reports. The Cofense Reporter module, an option in addition to Cofense Triage, shows the reputational standing of the users making the phishing reports. This helpful tool can come in handy when a particularly challenging spear phishing attack comes in. For example, an attack in which there is no malicious payload, no known bad link, just a few people get it, and there is a valid sender. Seven very highly -rated persons observe that it is purportedly an email from IT Support. However, it came from an email address in the Middle-East and reported the message as a phishing attack. The Triage Reporter system assigned these users very high reputational and credibility ratings. Thus a message which otherwise would not have triggered alarms is correctly brought the attention of the Response Team due to the users high level of reliability in reporting phishing emails. In this scenario, it is essential to know that the report is being made by several highly credible persons with very high reporting reputations, not 37 happy clickers who seem to report more false positives than actual phishing attacks.

Cofense Triage and Reporter make a very compelling solution. Cofense offers hosted, on-premises and managed versions of their product, in ascending order of cost.

5.3. Proofpoint / Threat Response Auto Pull

Proofpoint is a powerful, sophisticated, and costly system. Of all the products tested, it is the most comprehensive and the most suited to an advanced large scale environment. Relying on its own internal set of rules and logic, and taking its information directly from firewalls, Intrusion Detection Systems (IDS) and virus scanners, Proofpoint is far more than just a triage system. Proofpoint is a full threat detection and prevention system, which has many optional features. One of these modules is called Threat Response Auto Pull (TRAP),

which is a subset or stripped down version of its full Threat Response (TR) system. Proofpoint can be utilized in an on-premises system, as Proofpoint On Demand (POD) which is a cloud-based hosted system or a hybridized system.

As with Cofense Triage, everything a company adds to the installation of POD costs more money. POD is designed, built, and scaled as an enterprise-level solution, and it does an excellent job of protecting the enterprise. However, due to its cost, its complexity, and the high level of administration and maintenance it requires, POD would not be a viable solution for the smaller environment. The research environment of approximately 3000 active users and five global locations might be considered a lower threshold for the applicability of POD.

This research focuses on the testing of the TRAP portion of POD. Proofpoint is very rigorous and complex and complicated to install. Arranging a demo of the system in the virtual environment took a team of people, both engineers, and technicians from the researcher's company and engineers and technicians from Proofpoint. A Proof of Concept (POC) was arranged for the company to enable the testing of the TRAP module.

The TRAP module is the triage portion of Proofpoint. Its job is to automatically respond to reported threats according to its rules-based assessment of reported threat messages, and dispose of them accordingly. Because Proofpoint has such a complex rules framework and a priority for how it assesses threats and acts on them, Proofpoint can sometimes have unexpected, if unpredictable results for how it acts on reported threat messages. An example of this is seen below in Figure 3.

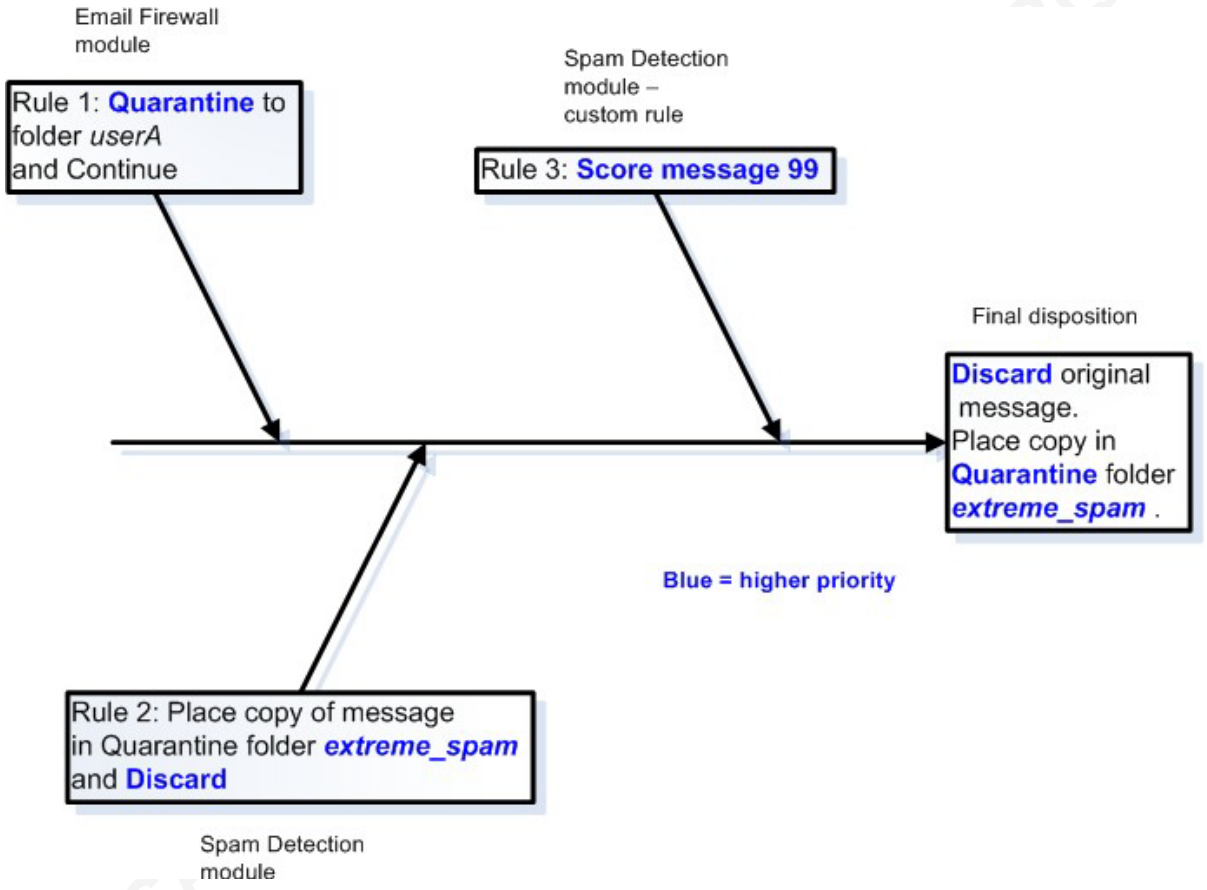

Figure 3. An example of processing rules in Proofpoint

5.4. StackStorm

StackStorm also worked as expected, but it relies on external appliances or software to determine where a message has a malicious payload or if it is likely to be phishing. For example, if the company is using Proofpoint for border protection, the Proofpoint Targeted Attack Protection (TAP) module may strip an attachment and leave a placeholder. StackStorm still must then ingest the message (Proofpoint TAP indicates it had a malicious attachment by inserting a placeholder in the message body). The message is then sent to an automated mailbox which will, in turn, trigger the programmed response to the user. Next, an analyst moves the message to the final folder for disposition. As a human must make the ultimate decision on an email item, and move it to a legitimate or illegitimate folder. In testing StackStorm, the researcher inserted a delay of 5 minutes to guard against mistriage by an analyst and give them sufficient time to correct an error. Also, a rule should be put in

place that only allows automated messages to be sent to internal mailboxes (Polley, 2017) which would prevent spoofing (e.g., anything with a @sign is not allowed).

StackStorm is a simple and straightforward system and does an adequate job for what it is. StackStorm lacks in some of the more refined features such as the intelligence that Triage Reporter has, or the flexibility that PhishER has, or the enterprise-level sophistication that Proofpoint has. It has no reporting and no graphics, and no other features apart from the actions it performs. For organizations that do not have a lot of financial resources, or are on the threshold of needing something, but are big enough or sophisticated enough to want to have and use a phish reporting button, it could be a serious alternative. Like its commercial counterpart in PhishER, StackStorm has applicability and scalability from the very modest company of just 5-10 employees up to thousands of employees. There is only a little variation in how fast or how well it runs based on the size of the enterprise. The essential difference to consider would be the volume of [Phish Reports] that are made based on email volume the company (and users) receive.

One observation about StackStorm that is not readily apparent until the system is used and experienced is that there is a real sense of joy, intrigue, and accomplishment in getting StackStorm to do its job. One feels continuously intrigued to wonder if StackStorm could do something X or Y or Z. While this is an intangible, StackStorm gives the user a sense of hands-on accomplishment that the other, more sophisticated software systems do not. StackStorm has an active user community in which this pride of creation and continuous process of creating new features, value, and capability plays a remarkable role.

5.5. PowerShell

This research was unable to create a working system with PowerShell to effectively triage the [Phish Alert] messages. While all aspects of the X-header information are available, and one could write and use scripts to do many useful things, the researcher could not assemble a meaningful collection of scripts using If-This-Then-That (IFTTT) or cause and effect scripting. The closest that the researcher could come to significant distribution are some scripts to retrieve and forward messages, but they could not ingest to a specific mailbox based on criteria or determination as to whether they were legitimate, spam, or phishing. They could not trigger IFTTT, and they had no way to evaluate or call meaningful routines to

assess. The research located a lot of useful scripts, particularly for working with Exchange Online and reporting on the Spam folders (Doron, 2017).

It may be possible to use a tool such as PowerShell to assist with triage, but it does not seem likely that this tool is suited for this purpose. It would very likely require a skilled and experienced programmer to create the necessary scripting and ancillary tools to produce a system based on PowerShell that could be used as a triage system for phish reporting. Such a system would likely need to be supported on an ongoing basis by a paid professional or a Full-Time Employee (FTE) in a dedicated position. Admittedly, the researcher is not such a professional programmer and has what must be considered a base level of skill with PowerShell. However, at the research and testing firm, two highly skilled and experienced professional programmers were consulted and were also unable to create a working system in conjunction with the researcher. It was deemed likely that third-party, add-on cmdlets would be needed and that PowerShell was likely not a suitable environment if one wishes to create a scripted triage system (see Appendix 1).

PowerShell was evaluated to see if it could be used as a potential triage automation response system for phish reporting. PowerShell was examined for suitability as a response tool using scripting alone. There may be other languages or combinations of languages and tools that could fulfill this function effectively, but in regards to PowerShell itself, it is not capable of doing this job on its own as referenced above and in Appendix 1.

6. Findings and Discussion

Two conclusions are clear based on the extensive research into an automated triage solution for reporting phishing. The first conclusion reached is that there is not one solution that will work for all organizations. However, many viable, substantial solutions exist that will work for nearly all enterprises and can eliminate the bottleneck and inefficient use of technician and analyst time to process Phish Reports.

These range in price from approximately $4.00-$6.00 per license seat at the low end for PhishER to $18-$22.00 per seat for Cofense. Adding more options and modules to Cofense Triage brings the cost higher. Proofpoint on Demand with TRAP will cost $20-25.00 per seat. Again, these are subject to pricing for the size of an organization, the number of years of licenses purchased, and the options selected. Opting to purchase a commercial

solution could potentially represent a significant financial investment even for a small company. At the modest size of 2500 users, the financial commitment is between $10,000 and $37,000.00 per year.

Please note that these prices are for licensing costs only. Installation, setup, configurations, support will all be factors and could add to the cost. The cost per technician on the purchasing company's side when their employees are dedicated to such a project and the time lost factor when other work and projects are not getting done should also be considered in the total cost of ownership.

The cost to Response Teams should be static or decrease once a system is in place as less time will be spent responding to tickets that the system handles.

In theory, factors such as streamlining processes, time to resolution of the ticket, and accuracy should all increase as the human element is taken out for the major percentage of handling Phish Reports. The speed to resolution should markedly decrease as wasted and lost time responding to spam, and false positives should drop to near zero. As well, the users will have a better response rate and further foster and support a culture of sensing and reporting. This improved culture is due to the automation of responses as overloaded response teams tend to ignore non-threatening messages and users then get no positive (or even intermittent) reinforcement for the fact that they are reporting phishing. With a triage system in place, the reporting users will get positive reinforcement consistently for reporting phishing. They will also get some Just In Time (JIT) training to refine their skills in identifying what should and should not be reported as phishing with the feedback they receive from the triage system.

StackStorm is a free system that is only based on IFTTT, has no bells and whistles but also works well. Requiring only a clean Linux 64 bit box to run and a minimum of configuration and setup; it is a viable alternative for those organizations simply wanting to implement a basic triaging autoresponder. A skilled analyst will be able to make ongoing enhancements to StackStorm and perhaps as time goes on both develop and find other useful add-ons to the system.

In terms of economy of scale, a single technician or analyst would probably be able to handle a small to mid-size organization using StackStorm.

Geoffrey Parker, obiwan324@gmail.com

In terms of feasibility for use, and suitability of purpose, while PowerShell can do a great many powerful and useful administrative things in the Windows environment, working as a triage system with an autoresponder to [Phish Reports] is not one of them.

The research confirmed what is already known: that the bad guys are always upping their game, and nothing will stop them.

The research shows that threat mitigation needs to be ever advancing in order to be effective. Threat mitigation can be effective in the following ways: awareness training for employees, modern systems which are frequently patched and upgraded, through innovative approaches to new threats as they are discovered in the wild, and through cultural change. It is incumbent upon IT Security professionals as custodians with a fiduciary responsibility to protect the CIA of critical information resources to find new ways to proactively protect and defend the organizations which everyone works so hard to build and maintain.

7. Recommendations and Implications

Two significant implications become clear. The first is that a [Phish Report] button does provide a quick and easy way for users to report phishing; this solves the reporting problem. Unfortunately, it creates a far more dangerous problem of how to process those alerts. Without a system in place to automate and treat [Phish Reports], once the organization becomes acculturated to using the [Phish Report] tool, the response team will become overwhelmed. Thus a [Phish Report] phishing button should not be put into play without a triage system to manage it.

The second implication is that for each [Phish Report] and its accompanying response, comes the opportunity for a teachable moment. Part of the installation manifest for a [Phish Report] tool is a lot of Happy Clickers. Since a template response will be created, the researcher believes that it is critical to conduct JIT training with the automated responses. The triage system will present an opportunity to train users on spam versus phishing and the proper protocol to follow. This opportunity should not be overlooked. The more the users become skillful at detecting a phishing attempt, the safer the organization.

Geoffrey Parker, obiwan324@gmail.com

7.1. Recommendations for Practice

This paper has discussed the use of a [Phish Report] button on system computers, but mobile devices also present an issue to be addressed. The full integration of a [Phish Report] button into a mobile device is mandatory and critical. Lack of such a reporting feature will continue to fill up the ticket queues. Users forwarding e-mails from their mobile devices will block the automated system and recreate the problem. May make the problem bigger as higher mobile use climbs. Any [Phish Report] and triage solutions must include a mobile reporting and response component to glean the benefit of the triage system and not create ancillary problems.

Most modern organizations conduct phishing assessments on a regular frequency. The human cost factor time alone for just one such campaign that does not have the means to automate the reporting and response process for reporting phishing could have significant implications.

A campaign in which 10,000 messages are sent that yields even a conservative reporting rate from a mobile device could produce a storm of Response Team tickets. For one such campaign alone, the potential exists for days of work for a response team if a mobile reporting and triage solution is not in place.

Therefore an integrated mobile solution which includes automated triage becomes a crucial aspect of any [Phish Report] installation.

7.2. Implications for Future Research

This study explores the need for a triage system to back up a phish reporting feature in an email client. Future studies may want to incorporate the statistics of phish reporting; how many reports come in per assessment, how many reports come in on average per campaign and how many reports per received real phish attempts are all relevant data that could be explored. The collection of this data and incorporation of the lessons learned from it may serve as a future basis for training of users on how to fend off such attacks and keep the organizations safer. Perhaps a correlation equation can be found between the volume of a user's reporting and the accuracy, quality, or both of their threat detection. There is much potential for far greater use and manipulation of metrics and Key Performance Indicators (KPIs).

Geoffrey Parker, obiwan324@gmail.com

The human element is the most significant factor and the greatest weakness in detecting and reporting attempted compromises from phishing. Research on the effective use of artificial intelligence in detecting and responding to threats, but also as a means to modify and deliver effective training is something that needs to be examined. A proactive mitigation and training strategy as the online and mobile world becomes ever more integrated into the corporate setting is going to be vital.

In the modern industrial infrastructure environment, Digital Control Systems (DCS) Industrial Control Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) systems are becoming more and more integrated into an organization's overarching technology structure. Marine navigation and control systems have grown tremendously over the past several years. These systems have not been given nor treated with the sophistication that corporate, administrative, and financial systems have been given. As the industrial control environment meshes more and more with the corporate network environment, and critical systems rely on safe and secure functioning, future studies should examine where the industry needs to pay attention to phishing attempts, threat detection and reporting, and automation to address such reported threats as well.

8. Conclusion

Phishing is one of the greatest threats to the organization. Email is a fundamental cornerstone of communications in the organization. Many jobs require that users open most if not all of the messages they receive. Because of this, phishing attempts to compromise an organization, steal money, or steal Intellectual Property (IP) run rampant. Every day news stories show just how bad the state of phishing is in 2019. Because of this, a simple, straightforward method to report threats from phishing attacks is necessary. Reporting alone is not enough. A means to process, as well as accurately and effectively address these threats is also mandatory, and one cannot be separated from the other. The people that do report need to be thanked for their actions, encouraged to continue and improve, and educated as to what to report and what to delete. Many adequate commercial solutions will enable organizations to implement such a system. Some of these systems have tools that include very robust reporting, metrics, and intelligence on the quality of the reports. There are also tools that are open source, very functional, and useful as well that should be considered.

Organizations which use these toolsets will be able to reduce costs and person-hours,

streamline processes, and become more efficient at analyzing, catching, and deterring such threats. In the final analysis, these tools help organizations and their people remain safer and achieve even grander accomplishments.

Geoffrey Parker, obiwan324@gmail.com

References

PowerShell, VB Script, SQL and JavaScript - TechNet IT Pros and Scripting Guys. (2018). Retrieved from https://gallery.technet.microsoft.com/scriptcenter/site

Doron, E. (2017, October 13). *How to use the spam mail report PowerShell script | Part 3#3*. Retrieved February 9, 2019, from https://o365info.com/how-to-use-the-spam-mail-report-powershell-script-part-3-3/

Doron, E. (2017, October 13). Using Get-MailDetailSpamReport PowerShell cmdlet | View and export spam mail report | Part 2#3. Retrieved February 9, 2019, from https://o365info.com/using-get-maildetailspamreport-powershell-cmdlet-view-and-export-spam-mail-report-part-2-3

Doron, E. (2017, October 13). *Office 365 spam mail report using PowerShell | Introduction | 1#3*. Retrieved February 9, 2019, from https://o365info.com/office-365-spam-mail-report-using-powershell-introduction-1-3

Egan, G. (2019). *Proofpoint 2019 State of the Phish Report*. Retrieved from https://www.proofpoint.com/us/corporate-blog/post/2019-state-phish-report-attack-rates-rise-account-compromise-soars

Guru99. (2019). PowerShell Tutorial for Beginners: Learn in 1 Day. Retrieved February 5, 2019, from https://www.guru99.com/powershell-tutorial.html

Virustotal.com. (2018).How it works. (n.d.). Retrieved February 4, 2019, from https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works

Proofpoint. (2019, February 28). Introducing PhishAlarm, Wombat's One-Click Email Reporting Button. Retrieved March 3, 2019, from https://www.proofpoint.com/us/security-awareness/post/introducing-phishalarm-wombats-one-click-email-reporting-button

ITIL Foundation Certification | ITIL. (2019, March 4). Retrieved from https://www.axelos.com/certifications/itil-certifications/itil-foundation-level

Heflo. (2016, August 8). *ITIL Service Desk Process Flow: Efficient Incident Management!*. Retrieved March 31, 2019, from https://www.heflo.com/blog/itil/itil-service-desk-process-flow/

Geoffrey Parker, obiwan324@gmail.com

Katz, E. (2018, December 7). Phishing Statistics: What Every Business Needs to Know. Retrieved April 2, 2019, from https://blog.dashlane.com/phishing-statistics/

Knowbe4.com. (n.d.). PhishER Product Manual. Retrieved February 18, 2019, from https://support.knowbe4.com/hc/en-us/articles/360010802673-PhishER-Product-Manual

KnowBe4. (n.d.). *History of Phishing*. Retrieved March 18, 2019, from http://www.phishing.org/history-of-phishing

Parker, G. (2019, March 21). *March 2019 Security Information Exchange* [PowerPoint]. Cheniere Energy, Inc.

PowerShell, VB Script, SQL and JavaScript - TechNet IT Pros and Scripting Guys. (2018). Retrieved March 13, 2019, from https://gallery.technet.microsoft.com/scriptcenter/site

SC Magazine. (2019, April 2). *The trickle-down effect of cyberwarfare: Protecting yourself when the bad gets worse*. Retrieved April 2, 2019, from https://www.scmagazine.com/home/opinion/executive-insight/the-trickle-down-effect-of-cyberwarfare-protecting-yourself-when-the-bad-gets-worse/?utm_source=newsletter&utm_medium=email&utm_campaign=SCUS_Newswire_20190402&hmSubId=zuMw4JZs1Q01&email_hash=bac0b5087d3175a7429cb1396b448af5&mpweb=1325-6970-70895

Verizon. (2018, April). *Verizon DBIR 2018*. Retrieved from http://verizonenterprise.com/DBIR2018

Writing YARA rules — YARA 3.5.0 documentation. (n.d.). Retrieved March 8, 2019, from https://yara.readthedocs.io/en/v3.5.0/writingrules.html

Zetter, K. (2015, April 14). *Email Phishing Attacks Take Just Minutes to Hook Recipients*. Retrieved February 28, 2019, from https://www.wired.com/2015/04/email-phishing-attacks-take-just-minutes-hook-recipients/

Bromiley, M. (2019). *Defend your business against phishing*. Retrieved from SANS Institute website: http://sans.org/reading-room

Cofense. (2018). *Cofense Triage Incident Response Platform*. Retrieved from http://cofense.com/contact

Cofense. (2018). *19 Minutes: A Minute-by-Minute Account of Collective Defense in Action*. Retrieved from http://cofense.com

Geoffrey Parker, obiwan324@gmail.com

Polley, S. (2017). *ComBAT Phishing with Email Automation*. Retrieved from SANS Institute website: http://sans.org/reading-room

Proofpoint. (2018). *Proofpoint message filtering overview*.

Proofpoint. (2018). *PTR vs. TRAP*. Retrieved from http://proofpoint.com

Proofpoint. (2018). *Filter Order - Quarantine Precedence*. Proofpoint Inc.

Shackleford, D. (2019). *How to Conquer Targeted Email Threats: SANS Review of Agari Advanced Threat Protection*. Retrieved from SANS Institute website: http://sans.org/reading-room

Wojewoda, T. (2019). *Hunting and Gathering with PowerShell*. Retrieved from SANS Institute website: http://sans.org/reading-room

Outtrim, M. (2018, February 12). *Sec Ops Team Procedures and SLA v. 2.0*. Cheniere Energy Inc.

Knowbe4, Inc. (2019). *PhishER*. Retrieved from www.knowbe4.com/products/phisher

Proofpoint. (2018, July 14). *Proofpoint Threat Response Auto Pull (TRAP)*." Retrieved from www.proofpoint.com/sites/default/files/pfpt-us-ds-threat-response-auto-pull.pdf

Proofpoint. (2018, June 10). *Proofpoint Email Fraud Defense EFD360.* Retrieved from www.proofpoint.com/sites/default/files/pfpt-us-ds-efd360.pdf

Anti-Phishing Working Group, Inc. (2019). *Technical Whitepapers and Briefings from APWG Sponsors. Retrieved from* www.antiphishing.org/resources/technical-whitepapers

Parker, G (2019, January 20). *Queries regarding the handling of false positives in phish reporting. Retrieved from* https://sth-community.sans.org

StackStorm (2018). *Event-driven automation*. Retrieved from www.docs.stackstorm.com/overview.html

Cofense, Inc. (2018). *Cofense Triage Incident Response Platform.* Retrieved from https://cofense.com/wp-content/uploads/2018/02/Cofense-Datasheet_Triage.pdf

Cofense. (2018). *19 Minutes: A Minute-by-Minute Account of Collective Defense in Action*. Retrieved from http://cofense.com

Parker, G. (2019). *SANS Awareness Community queries re Triage and automating phish reporting*.

StackStorm. (2019). *ST2: StackStorm 2 Overview*. Retrieved from http://stackstorm.com

StackStorm. (2019, April 5). StackStorm/st2web. Retrieved from https://github.com/StackStorm/st2web

Geoffrey Parker, obiwan324@gmail.com

StackStorm. (2019, April 5). StackStorm/st2docs. Retrieved from https://github.com/StackStorm/st2docs

Rauch, S. (2018, March 1). How to attach a file to an email with PowerShell. Retrieved April 23, 2019, from https://stackoverflow.com/questions/3997303/how-to-attach-a-file-to-an-email-with-powershell

Truher, J. W. (2018). *Send-MailMessage (Microsoft.PowerShell.Utility)*. Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/send-mailmessage?view=powershell-6

Tutorialspoint.com. (2019). Powershell If Else Statement. Retrieved from https://www.tutorialspoint.com/powershell/if_else_statement_in_powershell.htm

Clayman, S. (2017, September 18). Power shell script to send an email with an attachment. Retrieved from https://social.technet.microsoft.com/Forums/en-US/6296db9a-1403-4eae-b5c2-2581f6d64bfb/power-shell-script-to-send-an-email-with-an-attachment?forum=winserverpowershell

Delphix Corporation. (2019). *Example PowerShell Script for Email Notification*. Retrieved from https://docs.delphix.com/docs-old/delphix-administration/sql-server-environments-and-data-sources/customizing-delphix-for-sql-server/hooks-for-sql-server/cookbook-of-common-scripts-for-hooks-on-sql-server/example-powershell-script-for-email-notification

Thomas, G. (2019, January 9). PowerShell Basics: If -And & If -Or Statements | Examples. Retrieved from https://www.computerperformance.co.uk/powershell/if-and/

Appendix 1: Attempting to Create a PowerShell scripted Triage Autoresponder Script

1. Requirements

The PowerShell scripted autoresponder and triage system would have to perform the following functions:

1. Ingest messages from a [Phish Report] Button Press – a user reporting a phishing email
2. Determine if the message was an actual phishing attack or a false positive. A false positive could be spam or a legitimate message
3. Route the message based on the determination of real phishing or false positive
    a. For actual phishing send message to help which would automatically route to Sec Ops response team via Help Desk Ticketing System rules (based on the setup of our environment which automatically routes [Phish Alert] messages based on controls to Sec Ops response team).
    b. For false positives put in SPAM mailbox
4. Respond to the user with either:
    a. Thank you for reporting a phishing attack message. Our Response Team will analyze and revert presently.
    b. This message was reported as phishing but is not phishing. It is junk mail. Please handle it by deleting it, or blocking it or labeling as junk (include brief guidance in message)
5. Ignore messages flagged as internal phishing assessment campaign. For the research environment, this is a function of the Phish Reporting button itself; so, the researcher did not pursue this aspect of scripting. Though it could be done by examination of the X-header for a specific flag.

2. Methods

The Researcher is a novice PowerShell user; therefore, this research was undertaken with the idea in mind to see if creating the desired system was feasible and reasonable for most organizations.

The research was conducted on many sites and script repositories as reflected in the references above. Further, the researcher consulted with several expert PowerShell users at the research environment company to determine if the desired outcome and outputs were feasible, reasonable to create. If it was possible to create the triage system, what kind of time frame was involved and what kind of maintenance, administration, and setup would be required to support such an enterprise. Part of the reasoning for this expert consultation was that if it was going to need the ongoing efforts of a Full-Time Employee (FTE) to maintain the system, there was no real point in creating such a system. The researcher put in a total of approximately 6 hours of dedicated meeting time and more than 60 hours of dedicated researching and programming time during the study.

Many different attempts at coding were undertaken. The areas where the research met an unsolvable problem were:

1. Evaluating the [phish report] e-mail against a valid source such as Virus Total or a virus signature pattern database and determining if a match is found
2. Evaluating the [phish report] e-mail against a valid source for known malicious links and determining if the email contains a malicious link
3. Evaluating the [phish report] e-mail against a valid reputational source and determining if the reputation indicates the phishing
4. Routing the message to the correct mailbox using IFTTT (If This Then That) programming

3. Coding Examples

Scripting commands and cmdlets used to attempt to program the solution include but are not limited to:

```
Send-MailMessage
   [-To] <string[]>
   [-Subject] <string>
   [[-Body] <string>]
   [[-SmtpServer] <string>]-From <string>
   [-Attachments <string[]>]
   [-Bcc <string[]>]
```

[-BodyAsHtml]
[-Encoding <Encoding>][-Cc <string[]>]
[-ReplyTo <string[]>]
[-DeliveryNotificationOption <DeliveryNotificationOptions>][-Priority <MailPriority>]
[-Credential <pscredential>]
[-UseSsl]
[-Port <int>]
[<CommonParameters>]

Example:
PS> Send-MailMessage -From 'User01 <user01@researchco.com>' -To 'User02 <user02@ researchco.com>', 'User03 <user03@ researchco.com>' -Subject '[phish alert] ' -Body '[Phish alert].' -Attachments .\reported_phish.msg -Priority High -DeliveryNotificationOption OnSuccess, OnFailure -SmtpServer 'smtp. researchco.com'

Another approach would be to define an array of variables and then use them to send the emails:

Define Variables:

```
$fromaddress = "donotreply@pd.com"
$toaddress = "test@pd.com"
$Subject = "Test message"
$body = "Please find attached - test"
$attachment = "C:\temp\test.msg"
   $smtpserver = "mail.mailserver.com"
```

Using the variables in a script:

```
$message = new-object System.Net.Mail.MailMessage
$message.From = $fromaddress
$message.To.Add($toaddress)
$message.IsBodyHtml = $True
$message.Subject = $Subject
$attach = new-object Net.Mail.Attachment($attachment)
$message.Attachments.Add($attach)
$message.body = $body
$smtp = new-object Net.Mail.SmtpClient($smtpserver)
   $smtp.Send($message)
```

In the above example the researcher did not define a $cred because an acceptable method to encrypt the credentials could not be determined and credentials cannot be sent in clear text.

Several different methods and approaches were utilized to try to create the exact scripting that was needed.

To attempt to do database commands, research was done utilizing variables such as:

```
$message.Subject = "VirustotalDB '" + $env:VTDB_DATABASE_NAME + "' will be " +
$opName + " in 5 mins"
$message.Body = "<H1>SQLcompare operation " + $opName + " will occur in 5
mins</H1>"
$message.Body = $message.Body + "<BR><BR>Please save any work in this database
or it will be lost."
```

Research continued with if…elseif…else Boolean expressions and if…or expressions. The researcher wanted to see if this could be used in conjunction with or in place of IFTTT expressions to attempt to match a pattern in a virus signature or Virus Total database and then return a result which would then dictate to which mailbox the message would then be sent.

Format for the logic:

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...elseif statement.

- When using if, elseif, else, statements there are a few points to keep in mind.
- An if can have zero or one else's and it must come after any elseif's.
- An if can have zero to many elseif's and they must come before the else.
- Once an elseif succeeds, none of the remaining elseif's or else's will be tested.

The syntax of an else…if statement:

```
if(Boolean_expression 1) {
   // Executes when the Boolean expression 1 is true
}elseif(Boolean_expression 2) {
   // Executes when the Boolean expression 2 is true
}elseif(Boolean_expression 3) {
   // Executes when the Boolean expression 3 is true
}else {
   // Executes when none of the above conditions is true.
}
```

An example of this would be:

```
$x = 30

if($x -eq 10){
```

Geoffrey Parker,
golwan524@gmail.com

```
   write-host("Value of X is 10")
} elseif($x -eq 20){
   write-host("Value of X is 20")
} elseif($x -eq 30){
   write-host("Value of X is 30")
} else {
   write-host("This is else statement")
            }
```

With the Output of:

```
Value of X is 30
```

The researcher also examined If…and as well as If…or expressions in an attempt to dictate the disposition of a message should a match be found or a match not be found.

```
If (test1 -Or test2) {execute when true}

An example of which would be:

Clear-Host
$Calendar = Get-Date
If ($Calendar.day -eq '24' -And $Calendar.Month -eq '12') {"Christmas Day"}
ElseIf ($Calendar.day -eq '4' -And  $Calendar.Month -eq '7') {"4th of July"}
Else {"Today is not Christmas or the 4th of July"}
```

This approach used the threat payload or link in a message and compared it to a valid detected pattern in a given database and then returned an action of either reply to the user with a null match or (else) forward to help desk response team mailbox.

4. Conclusions

The research conducted failed to result in a functional system that met the minimum criteria of a triage system. While it seems theoretically possible that PowerShell, possibly with the addition of 3rd party cmdlets, could be developed to create a functional system, the following also appears to be true:

1. An experienced professional programmer must produce this type of a scripting system. The creation of the system will take a significant amount of time and resources to create and should be structured as an SDLC project.
2. Proper use of encryption techniques for transmission, comparison, and reception of data must be employed
3. IFTTT is not built into PowerShell, and a workaround must be determined to solve the correlation to virus, malicious or derogatory reputational patterns and then execute the next required functions
4. Such a system would require a substantial amount of administrative and maintenance time from an administrative user
5. Because other open source systems such as StackStorm exist to fill this need, there is little compelling reason to expend the resources to create a PowerShell version of the system.