Global Information Assurance Certification Paper

Generating Anomalies Improves Return on Investment: A Case Study for Implementing Honeytokens

GIAC (GCCC) Gold Certification and ISE6000

Author: Wesley Earnest, wes.earnest@gmail.com
Advisor: *Sally Vandeven*

Accepted: *October 2018*

Abstract

Putting the right information security architecture into practice within an organization can be a daunting challenge. Many organizations have implemented a Security Information and Event Management (SIEM) to comply with the logging requirements of various security standards, only to find that it does not meet their information security expectations. According to a recent survey, more than half of respondents say they are not satisfied with their organization's SIEM. The following case study deconstructs these logging requirements and the assumptions that lead to a typical SIEM implementation, and discusses an alternative approach focused on improving the organization's return on investment, decreasing security risk, and decreasing mean time to detection of a potential security breach.

1. Introduction

1.1. Smalltown Community Bank

Established in the late 1800s, Smalltown Community Bank (SCB) operates in a small rural community in mid-America. With just over $150 million in assets and 30 employees, it is commonplace for customers to be greeted on a first name basis when they walk through the front doors of the bank. SCB prides itself on the strong trust it has built with its customers and the local community. Because the bank's success depends on this trust, SCB has made significant investments in cybersecurity in recent years.

1.1.1. Defense in Depth

Beginning in 2012, after SCB hired its first in-house information technology (IT) employee, SCB started the process of evaluating and redesigning its approach to IT and information security (IS). Starting with small changes, SCB began enforcing 15-character passwords for all employees, removing local administrator permissions from users, and emphasizing security awareness training at employee meetings. SCB migrated from physical workstations to virtual desktop infrastructure (VDI) for all employees. The incident response procedures were rebuilt from the ground up and focused on quicker detection, eradication and recovery times. New network segmentation zones were created to segregate data and network traffic based on business function. SCB also implemented micro-segmentation through the use of host-based firewalls to prevent all desktop to desktop traffic within the workstation network segment. Additional network security monitoring (NSM) was installed using the Security Onion (SO) toolset, which focused mostly on the functionality of Bro and Enterprise Log Search and Archive (ELSA). Most notably, SCB spent a considerable amount of time and effort reviewing the cost, functionality, and security of its vendors. This analysis resulted in replacing numerous applications and systems, ultimately consolidating nine separate technology vendors down to one. These changes allowed SCB to build an IS strategy focused on defense-in-depth, but more importantly to create a culture of continuous improvement.

1.1.2. Past Red Team Penetration Testing Results

In 2014, SCB partnered with the Department of Homeland Security (DHS) for its Cyber Resilience Review, Cyber Hygiene (vulnerability scanning), and the Risk and

Vulnerability Assessment (penetration testing) services. SCB used the reports generated by these initial baseline tests in 2014-2015 to guide the design and implementation of its new defense-in-depth IS strategy. After implementing these changes, SCB partnered with DHS once again in 2016 and requested a new Risk and Vulnerability Assessment (RVA).

SCB worked with the National Cybersecurity Assessment and Technical Services (NCATS) team from DHS to define the rules of engagement for this test so that it would simulate a real-world attack as much as possible, including phishing, social engineering and post exploitation techniques. The goal of this test was not only to enumerate vulnerabilities that could be exploited, but to help quantify the risk to the bank's data should an attacker successfully breach the bank's preventative controls. Despite SCB's focused efforts to improve its security posture, the NCATS team was still successful in gaining access to SCB's network. The phishing campaign resulted in a twenty-five percent failure rate for SCB users. Given that a phishing attack will eventually succeed even in the most secure environment, the testing included executing a payload on a user workstation to simulate a real-world compromise.

The NCATS team then attempted to gain increased access on the compromised workstation as well as access to new systems. They identified a vulnerable service that enabled them to gain system-level access to the compromised workstation. Using this access, the NCATS team discovered the clear text credentials of the local administrator account. The NCATS team was unable to use these credentials to expand access to new systems within the SCB network due to SCB's policy of not reusing passwords across systems. The NCATS team then attempted to identify any information on file servers or shared folders that could enable further access to the SCB network. A configuration file containing possible database credentials was found in a shared folder; however, the NCATS team was unable to remotely access the database due to network segmentation. Nonetheless, several documents containing loan information, including names and account numbers, were found on a file server.

1.1.3. The Need for a Better Solution

While SCB was able to detect and alert on some of the red team activity during this test, there were still blind spots in the analyst's ability to see exactly what commands the attacker executed, and what data the attacker accessed. The RVA report included sufficient details to replicate the findings and indicators of compromise that could be used to detect similar types of attacks. First, the command and control connection to the attacker's machine was executed through code embedded in a Word document. Second, the vulnerable service was exploited using PowerShell Empire, which created a new local administrator account and retrieved the existing local administrator credentials. Lastly, the attacker was able to traverse several shared folders on the network with the phished user's credentials, which ultimately led to the potential breach of customer data. To improve visibility into these details, SCB began searching for a Security Information and Event Management (SIEM) solution that would detect and alert on this type of activity.

2. Review of the Literature

SIEM products have been in use within the industry for over a decade (Williams, 2005). The basic function of a SIEM is to ingest event logs from systems on a network, aggregate and correlate these events based on a set of rules, and then provide reports and alerts of events that indicate a potential security issue. Based on the presupposition that an attacker's actions generate log events during a breach, review and analysis of these event logs should be able to detect the presence of an attacker on a computer system or network. However, this is not necessarily the case. Influential IS firms have routinely published annual reports summarizing details found during the analysis of recent data breaches. One noteworthy metric included in the 2014 Verizon Data Breach Investigation Report was that less than one percent of breaches in 2013 were detected via log review (Verizon, 2014). This metric raises an important question about whether these breaches are going undetected as a result of log review and analysis not being the right tool for detection, or due to so many breaches going undetected because log review and analysis still has not become a mature process fully integrated into security operations.

2.1. Security Standards and Log Review Requirements

Most information security standards and regulations, such as GLBA, FFIEC, HIPAA, PCI-DSS, NIST, CIS, etc., require some form of event log collection, analysis, review, or correlation. Some of these logging requirements are very prescriptive, such as PCI DSS 10.7 which requires companies to "retain audit trail history for at least one year, with a minimum of three months immediately available for analysis (for example, online, archived, or restorable from backup)" (PCI, 2016). Other standards are less prescriptive and provide some flexibility in determining the organization's retention policy such as, "Management should have effective log retention policies that address the significance of maintaining logs for incident response and analysis needs" (FFIEC, 2017).

To help organizations meet these requirements, SANS published guidance for these event log requirements with the "Top 5 Essential Log Reports Version 1.0" (Brenton, 2006), and later, "The 6 Categories of Critical Log Information" (Czanik, 2013). The original list of five reports expanded to six categories and a total of 48 distinct reports. Organizations attempting to follow this recommendation would be obliged to review reports such as "all login failures and successes by user, system, business unit." Even in a small organization, reports such as these could be massive and require a huge time investment from IT or IS staff, who are already stretched thin with daily operations. A closer examination is required to determine the appropriate solution.

2.2. The Role of SIEM in Critical Security Control 6

The Center for Internet Security (CIS) Critical Security Controls version 6 (CSC) prioritizes "Maintenance, Monitoring, and Analysis of Security Audit Logs" at number six in its list of 20 controls (CIS, 2016). This represents a significant increase in priority over the previous ranking at number 14 in version 5 of the CSC. According to CIS, the intent of Critical Security Control 6 (CSC6) is to "collect, manage, and analyze audit logs of events that could help detect, understand, or recover from an attack" (CIS, 2016). In version 6, one of the sub-controls of CSC6 requires the implementation of a Security Information and Event Management (SIEM) to aggregate, consolidate, correlate, and analyze log files from various sources (CIS, 2016).

A recent survey shows that 84 percent of respondents said their SIEM is important, very important, or essential to their incident respondent process, which aligns with CSC6 (Ponemon Institute, 2017, March). Yet only 48 percent of respondents to that same survey claimed that they were satisfied with the actionable intelligence they receive from their SIEMs (Ponemon Institute, 2017, March). One reason for this lack of satisfaction can be found in such responses that current SIEM technologies do not provide the most accurate, prioritized and meaningful alerts (Ponemon Institute, 2017, March). More than half of the respondents (54 percent) say that SIEM generates too many alerts making it difficult for analysts to focus on what is truly important; only 25 percent say their SIEM prioritizes threat events (Ponemon Institute, 2017, March). The textbook example of this type of alert flooding condition would be the infamous Target breach.

During the Target data breach, it was reported that one of Target's security systems detected and alerted on the breach. However, Target did not take action on this alert. One explanation for this lack of action speculated that Target's security team received hundreds of such alerts on a daily basis, which would have made it tough to single out this specific alert as being a particularly malicious threat (Finkle, 2014).

2.3. Potential Alternatives

Some of these industry data breach reports have reported on metrics such as mean time to detection (MTTD). This metric has decreased in recent years from 416 in 2012 (Mandiant, 2016) to 99 in 2016 according to the 2017 Mandiant M-Trends report (Mandiant, 2017). It is important to consider that many of the regulations and security standards requiring or prescribing mass collection and storage of event log data originated years ago when MTTD was measured in months and years and has likely been the leading contributor to requirements such as CSC6, which prescribes the collection of logs from "every hardware device and the software installed on it" (CIS, 2016).

Given the statistics on SIEMs generating too many alerts and not providing actionable intelligence, it is worth questioning the efficacy of collecting more, potentially irrelevant logs from every hardware and software asset in the environment. Perhaps organizations have taken the wrong approach to complying with the spirit and intent of

these logging requirements. Most systems are configured to log a default set of events, but almost all systems have various logging functionality that is not enabled by default. Most SIEM implementations require custom filters or plugins to collect logs from various third-party applications that might be running on those systems. The key to extracting the most value in complying with these requirements is in selecting what types of events to log.

When comparing logging requirements and a video camera system for physical security, one could interpret these requirements to mean that a high-resolution camera is required to record every inch of carpet in the building and every inch of pavement in the parking lot in order to detect whether a burglar is on the premises. In this physical comparison, it is easy to observe that increasing the amount of video that is recorded will have greatly diminished returns after a certain minimum coverage is obtained. The same could be said for logging and alerting if the purpose of these logs is to detect, understand or recover from an attack.

The Critical Security Controls are intended to be implemented according to their prioritization. This means that control 1 should be fully implemented before moving on to control 2 and then control 3 and so on. According to SANS instructor James Tarala, "The critical controls provide a prioritized approach to implementing effective security controls. If you haven't implemented each part of the CSC yet, then you shouldn't be wasting time implementing things like honeypots" (Tarala, 2014). Given that more than half of SIEM implementations do not meet expectations (Ponemon, 2017, March), another approach may be warranted. The Ponemon survey also reports that on average, 25 percent of IS investment is spent on SIEM, which leaves fewer resources available to implement the remaining controls (Ponemon, 2017, March). This statistic represents a significant expense for a solution that falls short of expectations, despite its priority ranking in the CSC. According to another recent study, the top three metrics actually used to determine the effectiveness of IS architecture are: return on investment (ROI), decreased security risk, and MTTD of a security breach (Ponemon Institute, 2017, January).

The case study presented in this paper seeks to answer the question: do other technologies exist that could be considered more effective in terms of ROI, decreased security risk, and MTTD of an attack or is a traditional SIEM the best solution for achieving the goals of CSC6, to detect, understand, and recover from an attack?

3. Methods

Smalltown Community Bank began researching potential solutions with the goals of minimizing MTTD and reducing blind spots concerning what commands an attacker executed and what data an attacker accessed from a compromised workstation. Along with this goal came the primary constraint that many small businesses are faced with: no budget. SCB also had to consider the limited resources of only one full-time employee (FTE), who was responsible for both daily IT operations as well as IS. Because of this, an effective solution would need to minimize the effort required to manage and monitor alerts, as well as minimize the amount of time spent troubleshooting and maintaining the solution itself. Due to the size of the environment, three potential solutions were installed and tested in a production environment consisting of 30 workstations and a file server.

3.1.1. Security Onion / ELSA Implementation

For the first candidate, SCB chose the Security Onion (SO) Linux distribution. Designed as a network security monitoring platform, SO includes numerous open-source tools for monitoring, recording and alerting on network activity. SO also includes the Enterprise Log Search and Archive (ELSA) tool, designed to receive, index, and archive syslog messages from various devices on the network. ELSA also provides a web interface for full-text searching of log messages, building and saving queries, and sending alerts based on saved queries.

Because SCB is a small environment, a single virtual server was created to accommodate standalone installation. User workstations and a file server were configured to send Application, Security and System event logs to the SO server using the eventlog-to-syslog utility as recommended by ELSA's creator, Martin Holste. Step by step procedures for installing SO and ELSA can be found on the Production Deployment page of the Security Onion site (Security Onion, 2018).

3.1.2. Windows Event Collection Implementation

To overcome some of the limitations experienced while testing ELSA, SCB designed the second candidate around the built-in Windows Event Collection service and PowerShell. Using examples from the National Security Agency document, *Spotting the Adversary with Windows Event Log Monitoring,* as a starting point, SCB implemented a Windows 2012 R2 virtual machine to act as an event log collector (National Security Agency, 2013). Each workstation and the file server were configured to forward the default Application, Security and System events to the collector. Specific steps for configuring Windows Remote Management, Group Policy and Event Subscriptions are detailed in the *Spotting the Adversary* documentation (National Security Agency, 2013). SCB then created a script using PowerShell to parse the event logs. The parsed events were then used to alert on specific events of interest and provide reports of the aggregated data.

3.1.3. Honeytoken Implementation

Finally, SCB chose to take a step back and examine the assumptions that led to the "need" for a SIEM. More clearly stated, the solution must detect malicious events, alert on those events and provide enough context to take relevant action. To address these concerns, SCB went back to the previous penetration test report to identify the specific events that could have been logged based on the attacker's activity. SCB also identified events that could be distinguished from normal user behavior and verified whether these events were generating adequate log entries and alerts.

Based on the feedback from SANS instructor and course author, Justin Henderson, SCB decided to test the use of honeytokens to detect unauthorized access to data. A honeytoken is a piece of seemingly enticing information that has no useful value and therefore would never be accessed by a normal user (Thompson, 2003). When someone attempts to access a honeytoken object, an alert is triggered. Since a honeytoken object has no legitimate use, all alerts triggered by a honeytoken are, by definition, an anomaly. Recent research on deception technologies shows that respondents ranked the effectiveness of honeytokens at an average score of 7.231 out of 10 (Dominguez, 2017).

The resulting honeytoken candidate was designed to run locally on each workstation and file server. Windows file access auditing was enabled to detect access to strategically placed files within the file system. Access to these files generated an event log entry. The built-in Windows Task Scheduler invoked the PowerShell script each time the specific event ID 4663 was detected, and the script then parsed the relevant details from the log entry and sent an email to an analyst in a presentable format. Detailed procedures to replicate this configuration are included in Appendix A and Appendix B.

4. Findings and Discussion

4.1. Usability

Each of the three candidates, Security Onion, Windows Event Collection, and honeytokens, had its own advantages and shortcomings. As the testing progressed, the needs of the organization and the value of each of the potential solutions became clearer. After initial testing of both the Windows Event Collection and honeytoken candidates, SCB decided to include Windows AppLocker and Sysmon logs to provide additional context and visibility into the actions of the attacker. The following analysis provides details of how each of the three options compared to one another and to the needs of the organization itself.

4.1.1. Security Onion

Security Onion was by far the easiest of the three solutions to install initially. The online documentation and the setup wizard provide a step-by-step process able to be completed in just a few minutes. The fact that SO includes so many tools pre-installed means that the tool is able to provide insight into the monitored network as soon as it is installed. One of these, which SCB found to be very useful, was the Bro framework. By default SO is also configured to ingest the Bro logs directly into ELSA for easy searching. Another intrusion detection tool, Snort, is useful but does require some effort to tune how rules are applied to the environment it is monitoring. For example, the "GPL SNMP public access udp" rule began generating thousands of alerts within the first few hours. Installing the log forwarder, eventlog-to-syslog, on the workstations and file server is very straightforward and ran without any issues. This allowed searching,

correlation and aggregation of the default Windows Application, Security, and System log within ELSA.

Using ELSA to search log data is fairly intuitive. It supports a variety of query functionality such as limiting searches by source IP or destination port or event ID. It also provides access to some interesting plugins via Transforms which allow the result of one query, such as retrieving all destination IPs of outbound web requests, to feed the input of a new query, such as obtaining the related 'whois' results for those IPs. ELSA, through its integration of Bro logs, proved to be very useful for analyzing traffic patterns and correlating network traffic with host event logs.

While SO provided some much-appreciated visibility into the environment, it was not without its disadvantages. This, in part, has to do with the details of the implementation and not SO itself. However, for a small environment, the ease of maintaining the system over time is a key variable for success. There were numerous times when the ELSA database would become corrupt and would need to be purged, losing the previously collected log data. There were issues with the indexing daemon crashing, resulting in unindexed and unsearchable logs when using keyword searches. ELSA also presented limitations in its ability to generate reports with nested result sets, requiring additional time to run individual queries for each IP to display results summed or grouped by attributes like destination port or user agent string. The only disadvantage to using eventlog-to-syslog is that it does not provide a way to include any other logs beside the default Windows Application, Security and System logs.

Overall, the SO candidate provided value to the organization, but it also added a significant amount of complexity to the environment. This is, in part, because of the large number of tools that are included and the integrations between those tools not always working as expected, such as various components of Bro, Syslog-ng, Sphinx, the ELSA parser, and ELSA MySQL database. The ongoing cost of troubleshooting, and occasionally losing data, reduced the value of SO as a solution for this particular use case. SCB also tested the beta version of SO 14.04.5.13 which includes the Elastic stack as an alternative to ELSA and found similar issues with data corruption and lack of reliability. While there are likely fixes for the specific issues observed during this testing,

it must also be taken into consideration the amount of time needed to troubleshoot and maintain a solution like this in a small environment with only one FTE responsible for all IT and IS duties.

4.1.2. Windows Event Collection

After installing Security Onion and experiencing some of the challenges associated with the ELSA search functionality, collecting the Windows events natively within the Windows Event Log and parsing them using PowerShell became an attractive alternative. PowerShell provided the flexibility to generate reports and format them according to the organization's requirements.

Windows Event Collection also provided another benefit in its ability to collect additional event logs such as Windows AppLocker or Microsoft Sysmon. The eventlog-to-syslog utility can only send the default Application, Security, and System logs to a syslog server. Other tools, such as NXLog or OSSEC, could have been used to forward these logs to SO or some other SIEM option, but this would again increase complexity in the environment and be one more cog in the wheel that could require time and effort to install, troubleshoot, and maintain. The Windows Event Collection is built into Windows and can be centrally managed via group policy.

The initial setup was slightly more involved than the SO installation but was still not too burdensome. One issue that came up was related to SCB's use of VDI. After deploying a new base image or refreshing a desktop to the current baseline, each affected desktop required an additional reboot for the GPO to be applied to the computer object with the log forwarding settings. This issue was resolved by setting enforcement of the GPO to the Organizational Unit where the VDI computer objects reside.

Overall, the addition of the AppLocker and Sysmon logs provided value to the organization in that they can alert on activity not contained within the default Application, Security or System logs. The AppLocker logs provided visibility into any executables or scripts that were run from locations outside of the known baseline of applications and scripts. The Sysmon logs allowed greater visibility into the detail of applications an attacker launched, command line parameters used and networks connections established. This allowed alerts to be generated on the unexpected usage of

applications such as “cmd.exe” or “powershell.exe” as well as other built-in Windows utilities typically used by attackers deploying “living off the land” style attacks (Graeber, 2013). While alerting on these items can be beneficial, their usage is not explicitly malicious, and further investigation could be necessary.

During testing, two major disadvantages of the Windows Event Collection candidate were discovered. The first issue was the practical limits of the Windows event log files. Microsoft recommends not exceeding 4GB per log file and no more than 16GB total for all logs (Microsoft, 2015). Even at these maximum settings and in a small environment of thirty workstations, it was only possible to retain a few days’ worth of Security and Sysmon events. The second issue was related to the performance of PowerShell when querying these large event log files. For example, the script used to parse the Sysmon log file and extract the relevant attributes from the Message field, on an event log with approximately 1.7 million records, routinely took between four to five hours to run. Due to these constraints, the log analysis script and alerts were scheduled to run once per day.

4.1.3. Honeytokens

Working through the first two candidates provided valuable insights that allowed the third option to succeed in ways that they did not. The setup for the honeytoken candidate is very simple, but slightly more time consuming than the others to create and distribute the honeytoken files across the workstations and file server. The flexibility of using PowerShell to parse the log files and generate alerts allowed the final solution to be customized to the exact needs of the organization. This solution was extremely effective. When other organizations are measuring MTTD in days, month or even years, SCB is able to minimize this metric to a maximum of a few hours. In terms of ROI, there is virtually no cost regarding ongoing maintenance or troubleshooting. The simplicity of the solution stands in stark contrast to the complexity of even the very basic SIEM solutions that were tested during this case study. Because honeytoken events are rare, there is minimal time required for investigating these alerts.

Honeytokens provide a unique ability to detect malicious activity after other preventive measures have failed but possibly before any critical data has been exposed or

exfiltrated. Honeytokens can be placed in various locations on workstations or servers that are likely to attract the attention of an attacker on an already compromised system, mapped drive, or shared network folder. Placing extraneous folders in the root of the OS drive, or creating additional home directories with honeytoken files inside, and assigning read permissions to the "Everyone" group, is an alluring target. Some of these locations could include mock configuration files where keywords that attackers are likely to search for can be seeded, such as "admin", "username", "user", "password", "passwd", etc. This also could include Office documents and spreadsheets with names like "Annual Bonus Calculations" or "Customer Account List Export" which are likely to catch the attention of an attacker scanning through a hard drive. It is important to place these files in locations that normal users would not peruse to reduce the occurrence of false positives. It is also important to disperse a large enough number of honeytokens throughout the environment to increase the probability that an attacker will trigger an alert.

Because the AppLocker and Sysmon event logs provided greater visibility into the applications and scripts running in the environment, it was easy to identify that the majority of users do not run executables such as "cmd.exe", "powershell.exe", "ipconfig.exe", "ping.exe", "tasklist.exe", or "netsh.exe", to name a few. Because normal, non-adminitrator users do not have a legitimate use for these items, they could also be effectively labeled and used as honeytokens within the environment. It is also worth reviewing this baseline of executables as the tactics, techniques and procedures of attackers are modified in an attempt to evade these detective measures. This is one case where a traditional SIEM or at least centralized logging would be useful in conjunction with the honeytoken solution to see longer-term trends in normal user behavior. The script previously mentioned in Appendix B was parameterized and updated (see Appendix C for a detailed example) to include the generation of alerts based on these items. This new script was then scheduled to run hourly within the Windows Task Scheduler.

4.2. Independent Testing

4.2.1. Red Team Penetration Testing

In 2017, SCB was unable to utilize the DHS RVA service due to the limited availability of DHS resources. Instead, SCB contracted with a reputable IS firm to conduct a network penetration test. Once again, the rules of engagement were spelled out to simulate as much of a real-world attack as possible. The scope included every device connected to the SCB network at both the main office and branch locations. A remotely accessible laptop running Kali was plugged into the physical network and a simulated phishing attack granted access to a user workstation as seen in the previous test. During the five days of testing, numerous alerts were triggered. The most relevant alerts were from Bro and the honeytokens solution. The attacker was unable to connect to the Kali system due to firewall rules blocking outbound Internet access for unknown devices. Exceptions were then granted to the firewall rules to allow testing to continue. The attacker was then unable to access any customer data using the physically-connected Kali system. The testing of the compromised workstation also verified that all of the previously reported vulnerabilities were either remediated or protected through monitoring, alerting and the incident response procedures that allow a suspected compromised workstation to be refreshed to baseline in a matter of minutes using the VDI administrator console. In the end, the attacker was unable to access any customer data without SCB being alerted and triggering the incident response procedures.

4.2.2. Third Party Audit

In conjunction with the red team penetration testing, SCB contracted an audit of the bank's operations for compliance with the Federal Financial Institutions Examination Council (FFIEC) regulations that focus on risk management and the protection of customer data. During the audit, the bank's approach to logging was reviewed. The auditors noted the use of honeytokens as an advanced technique that is not commonly implemented among other banks audited by this firm. The honeytoken alerts also provided evidence of an automated log review process. However, the honeytokens alone did not satisfy all of the FFIEC logging

requirements, but that the use of either ELSA or the Windows Event Collection met the requirements regarding centralized logging and event log retention.

4.2.3. Federal Reserve Compliance Examination

In accordance with the FFIEC examination schedule, SCB was also examined for IT and IS compliance by the Federal Reserve in 2017. As part of this examination, the Federal Reserve examiners focused on the bank's responses to the newly issued FFIEC Cybersecurity Assessment Tool (FFIEC, 2017, May). The examiners found the bank's log practices to be more than adequate given the size and risk profile of the organization. The examiners also noted the bank's implementation of honeytokens to be considered innovative compared not only to other banks of similar size but even among much larger banks.

5. Recommendations and Implications

Dr. Eliyahu Goldratt taught that "Technology can bring benefits, if and only if it diminishes a limitation" (Goldratt, 2005). Along with this maxim, he provided a framework to evaluate how much benefit can be expected based on the following four questions:

1. What is the power of the technology?
2. What limitation does it diminish?
3. What rules helped us accommodate the limitation?
4. What rules should we use now?

When applying these rules to the implementation of a SIEM, the answers start to become quite elusive. Many vendors include different functionality within SIEM products, from event log collection, to event aggregation, to correlation, to normalization of the data format, to event enrichment to add context to security events, to search and pivoting, to reporting and alerting. However, there is no clear consensus on which combination or interpretation of these functions actually defines a SIEM or if products labeled as something else, such as network security monitoring (NSM) or event log management (ELM) tools which provide some of these functions, should be considered a SIEM.

In terms of CSC6, the power of the technology must be understood as the functions that enable an organization to "detect, understand, or recover from an attack" (CIS, 2016). To accomplish this goal, the technology must first create the log entries that properly identify malicious behavior. Very few technology vendors, if any, design the generation of event logs in such a way to distinguish between "authorized" or "malicious" user behaviors. Even in a small environment, basic login failure events are an everyday occurrence and do not necessarily indicate malicious activity. Furthermore, events that recur every single day are by definition, not anomalous. The use of honeytokens, as part of a well-planned defense in depth strategy, generates events that are specifically anomalous, and most likely malicious.

The limitation inherent in SIEM technologies is that SIEM can only alert on the events that it ingests, the vast majority of which are benign events. To accommodate this limitation, many SIEM vendors, along with the security standards and regulations, promote the approach of collecting every possible log and alerting on some sort of fancy correlation algorithm regardless of the value of each individual event message. The fallacy of this method is that collecting more events for the sake of establishing the context of the environment does not inherently improve visibility into the environment. Collecting more events could very well be minimizing the value of the data collected because of its lack of uniqueness (Ponemon Institute, 2017, March). Frequently, this approach also requires expensive hardware capable of receiving and processing large quantities of events per second (EPS), along with enough overhead to handle even higher bursts of EPS. This approach inevitably leads to blind spots for the analyst tasked with spotting malicious activity. The honeytoken solution put forward in this study diminishes these limitations by focusing specifically on the generation of rare events.

The set of rules that many organizations implicitly operate within assume that there is value in collecting and retaining all event logs, even if merely for compliance purposes. This mode of operation then attempts to compensate for the limitation that malicious activity is difficult to distinguish from normal user

behavior by either generating false positives which create an enormous overhead for the IS team to investigate, or ignoring potentially malicious events via false negatives (Ponemon Institute, 2017, March). The rules then must include subordinate assumptions that the IS team either must work through an infinite backlog of false positive alerts or blindly trust that the environment is secure when in fact it is compromised. Both of these states drive up total information security costs and increase MTTD.

In contrast, the rules that allow the host-based honeytoken solution to be so effective, focus on and prioritize rare events by setting up scenarios that generate the necessary logs from those events. There are several ways to go about this process, either by implementing honeytokens centrally on file shares or locally on user workstations, preferably through a centrally -managed image. Once this has been established, the monitoring and alerting function can truly identify anomalous activity, and the location of the honeytokens can be adjusted if false positives become too frequent.

5.1. Tailoring the Solution for the Organization

Neither a SIEM nor a honeytoken implementation is a panacea to fix the problem of detecting a security breach. However, it is worth examining whether the requirements that most organizations blindly follow for compliance reasons actually perform the expected task and provide value to the organization. It is also important to note that these solutions are not mutually exclusive. The findings of this case study are presented in such a way as to contrast the assumptions that underlie a typical SIEM implementation, and provide a working example of how a result of equal or better quality can be achieved with minimal cost and can improve the organization's return on investment, at least in terms of the IS budget using this simple honeytoken implementation.

According to Don Murdoch, author of the Blue Team Handbook, "a SIEM is more valuable if the analyst understands how to produce the results from the source system itself" (Murdoch, 2018). In other words, it is more important to point

the existing security cameras in the directions most likely to catch the bad guy, instead of simply adding more cameras. In this case study, the honeytoken solution fits well with the organization's budgetary constraints, IS resource constraints and existing tactics, techniques and procedures. Based on SCB's existing IR procedures and VDI environment, the organization prioritized mean time to recovery (MTTR) by instantly redeploying suspect desktop, versus spending the time to conduct a full forensic analysis for every alert that a typical SIEM might generate.

5.2. Implications for Future Research

The first option to further this research would be to test the honeytoken implementation to see if similar results could be replicated. Because this case study was based on a qualitative assessment in a small environment, it could also be beneficial to test the honeytoken implementation in a larger organization to see how these results would scale. Aside from the propagation of this particular honeytoken implementation, other case studies on the specific use cases for honeytokens could help drive further adoption of these techniques.

Other opportunities for further research exist in creating more low-cost, yet reliable, SIEM solutions for small organizations that would not require as many ongoing operational resources. And finally, the results of the Ponemon surveys prove that current SIEM implementations are either not properly diminishing the limitations that the technology is promising, or organizations have not yet discovered the proper rules of operation for their SIEM to provide the expected value. The state of information security could greatly benefit from more case studies that can demonstrate the true value and ROI of a SIEM to help detect, understand, or recover from an attack.

6. Conclusion

Many organizations have implemented a SIEM to comply with the logging requirements of various security standards and regulations, only to find that it does not meet their information security expectations. In some organizations, this has given rise to the recent growth of "tactical SIEM" implementations that focus on collecting specific logs for use by an IS team or security operations center (SANS,

2018). However, 68 percent of organizations surveyed claimed that additional staff would be required to maximize the value of their SIEM implementation (Ponemon Institute, 2017, March). If organizations are failing to extract the expected value out of one SIEM, it certainly begs the question of how these organizations are going to manage the added workload and complexity of multiple SIEMs in the environment.

Putting the right information security architecture into practice within an organization can be a daunting challenge. One recent research study claimed that deployment of honey technologies is "limited given their benefits" and that "honey technologies are viewed as desirable but difficult to implement" (Dominguez, 2017). While both of these statements may be true in the broader context of deception technologies as a whole, it is important to drill down into the distinction between solutions such as honeypots and honeytokens. The information generated by a honeypot has limited value unless that information can be acted upon to implement better preventive or detective defenses. The implementation of honeytokens focuses specifically on the generation of events for detective purposes.

When faced with the challenge of prioritizing resources to achieve an organization's IT and IS objectives, there will always be trade-offs between improving the organization's return on investment, decreasing security risk, and decreasing mean time to detection of a potential security breach. However, there lessons to be learned in how requirements are defined and the outcomes they are intended to generate. By carefully examining the context of the requirements that traditional solutions are built upon, new and effective approaches can be created that are not bound by the assumptions that lead to the previous status quo. Evidence of this process can even be seen in the evolution of the security standards and regulations themselves. As this paper was being written, the Center for Internet Security released version seven of the Critical Security Controls. Several of the issues raised by this paper are addressed in this latest version, specifically control 6.5, which allows organizations the ability to determine the value of specific logs and log entries and customize and tune their SIEM deployment in accordance with their needs. It was exactly this process that directed this case study to examine the value of implementing

honeytokens. The result is that honeytokens are not inherently difficult to implement, and can provide significant ROI in comparison to, and also in conjunction with, a SIEM implementation.

References

Brenton, C., Bird, T., & Ranum, M. (2006). Top 5 Essential Log Reports Version 1.0. Retrieved from https://www.sans.org/security-resources/top5-logreports.pdf

Center for Internet Security. (2016). CIS Control 6 Maintenance, Monitoring, and Analysis of Audit Logs. Retrieved from https://www.cisecurity.org/controls/maintenance-monitoring-and-analysis-of-audit-logs/

Czanik, P. (2013). The 6 Categories of Critical Log Information. Retrieved from http://www.sans.edu/research/security-laboratory/article/sixtoplogcategories

Federal Financial Institutions Examination Council (FFIEC). (2017). FFIEC IT Examination Handbook Infobase. Retrieved from https://ithandbook.ffiec.gov/it-booklets/information-security/

Federal Financial Institutions Examination Council (FFIEC). (2017, May). Cybersecurity Assessment Tool. Retrieved from https://www.ffiec.gov/cyberassessmenttool.htm

Dominguez, A. (2017, November). The State of Honeypots: Understanding the Use of Honey Technologies Today. Retrieved from https://www.sans.org/reading-room/whitepapers/detection/state-honeypots-understanding-honey-technologies-today-38165

Earnest, W. (2016, August). Investing in Information Security: A Case Study in Community Banking. Retrieved from https://www.sans.org/reading-room/whitepapers/leadership/investing-information-security-case-study-community-banking-37167

Finkle, J. & Heavey, S. (2014, March 13). Target Says It Declined to Act on Early Alert of Cyber Breach. Retrieved from https://www.reuters.com/article/us-target-breach/target-says-it-declined-to-act-on-early-alert-of-cyber-breach-idUSBREA2C14F20140313

Goldratt, E. (2005). Beyond the Goal. Gildan Media.

Graeber, M. (2013, September 14) Living Off the Land: A Minimalist's Guide to Windows Post-Exploitation. Retrieved from: http://obscuresecurity.blogspot.com/p/presentation-slides.html

Mandiant. (2016). M-Trends 2016. Retrieved from https://www2.fireeye.com/rs/848-DID-242/images/Mtrends2016-NEW.pdf

Mandiant. (2017). M-Trends 2017 A View From the Front Lines. Retrieved from https://www.fireeye.com/current-threats/annual-threat-report/mtrends.html

Microsoft. (2015, August 14).Recommended Settings for Event Log Sizes in Windows. Retrieved from: https://support.microsoft.com/en-us/help/957662/recommended-settings-for-event-log-sizes-in-windows

Murdoch, D. (2018, May 23). Personal Interview.

National Security Agency. (2013, December 16). Spotting the Adversary with Windows Event Log Monitoring. Retrieved from https://www.iad.gov/iad/library/reports/spotting-the-adversary-with-windows-event-log-monitoring.cfm

Payment Card Industry (PCI). (2016, April). Payment Card Industry (PCI) Data Security Standard, v3.2. Retrieved from

https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2.pdf?agreement=true&time=1516382952033

Ponemon Institute. (2017, January). The Cost & Consequences of Security Complexity. Retrieved from https://www.mobileiron.com/sites/default/files/analysis-report/files/MobileIron%20Report%20on%20Complexity%20Final%202.pdf

Ponemon Institute. (2017, March). Challenges to Achieving SIEM. Retrieved from: Optimization http://go.cyphort.com/rs/181-NTN-682/images/Cyphort-Ponemon-SIEM-Report.pdf

SANS (2018). SEC555: SIEM with Tactical Analytics. Retrieved from https://www.sans.org/course/siem-with-tactical-analytics

Security Onion (2018). Production Deployment. Retrieved from https://github.com/Security-Onion-Solutions/security-onion/wiki/ProductionDeployment

Tarala, J. (2014, April). SEC566 Implementing and Auditing Critical Security Controls. SANS Orlando 2014.

Thompson, N. (2003, April 28). New Economy; The "honeytoken," an innocuous tag in a file, can signal an intrusion in a company's database. Retrieved from: http://www.nytimes.com/2003/04/28/business/new-economy-honeytoken-innocuous-tag-file-can-signal-intrusion-company-s.html

Trustwave. (2017). Trustwave Global Security Report. Retrieved from https://www2.trustwave.com/rs/815-RFM-693/images/2017%20Trustwave%20Global%20Security%20Report-FINAL-6-20-2017.pdf

Wes Ea)gmail.com

Williams, A. (2005-05-02). Improve IT Security With Vulnerability Management. Retrieved from https://www.gartner.com/doc/480703/improve-it-security-vulnerability-management

Verizon. (2014). 2014 Data Breach Investigations Report. Retrieved from http://www.verizonenterprise.com/DBIR/2014/

Appendix A

Honeytoken Setup Procedures:

1. Create a new Group Policy Object (GPO) that will be used to push the logging settings to systems where alerts will be generated.
2. Assign the following settings:
    a. Computer Configuration → Policies → Windows Settings → Security Settings → Local Policies → Audit Policy → Audit object access = Success, Failure
    b. Computer Configuration → Policies → Windows Settings → Security Settings → Advanced Audit Configuration → Object Access → Audit File Share = Success, Failure
    c. Computer Configuration → Policies → Windows Settings → Security Settings → Advanced Audit Configuration → Object Access → Audit File System = Success, Failure

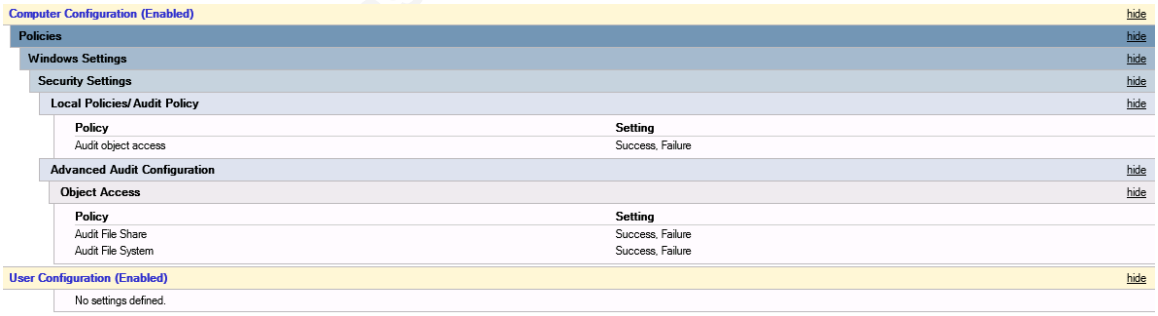

Figure 1: Honeytoken GPO Settings Example

3. Apply the GPO to an OU and/or set the Scope of the GPO to include the Computer objects where the GPO will be applied.
4. Create a file on the system to be monitored.
    a. Example: Office documents or plain text config files are great options that can be seeded with bogus data. The files can be named something enticing such as "Annual Bonus.xlsx" or "Customer Database Extract.xlsx" or "admin.config". The seed data should include keywords that an attacker is likely to search for when crawling the network, such as "username", "password", "user", "admin", "passwd", etc.
5. Save the file
6. Right click on the new file and select "Properties"
7. Go to the "Security" tab
8. Click on the "Advanced" button
9. Click on the "Auditing" tab

10. Click the "Edit" button
11. Click the "Add" button
12. Enter "Everyone" in the textbox and click "OK"
13. Select the types of events to be audited.
    a. Note: If the file is in a directory that normal users browse on a regular basis it may be best to uncheck the "List folder/read data" option.
14. Click "OK"
15. Click "OK"
16. Click "OK"
17. Click "OK"
18. Next, open the file
19. Then open the Windows Event Viewer
20. Go to the Security log
21. Search for Event ID 4663 to verify that the log entry is being generated correctly. If not, verify that the GPO is getting applied successfully.

Monitoring and Alerting Setup Procedures:

If there is already an existing SEIM in place and Windows Security logs are being forwarded to the SEIM, the setup an alert for Windows Security log Event ID 4663. However, it is important to note that this solution does not require a SEIM to implement. Even if there is not a SEIM running in the environment, alerts can still be generated by these events using Windows Task Scheduler and Powershell.

1. Create a folder on the system to be monitored, such as C:\Scripts
2. Edit the NTFS permissions of the folder to only allow administrators access to the files in the folder
3. Place the Get-HoneyTokenAlerts.ps1 script in the folder
    a. See Appendix B for Get-HoneyTokenAlerts.ps1 code
    b. Be sure to fill in the $email_to, $email_from, and $email_server variables, as well as any filters necessary to avoid false positive alerts
4. Open Task Manager and create a new basic task with the following settings:
    a. Under the General tab, set the task to run as "SYSTEM"
    b. Under the Triggers tab select "When a specific event is logged"
    c. Set the Log as Security and the Event ID as 4663

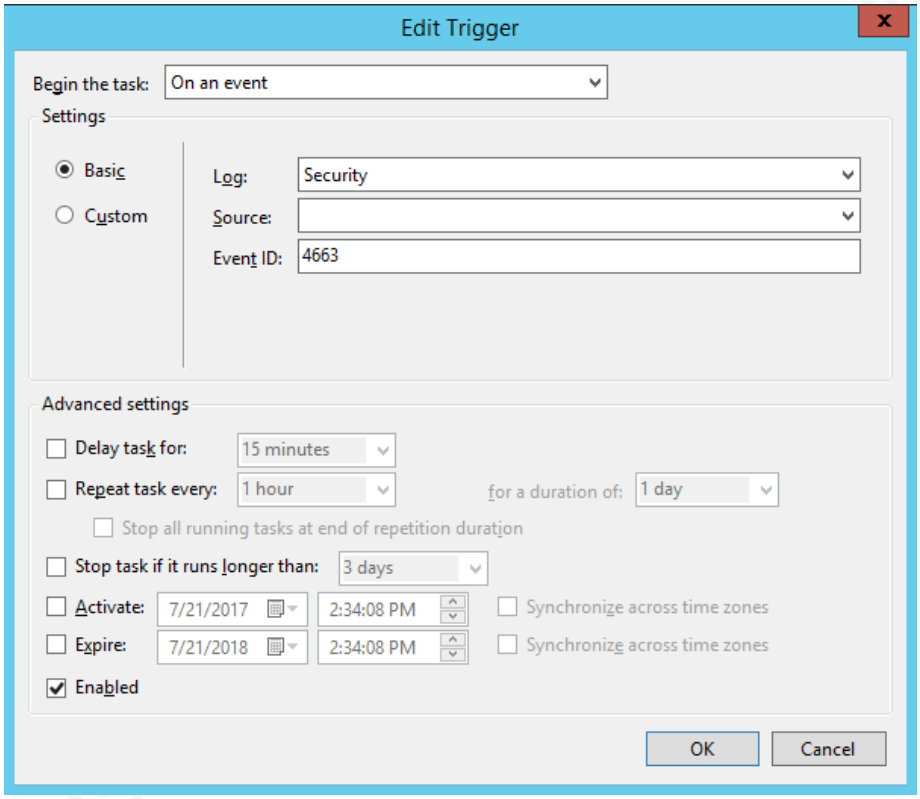

Figure 2: Task Scheduler Event Trigger

d. Under the Actions tab, select "Start a program"
e. Enter "powershell.exe" as the Program/script
f. Enter "-ExecutionPolicy Unrestricted C:\Scripts\Get-HoneyTokenAlerts.ps1" as the Add arguments box

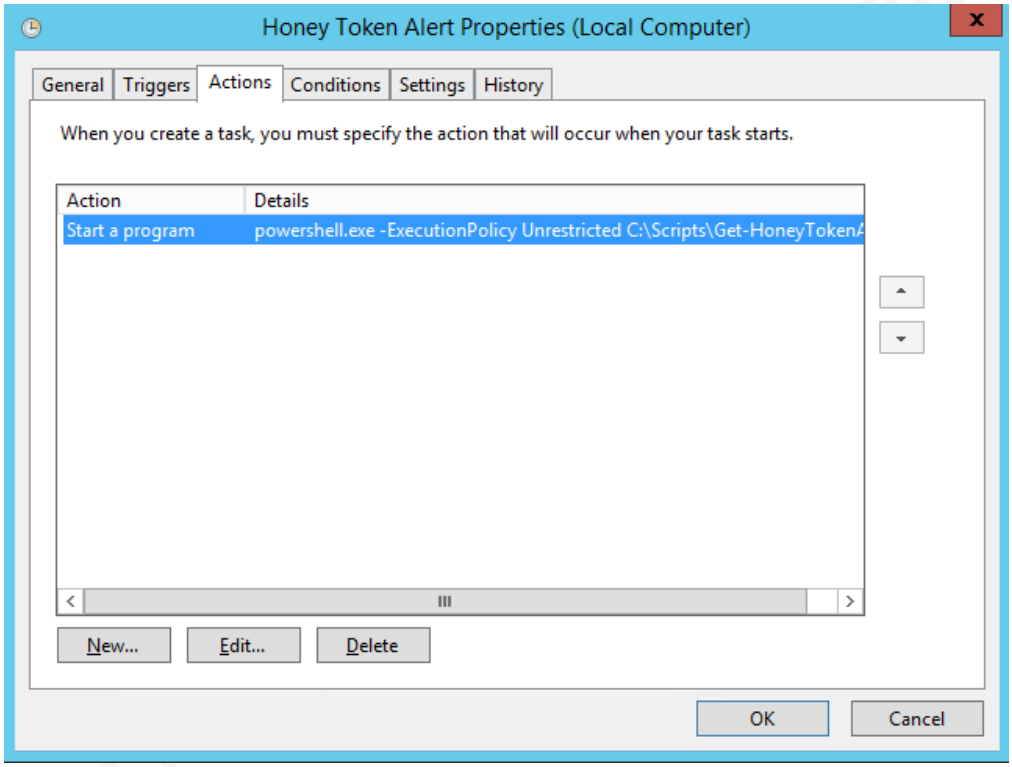

Figure 3: Task Scheduler Actions

g. Click "OK' to save the new task

Appendix B

Get-EventLogAlerts.ps1

```
$email_to = "<INSERT_TO_EMAIL_HERE>"
$email_from = "<INSERT_EMAIL_HERE>"
$email_server = "<INSERT_MAIL_SERVER_NAME_HERE>"
$email_subject = "**ALERT** Honey Token Events"

$dateStart = [DateTime]::Now.AddMinutes(-1)
$out = @()
$evnts = Get-WinEvent -FilterHashtable @{logname="security";id="4663"} | `
    Where-Object {$_.TimeCreated -ge $dateStart}

foreach($evt in $evnts) {
    $xml = [xml]$evt.ToXml()

    $builder = New-Object System.Object
    $builder | Add-Member -Type NoteProperty -Name Timestamp `
               -Value $evt.TimeCreated
    $builder | Add-Member -Type NoteProperty -Name ServerName `
               -Value $xml.Event.EventData.Data[2].'#text'
    $builder | Add-Member -Type NoteProperty -Name EventID `
               -Value $evt.id
    $builder | Add-Member -Type NoteProperty -Name UserName `
               -Value $xml.Event.EventData.Data[1].'#text'
    $builder | Add-Member -Type NoteProperty -Name ObjectName `
               -Value $xml.Event.EventData.Data[6].'#text'
    $builder | Add-Member -Type NoteProperty -Name ProcessName `
               -Value $xml.Event.EventData.Data[11].'#text'

                    # Example to filter out specific files or directories
    $out += $builder | Where-Object {($_.ObjectName.Substring(0,18) -ne "C:\Windows\WinSxS\") `
                     -and ($_.ObjectName -ne "<INSERT_PATH_TO_EXCLUDE_FROM_ALERTING>")`
                     -and ($_.ProcessName -ne "<INSERT_PROCESS_TO_EXCLUDE_FROM_ALERTING>")}
}

if($out){

    $css = "<style>BODY{font-family: Arial; font-size: 10pt;} TABLE{border: 1px solid black;
border-collapse: collapse;} TH{border: 1px solid black; background: #dddddd; padding: 5px;}
TD{border: 1px solid black; padding: 5px;}</style>"

    Send-MailMessage -To $email_to -From $email_from -SmtpServer $email_server -Subject
$email_subject -BodyAsHtml ($out | ConvertTo-Html -Head $css | Out-String)

}
```

Wes Earnest, wes.earnest@gmail.com

Appendix C

Get-HoneyTokenAlerts.ps1

```
Param(
[Parameter(Mandatory=$True)]
[string]$LogFile,
[string]$EventId,
[string]$TimeSpan
)

#################################################################################
# Example Alerts:

# $LogFile = Security
# $EventId = 4663   # File Auditing Event (Honey Token)
# $TimeSpan = 1

# $LogFile = "Microsoft-Windows-AppLocker/EXE and DLL"
# $EventId = "8003" # AppLocker Executable
# $TimeSpan = 60

# $LogFile = "Microsoft-Windows-AppLocker/MSI and Script"
# $EventId = "8006" # AppLocker Script
# $TimeSpan = 60

# $LogFile = "Microsoft-Windows-Sysmon/Operational"
# $EventId = "1"    # Suspicious Process Creation
# $TimeSpan = 60

# $LogFile = "Microsoft-Windows-Sysmon/Operational"
# $EventId = "3"    # Suspicious Powershell Network Connection
# $TimeSpan = 60
#################################################################################

$email_to = "<INSERT_TO_EMAIL_HERE>"
$email_from = "<INSERT_EMAIL_HERE>"
$email_server = "<INSERT_MAIL_SERVER_NAME_HERE>"
$email_subject = "**ALERT** Test Message"

if ($LogFile -eq "Security") {
    $LogFile = "Security"
} elseif ($LogFile -eq "AppLocker-EXE") {
    $LogFile = "Microsoft-Windows-AppLocker/EXE and DLL"
} elseif ($LogFile -eq "AppLocker-Script") {
    $LogFile = "Microsoft-Windows-AppLocker/MSI and Script"
} elseif ($LogFile -eq "Sysmon") {
    $LogFile = "Microsoft-Windows-Sysmon/Operational"
}

$dateStart = [DateTime]::Now.AddMinutes(-$TimeSpan)
$evnts = Get-WinEvent -FilterHashtable @{logname="$LogFile";id="$EventId"} | Where-Object
{$_.TimeCreated -ge $dateStart}
$out = @()

if ($evnts.Count -gt 0) {

    if ($LogFile -eq "Security"){
        $email_subject = "**ALERT** Honey Token Events"

        foreach($evt in $evnts) {
            $xml = [xml]$evt.ToXml()

            $builder = New-Object System.Object
            $builder | Add-Member -Type NoteProperty -Name Timestamp -Value $evt.TimeCreated
            $builder | Add-Member -Type NoteProperty -Name ServerName -Value
```

Wes Earnest, wes.earnest@gmail.com

```
$xml.Event.EventData.Data[2].'#text'
            $builder | Add-Member -Type NoteProperty -Name EventID -Value $evt.id
            $builder | Add-Member -Type NoteProperty -Name UserName -Value
$xml.Event.EventData.Data[1].'#text'
            $builder | Add-Member -Type NoteProperty -Name ObjectName -Value
$xml.Event.EventData.Data[6].'#text'
            $builder | Add-Member -Type NoteProperty -Name ProcessName -Value
$xml.Event.EventData.Data[11].'#text'

            $out += $builder
        }
    }

    elseif ($LogFile -eq "Microsoft-Windows-AppLocker/EXE and DLL") {
        $email_subject = "**ALERT** AppLocker Events - EXE"
        $out = $evnts | Select-Object TimeCreated,Id,Message,MachineName,UserId
    }

    elseif ($LogFile -eq "Microsoft-Windows-AppLocker/MSI and Script") {
        $email_subject = "**ALERT** AppLocker Events - Script"
        $out = $evnts | Select-Object TimeCreated,Id,Message,MachineName,UserId
    }

    elseif ($LogFile -eq "Microsoft-Windows-Sysmon/Operational") {
        if ($EventId -eq 1) {
            foreach($evt in $evnts) {
                $xml = [xml]$evt.ToXml()

                if ( $xml.Event.EventData.Data[4].'#text'.Length -gt 500 -or
$xml.Event.EventData.Data[15].'#text'.Length -gt 500) {

                    $builder = New-Object System.Object
                    $builder | Add-Member -Type NoteProperty -Name Timestamp -Value
$evt.TimeCreated
                    $builder | Add-Member -Type NoteProperty -Name Hostname -Value
$evt.MachineName
                    $builder | Add-Member -Type NoteProperty -Name ProcessID -Value
$xml.Event.EventData.Data[2].'#text'
                    $builder | Add-Member -Type NoteProperty -Name Image -Value
$xml.Event.EventData.Data[3].'#text'
                    $builder | Add-Member -Type NoteProperty -Name CommandLine -Value
$xml.Event.EventData.Data[4].'#text'
                    $builder | Add-Member -Type NoteProperty -Name CurrentDirectory -Value
$xml.Event.EventData.Data[5].'#text'
                    $builder | Add-Member -Type NoteProperty -Name User -Value
$xml.Event.EventData.Data[6].'#text'
                    $builder | Add-Member -Type NoteProperty -Name Hashes -Value
$xml.Event.EventData.Data[11].'#text'
                    $builder | Add-Member -Type NoteProperty -Name ParentProcessId -Value
$xml.Event.EventData.Data[13].'#text'
                    $builder | Add-Member -Type NoteProperty -Name ParentImage -Value
$xml.Event.EventData.Data[14].'#text'
                    $builder | Add-Member -Type NoteProperty -Name ParentCommandLine -Value
$xml.Event.EventData.Data[15].'#text'

                    $email_subject = "**ALERT** Sysmon Events - Command Line Length"
                    $out += $builder
                }

                if ($xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\net1.exe" -or
                    $xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\whoami.exe" -or
                    $xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\ipconfig.exe" -
or
                    $xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\route.exe" -or
                    $xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\arp.exe" -or
                    $xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\netstat.exe" -or
                    $xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\netsh.exe" -or
                    $xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\tasklist.exe" -
or
```

Wes Earnest, wes.earnest@gmail.com

```
                    $xml.Event.EventData.Data[3].'#text' -eq "C:\Windows\System32\wmic.exe" -or
                    $xml.Event.EventData.Data[3].'#text' -like "*wscript.exe*" -or
                    $xml.Event.EventData.Data[3].'#text' -like "*dropbox*" -or
                    $xml.Event.EventData.Data[3].'#text' -like "*bomgar*" -or
                    $xml.Event.EventData.Data[3].'#text' -like "*webex*" -or
                    $xml.Event.EventData.Data[3].'#text' -like "*skype*" -or
                    $xml.Event.EventData.Data[3].'#text' -like "*citrix*" -or
                    $xml.Event.EventData.Data[3].'#text' -like "*teamviewer*")
                {

                    $builder = New-Object System.Object
                    $builder | Add-Member -Type NoteProperty -Name Timestamp -Value
$evt.TimeCreated
                    $builder | Add-Member -Type NoteProperty -Name Hostname -Value
$evt.MachineName
                    $builder | Add-Member -Type NoteProperty -Name ProcessID -Value
$xml.Event.EventData.Data[2].'#text'
                    $builder | Add-Member -Type NoteProperty -Name Image -Value
$xml.Event.EventData.Data[3].'#text'
                    $builder | Add-Member -Type NoteProperty -Name CommandLine -Value
$xml.Event.EventData.Data[4].'#text'
                    $builder | Add-Member -Type NoteProperty -Name CurrentDirectory -Value
$xml.Event.EventData.Data[5].'#text'
                    $builder | Add-Member -Type NoteProperty -Name User -Value
$xml.Event.EventData.Data[6].'#text'
                    $builder | Add-Member -Type NoteProperty -Name Hashes -Value
$xml.Event.EventData.Data[11].'#text'
                    $builder | Add-Member -Type NoteProperty -Name ParentProcessId -Value
$xml.Event.EventData.Data[13].'#text'
                    $builder | Add-Member -Type NoteProperty -Name ParentImage -Value
$xml.Event.EventData.Data[14].'#text'
                    $builder | Add-Member -Type NoteProperty -Name ParentCommandLine -Value
$xml.Event.EventData.Data[15].'#text'

                    $email_subject = "**ALERT** Sysmon Events - Suspicious Process Creation"
                    $out += $builder
                }
            }
        }

        elseif ($EventId -eq 3) {
            foreach($evt in $evnts) {
                $xml = [xml]$evt.ToXml()

                # Exclude execution of this script from the alerts
                if ($xml.Event.EventData.Data[3].'#text' -like "*powershell.exe*" -and
                    ($xml.Event.EventData.Data[13].'#text' -ne <INSERT_EMAIL_SERVER_IP_HERE> -and
$xml.Event.EventData.Data[15].'#text' -ne "25")) {

                    $builder = New-Object System.Object
                    $builder | Add-Member -Type NoteProperty -Name Timestamp -Value
$evt.TimeCreated
                    $builder | Add-Member -Type NoteProperty -Name Hostname -Value
$evt.MachineName
                    $builder | Add-Member -Type NoteProperty -Name ProcessID -Value
$xml.Event.EventData.Data[2].'#text'
                    $builder | Add-Member -Type NoteProperty -Name Image -Value
$xml.Event.EventData.Data[3].'#text'
                    $builder | Add-Member -Type NoteProperty -Name User -Value
$xml.Event.EventData.Data[4].'#text'
                    $builder | Add-Member -Type NoteProperty -Name Protocol -Value
$xml.Event.EventData.Data[5].'#text'
                    $builder | Add-Member -Type NoteProperty -Name SourceIP -Value
$xml.Event.EventData.Data[8].'#text'
                    $builder | Add-Member -Type NoteProperty -Name SourceHostname -Value
$xml.Event.EventData.Data[9].'#text'
                    $builder | Add-Member -Type NoteProperty -Name DestinationIP -Value
$xml.Event.EventData.Data[13].'#text'
                    $builder | Add-Member -Type NoteProperty -Name DestinationPort -Value
```

```
$xml.Event.EventData.Data[15].'#text'

                    $email_subject = "**ALERT** Sysmon Events - Suspicious Powershell Network
Connection"
                    $out += $builder
                }
            }
        }
    }

    if ($out) {
        $css = "<style>BODY{font-family: Arial; font-size: 10pt;} TABLE{border: 1px solid black;
border-collapse: collapse;} TH{border: 1px solid black; background: #dddddd; padding: 5px;}
TD{border: 1px solid black; padding: 5px;}</style>"
        Send-MailMessage -To $email_to -From $email_from -SmtpServer $email_server -Subject
$email_subject -BodyAsHtml ($out | ConvertTo-Html -Head $css | Out-String)
    }
}
```