2017-03

Cyber indicators of compromise: a domain ontology for security information and event management

Rowell, Marsha D.

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CYBER INDICATORS OF COMPROMISE: A DOMAIN ONTOLOGY FOR SECURITY INFORMATION AND EVENT MANAGEMENT

by

Marsha D. Rowell

March 2017

Thesis Co-Advisors: J. D. Fulp
Gurminder Singh

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE CYBER INDICATORS OF COMPROMISE: A DOMAIN ONTOLOGY FOR SECURITY INFORMATION AND EVENT MANAGEMENT	**5. FUNDING NUMBERS**
6. AUTHOR(S) Marsha D. Rowell	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

It has been said that cyber attackers are attacking at wire speed (very fast), while cyber defenders are defending at human speed (very slow). Researchers have been working to improve this asymmetry by automating a greater portion of what has traditionally been very labor-intensive work. This work is involved in both the monitoring of live system events (to detect attacks), and the review of historical system events (to investigate attacks). One technology that is helping to automate this work is Security Information and Event Management (SIEM). In short, SIEM technology works by aggregating log information, and then sifting through this information looking for event correlations that are highly indicative of attack activity. For example: Administrator successful local logon and (concurrently) Administrator successful remote logon. Such correlations are sometimes referred to as indicators of compromise (IOCs). Though IOCs for network-based data (i.e., packet headers and payload) are fairly mature (e.g., Snort's large rule-base), the field of end-device IOCs is still evolving and lacks any well-defined go-to standard accepted by all. This report addresses ontological issues pertaining to end-device IOCs development, including what they are, how they are defined, and what dominant early standards already exist.

14. SUBJECT TERMS IOCs, events, rules, incident, SIEM, CANES, NetIQ	**15. NUMBER OF PAGES** 103
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

CYBER INDICATORS OF COMPROMISE: A DOMAIN ONTOLOGY FOR SECURITY INFORMATION AND EVENT MANAGEMENT

Marsha D. Rowell
Lieutenant, United States Navy
B.S., Auburn University, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2017

Approved by: J. D. Fulp
Co-Advisor

Dr. Gurminder Singh
Co-Advisor

Dr. Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

It has been said that cyber attackers are attacking at wire speed (very fast), while cyber defenders are defending at human speed (very slow). Researchers have been working to improve this asymmetry by automating a greater portion of what has traditionally been very labor-intensive work. This work is involved in both the monitoring of live system events (to detect attacks), and the review of historical system events (to investigate attacks). One technology that is helping to automate this work is Security Information and Event Management (SIEM). In short, SIEM technology works by aggregating log information, and then sifting through this information looking for event correlations that are highly indicative of attack activity. For example: Administrator successful local logon and (concurrently) Administrator successful remote logon. Such correlations are sometimes referred to as indicators of compromise (IOCs). Though IOCs for network-based data (i.e., packet headers and payload) are fairly mature (e.g., Snort's large rule-base), the field of end-device IOCs is still evolving and lacks any well-defined go-to standard accepted by all. This report addresses ontological issues pertaining to end-device IOCs development, including what they are, how they are defined, and what dominant early standards already exist.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AS&W	Attack Sensing and Warning
CAPEC	Common Attack Pattern Enumeration
CEE	Common Event Expression
CIRT	Computer Incident Response Team
CJCSM	Chairman of the Joint Chiefs of Staff Manual
CND	Computer Network Defense
CNDSP	Computer Network Defense Service Provider
COA	Courses of Action
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DDOS	Distributed Denial of Service
DLP	Data Loss Protection
FQDN	Fully Qualified Domain Name
IDS	Intrusion Detection System
IDSP	Intrusion Detection and Prevention System
IM	Information Management
IOCs	Indicators of Compromise
IS	Information System
ISAC	Information Sharing and Analysis Center
ISP	Internet Service Provider
IR	Incident Response
LM	Log Management
MAEC	Malware Attribute Enumeration and Characterization
NIDS	Network Intrusion Detection System
NIPS	Network Intrusion Prevention System
NVD	National Vulnerability Database
OVAL	Open Vulnerability and Assessment Language
PRG	Processing Rule Group
SIEM	Security Information and Event Management
S/N	Signal to Noise

SM	Security Manager
TTPs	Tactics Techniques and Procedures
VB	Visual Basic
WMI	Windows Management Instrumentation

ACKNOWLEDGMENTS

I would like to thank J. D. Fulp, my thesis advisor, for his professional guidance and expertise, for helping me maintain focus, and most of all, for his patience throughout this process. Thank you also goes to Dr. Gurminder Singh, my second thesis advisor, for helping me make it through this long journey.

To my best friend, Michelle Joseph, who consistently drives me to succeed, and whom I owe many thanks for providing motivation and support during this long project. A heartfelt “thank you” for pushing me to succeed and never give up, despite the many setbacks along the way.

To my family: Lois Rowell, Kelly Rowell, Kenneth Rowell, Keith Rowell, and Mike Watford, I love you and thank you for your continued love and support throughout my many endeavors.

Finally, I would like to thank NCDOC for the outstanding visit and wealth of information they provided to help me in my research. Thank you for taking the time to answer my many questions and helping to point me in the right direction.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. CYBER WAR

Cyberspace as defined in JP 3-12, is “a global domain within the information environment consisting of the interdependent network of information technology infrastructures and resident data, including the internet, telecommunications networks, computer systems, and embedded processors and controllers” [1]. It is a very complex, dynamic virtual environment facilitated by a ubiquitous digital global infrastructure. It provides the perfect setting for the day-to-day execution of processes in commerce, communication, government, military, utilities, and education. Threats to cyberspace, or a cyber threat, originate from individuals who try to access a network and/or a control system device using a data communications path without authorization [2]. Threat actors arise from many different sources and motives, including disgruntled employees, terrorist groups, those seeking monetary gain, those pushing a political or philosophical point of view, and adversarial governments.

Cyber threats during the 2013–2015 timeframe were ranked as the top strategic threat against the United States, even placing ahead of terrorism [3]. Threat actors have invested considerably in cyberspace, as it affords them with a practical, credibly deniable skillset with which to target the U.S. and cause damage to its interests. The use of cyber to cause harm to our national security interests has been a concern going back to at least the 1990s [4]. The measure and scope of these threats reveal that 97 percent of Fortune 500 businesses have been hacked, and over a hundred governments are preparing for battles in this virtual domain. These problems have caused political concerns that are exemplified through events such as the WikiLeaks scandals, new cyberweapons like Stuxnet, domestic monitoring by NSA, individual concerns over personal privacy, and social networking’s role in events such as the Arab Spring revolutions. President Barack Obama stated that “cybersecurity risks pose some of the most serious economic and national security challenges of the 21st century,” and this sentiment has been echoed by leaders in countries around the world [5]. For this reason, the field of cybersecurity has become the fastest technological growth area in the world.

Bucci, Rosenzweig, and Inserra [6] wrote that cybersecurity has become a very important part of information technology and the internet. The internet is one of the most rapid growth regions of technological groundwork development. Around the world, access to the internet increased by more than two billion people in just the past eleven years. It has changed the way businesses use information technology to share their information and conduct business online. The authors also point out they now use next generation mobile computing and cloud computing for this. Countries also depend on cyberspace for nearly all of their daily management activities, such as movement of troops and financial business. This makes the world we live in today a "wired" world. This means all of our information is in digital form and very vulnerable to exploitation, compromise, and attack. This very fast expansion of the internet has provided a means to connect the whole world digitally and give everyone, from your everyday citizen to criminals and terrorists, a means to reach out and access large amounts of data almost instantly. An individual from anywhere in the world can use cyber capabilities to attack a network thousands of miles away, causing a disruption in business, the destruction of data, or even the shutdown of critical infrastructure. Thus, cybersecurity is a very critical and pressing matter for the U.S. today [6].

According to "A Congressional Guide: Seven Steps to U.S. Security, Prosperity, and Freedom in Cyberspace" [6], the U.S. faces three different types of cyber threats. These threats are, cyber crime, cyber espionage, and cyberwarfare. Cyber crime affects many people in the form of cyber vandalism, identity theft, and phishing. Cyber espionage is frequently state-sponsored and goes after sizeable targets of great importance, such as proprietary ideas and military designs. Cyberwarfare attacks are acts of a nation-state to infiltrate another nation's computers and information networks in an effort to cause damage and disruption. These nations target the critical systems/ infrastructures that are connected to and rely on those computers and networks. Taking down those critical systems such as transportation, communications, or power, would severely impair a response by the U.S. to a physical attack by an adversary [6]. The challenge of guarding against cyber threats is not only owed to their dispersed and diverse nature, but also due to the fact that so much depends on how establishments act in

response to cyber threats when the situation is critical and the time has come for action [5]. It is important to distinguish between the notion of a vulnerability and that of a threat when discussing cyber incidents or potential incidents. For example, a door left unlocked is considered a vulnerability (not a threat). The threat associated with the unlocked door, would be the terrorist desiring access to the building. Note that each vulnerability can be considered to "invite," and facilitate threats that are designed to exploit the vulnerability: the door that is left unlocked might lead to terrorists slipping in a bomb, rivals walking out with the company's trade secrets, robbers stealing valuable goods, or local thugs destroying property. The essential characteristics of threats are the actor and the consequence [5]. In order to measure and evaluate the characteristics of a cyber threat, cybersecurity incidents must be thoroughly examined.

B. CYBER INCIDENTS

The United States describes cybersecurity incidents and cyber events in a different manner than many other countries around the world. A cybersecurity incident is defined as, "actions taken through the use of computer networks that result in an actual or potentially adverse effect on an information system and/or the information residing therein" [7]. A cyber event is defined as, "a cybersecurity change that may have an impact on organizational operations (including mission, capabilities, or reputation)" [8]. When an event is caused by a malicious act, or else is suggestive of a malicious act, then that event is an indicator of an incident. The term "incident," by definition, is indicative of a hostile action or consequence. The importance in understanding the difference between an incident (direct indication of maliciousness) and event (non-direct indication of maliciousness) can be seen in the *DOD Cyber Incident Handling Program*—the Chairman Joint Chiefs of Staff Manual 6510.01B [9]. Table 1 lists ten event and/or incident categories from this manual, which is detailed in Appendix A to Enclosure B [10].

Table 1. CJCS Manual 6510.01B Event and/or Incident Categories. Source: [10].

Precedence	Category	Description
0	0	Training and Exercises
1	1	Root Level Intrusion (Incident)
2	2	User Level Intrusion (Incident)
3	4	Denial of Service (Incident)
4	7	Malicious Logic (Incident)
5	3	Unsuccessful Activity Attempt (Event)
6	5	Non-compliance Activity (Event)
7	6	Reconnaissance (Event)
8	8	Investigating (Event)
9	9	Explained Anomaly (Event)

The four categories described as incidents are the ones of interest for the focus of this thesis, and are explained in detail below. These categories are referenced straight from the DOD Cyber Incident Handling Program—the Chairman Joint Chiefs of Staff Manual 6510.01B [10].

1. Category 1: Root Level Intrusion (Incident)

Unauthorized privileged access to an information system (IS). Privileged access, often referred to as administrative or root access, provides unrestricted access to the IS. This category includes unauthorized access to information or unauthorized access to account credentials that could be used to perform administrative functions (e.g., domain administrator). If the IS is compromised with malicious code that provides remote interactive control, it will be reported in this category.

2. Category 2: User Level Intrusion (Incident)

Unauthorized non-privileged access to an IS. Non-privileged access, often referred to as user-level access, provides restricted access to the IS based on the privileges granted to the user. This includes unauthorized access to information or unauthorized access to account credentials that could be used to perform user functions such as accessing Web applications, Web portals, or other similar information resources. If the IS is compromised with malicious code that provides remote interactive control, it will be reported in this category.

3. Category 4: Denial of Service (Incident)

Activity that denies, degrades, or disrupts normal functionality of an IS or DOD information network.

4. Category 7: Malicious Logic (Incident)

Installation of software designed and/or deployed by adversaries with malicious intentions for the purpose of gaining access to resources or information without the consent or knowledge of the user. This only includes malicious code that does not provide remote interactive control of the compromised IS. Malicious code that has allowed interactive access should be categorized as category 1 or category 2 incidents, not category 7. Interactive active access may include automated tools that establish an open channel of communications to and/or from an IS [10].

C. CYBER INCIDENT RESPONSE

According to *Incident Response and Computer Forensics* [11], not all incidents are preventable, but to lower their likelihood, risk assessments should be conducted, followed by the implementation of appropriate security controls. A computer security incident has the following characteristics: committed to cause harm, initiated or executed by a person, and involves a computing resource. This book observed that looking at the first two characteristics, we can see they are consistent with numerous kinds of everyday non-technical incidents, like assault, arson or theft. Without the commitment to cause harm, it is difficult to label an event as an incident. Also, not all incidents are going to cause immediately detectable harm, even though the intentions to cause harm are indisputable. The second characteristic—executed by a person—would preclude events such as chance system failures or power outages, unless these events were caused by a person. The last characteristic—includes a computing resource—, is what marks the occurrence as a *computer* security incident. A computing resource could be any number of different automated/digitized information technologies, such as phones, cameras, printers, TVs, tablets, and countless others. The book also suggested that nowadays, these devices are everywhere one looks and it is easy to forget the voluminous amount of information they store, what they are connected (or capable of connecting) to, and what they have the ability to control [11].

The *Computer Security Incident Handling Guide* [12] notes the most demanding and difficult element of the incident response procedure is the process of trying to determine if an incident truly has occurred, and if it is proven to be a true incident, determine its category, scope, and root cause. This guide also states what makes this process so difficult is a blending of three elements:

1. The detection of incidents occurs through a variety of means, with differing levels of detail and reliability. Detection can be automated or occur through manual means. Automated detection is comprised of such devices as host and network based intrusion detection and prevention systems (IDSPs), log analyzers, and antivirus software. Manual means of detection would include a user reporting a problem, or an analyst that is manually reviewing system log data and discovering some anomaly.

2. The volume of possible indicators of incidents is normally tremendous for a company. It is not unusual for these companies to have thousands or even millions of intrusion detection sensor alarms each day.

3. The thorough and accurate evaluation of incident-related information requires a deep, expert-level, technical knowledge of all involved technology. [12]

According to the *Computer Security Incident Handling Guide* [12], indicators and precursors are the two categories of information that can lead to detection of an incident. Indicators serve as clues that an incident might have happened or is happening *now*. Precursors are clues an incident might happen in the *future*. Examples of precursors are threats from an individual or group stating their intent to attack a target, or log entries from a Web server indicating the use of a vulnerability scanner. Precursors are relatively (to indicators) rare, but indicators occur quite regularly. Examples of indicators would be alerts from the antivirus software detecting a host is infected with malware, an email administrator detecting a very high volume of bounced emails with questionable content, or numerous unsuccessful login attempts from an unrecognized remote system.

There are numerous sources for the identification of indicators and precursors. The most common sources are people, logs, publicly available information (e.g., news

report indicating that a terrorist group has announced a specific new target), and computer security software notifications. These sources are described in Table 2.

Table 2. Common Sources of Indicators and Precursors for each Category. Source: [12].

Source	Description
Alerts	
IDPSs	IDPS products identify suspicious events and record pertinent data regarding them, including the date and time the attack was detected, the type of attack, the source and destination IP addresses, and the username (if applicable and known). Most IDPS products use attack signatures to identify malicious activity; the signatures must be kept up to date so that the newest attacks can be detected. IDPS software often produces false positives—alerts that indicate malicious activity is occurring, when in fact there has been none. Analysts should manually validate IDPS alerts either by closely reviewing the recorded supporting data or by getting related data from other
SIEMs	Security Information and Event Management (SIEM) products are similar to IDPS products, but they generate alerts based on analysis of log data (see below).
Antivirus and antispam software	Antivirus software detects various forms of malware, generates alerts, and prevents the malware from infecting hosts. Current antivirus products are effective at stopping many instances of malware if their signatures are kept up to date. Antispam software is used to detect spam and prevent it from reaching users' mailboxes. Spam may contain malware, phishing attacks, and other malicious content, so alerts from antispam software may indicate attack attempts.
File integrity checking software	File integrity checking software can detect changes made to important files during incidents. It uses a hashing algorithm to obtain a cryptographic checksum for each designated file. If the file is altered and the checksum is recalculated, an extremely high probability exists that the new checksum will not match the old checksum. By regularly recalculating checksums and comparing them with previous values, changes to files can be detected.

(continued on next page)

Source	Description
Third-party monitoring services	Third parties offer a variety of subscription-based and free monitoring services. An example is fraud detection services that will notify an organization if its IP addresses, domain names, etc., are associated with current incident activity involving other organizations. There are also free real-time blacklists with similar information. Another example of a third-party monitoring service is a CSIRC notification list; these lists are often available only to other incident response teams.
Logs	
Operating system, service and application logs	Logs from operating systems, services, and applications (particularly audit-related data) are frequently of great value when an incident occurs, such as recording which accounts were accessed and what actions were performed. Organizations should require a baseline level of logging on all systems and a higher baseline level on critical systems. Logs can be used for analysis by correlating event information. Depending on the event information, an alert can be generated to indicate an incident. Section 3.2.4 discusses the value of centralized logging.
Network device logs	Logs from network devices such as firewalls and routers are not typically a primary source of precursors or indicators. Although these devices are usually configured to log blocked connection attempts, they provide little information about the nature of the activity. Still, they can be valuable in identifying network trends and in correlating events detected by other devices.
Network flows	A network flow is a particular communication session occurring between hosts. Routers and other networking devices can provide network flow information, which can be used to find anomalous network activity caused by malware, data exfiltration, and other malicious acts. There are many standards for flow data formats, including NetFlow, sFlow, and IPFIX.

(continued on next page)

Publicly Available Information	
Information on new vulnerabilities and exploits	Keeping up with new vulnerabilities and exploits can prevent some incidents from occurring and assist in detecting and analyzing new attacks. The National Vulnerability Database (NVD) contains information on vulnerabilities. Organizations such as US-CERT and CERT®/CC periodically provide threat update information through briefings, Web postings, and mailing lists.
People	
People from within the organization	Users, system administrators, network administrators, security staff, and others from within the organization may report signs of incidents. It is important to validate all such reports. One approach is to ask people who provide such information how confident they are of the accuracy of the information. Recording this estimate along with the information provided can help considerably during incident analysis, particularly when conflicting data is discovered.
People from other organizations	Reports of incidents that originate externally should be taken seriously. For example, the organization might be contacted by a party claiming a system at the organization is attacking its systems. External users may also report other indicators, such as a defaced Web page or an unavailable service. Other incident response teams also may report incidents. It is important to have mechanisms in place for external parties to report indicators and for trained staff to monitor those mechanisms carefully; this may be as simple as setting up a phone number and email address, configured to forward messages to the help desk.

The *Computer Security Incident Handling Guide* noted these sources of precursors and indicators are not guaranteed to be reliable/accurate, so it is often necessary to corroborate each one with additional precursors and indicators. The large volume of indicators received by an organization on a daily basis makes this task very daunting. Just by determining that an indicator is accurate, does not necessarily indicate an actual incident occurred. Such an instance is referred to as a false positive. In incident handling/response, timely detection of all *true* positives (i.e., actual incidents) is a challenging task. This is what makes the automated correlation of individual precursors and indicators so important and necessary for improved incident detection [12].

The *Computer Security Incident Handling Guide* recognized incident response is an organized and well thought out methodology to go from incident discovery to resolution. Typically, an incident response effort is conducted by an investigating team that makes informed determinations regarding the technical details of what occurred, then takes appropriate response actions to remedy any damage done and—ideally—improve the targeted system's defenses against the same or similar attack in the future. There are several phases to incident response. Though the exact naming of these phases varies by organization, the phases are typically named (and ordered) as follows: preparation, detection and analysis, containment, eradication and recovery, and post-incident activity [12]. This document stated the first phase comprises the establishment and training of an incident response team, and obtaining the essential resources and tools. This preparation phase is where an organization tries to curb the number of incidents by deciding on a set of controls based on risk assessment results, and then implementing them *before* an incident is actually encountered. Even after these controls are in place, there will still be some residual risk. This is what makes detection of security breaches essential for alerting an organization when an incident has occurred. Depending on the gravity of the incident, the organization can diminish an incident's effect by containing it and—as quickly as possible—recovering any/all affected systems and information.

The next phase of the incident life cycle is the detection and analysis phase. This is one of the most challenging elements of the incident response process because here one must accurately detect and assess possible incidents, determining whether what has occurred is truly an incident, or simply an accident or benign anomaly. Also, the methods/mechanisms that are used to introduce malicious artifacts/activities into the targeted system are numerous. These are often referred to as attack vectors. Attack vectors allow malicious actors to exploit vulnerabilities within the network. They can be any path or means an attacker would use to deliver a malicious payload to the system. Web, email, and attrition are examples of attack vectors. The behaviors associated with these attack vectors can be utilized in creating IOC rules. Behaviors associated with both Web and email are the way a system acts/responds when infected by malware. These behaviors could be registry keys that are added to the registry settings of the computer to

allow for persistence by the malicious code which would be indicative of malware. The malicious registry keys would be the IOCs used in creating the rule. Behaviors associated with a brute force attack would be indicative of an attrition attack vector. Usually a rootkit is installed on a system by an attacker attempting a brute force password attack. Behaviors associated with a rootkit that could be used as IOCs are MD 5 hash, process handles, unique filenames, and digital signature information that is invalid. Viewed separately, these IOCs may not provide a very good indication that a system has been compromised. The information could be combined into a correlation rule that would look for all of the aforementioned IOCs therefore providing a stronger indication that your system was actually compromised. The incident response team should be able to respond to any of these attacks quickly, analyzing and confirming each incident and documenting each step they took along the way. The scope of the incident; such as what systems, applications, or networks are affected; what or who initiated the incident; and how the incident is transpiring should be determined by the team. This information should give the team enough data to prioritize appropriate follow-on actions; e.g., is additional/deeper analysis needed, or is the team ready to proceed to containment and/or eradication actions [12].

The *Computer Security Incident Handling Guide* wrote the next phase of the incident life cycle is containment, eradication, and recovery. The first thing the incident response team must do is contain the incident so as to prevent further damage. Containment gives the team time to establish a custom-made remediation plan. During this process, they must decide whether to remove the infected device from the network, shut the system down, or disable certain functions on the device. An organization's containment strategy will differ based on the type of incident that has occurred. Preset containment strategies and actions should be developed for each main incident type, with well-defined criteria documented to enable more quick and precise decision-making. These criteria are: the resources and time required to carry out the plan, success of the plan, service accessibility, possible theft and damage of resources, need to secure evidence, and how long the solution will last [12]. After containment of the incident is complete, eradication might be required in order to remove certain elements of the

incident. These elements could be anything from deactivating penetrated user accounts, to removing malware. It is imperative that all affected hosts are identified within the organization to ensure full remediation. Once this is accomplished the organization can begin the recovery process. Administrators will return systems to their normal operating conditions and verify they are functioning within normal (pre-incident) standards. This will typically be accomplished via clean backups of the system, replacement of compromised files with new versions, changing passwords, installing patches, or—in more extreme incidents—conducting a bare-metal reconstruction of the system. The recovery process will focus on longstanding security changes the organization can make to ensure they are as protected as they can possibly be against the same or similar attack in the future [12].

The last phase, post-incident activity phase, is important because this is where the organization learns from their mistakes and improves upon their plans, policies, and defense posture. The "lessons learned" data can be used to help the organization determine how many man-hours were spent, and the total cost of the incident. Incident response teams use the lessons learned data to request and justify their need for additional funding. Another important task for the incident response team is to produce a follow-up report that can be referenced in the future to deal with incidents that are similar in nature [12].

D. SECURITY INFORMATION AND EVENT MANAGEMENT

SIEM technology employs a combination of what has historically been two separate categories of IT security management tools: security information management (SIM) and security event management (SEM) [9]. According to *Network and System Security* [13], the SIEMs underlying principle is the collection of pertinent data about an organization's security posture gathered from near- or real-time observation of events and actions occurring on an organization's essential systems. The clients, servers, and other security devices are those essential systems where the "raw" indicators and precursors of an incident comes from. The SIEM is a collector *of* all that "raw" data arriving from those essential systems. As such, the SIEM stands apart. It gathers all those indicators

and precursors and – owing to some well-written rules – will aggregate and correlate this data to add additional value to it. This value added is the core of what SIEM solutions with well-defined IOC rules bring to the cyber defense table.

Organizations use log management and/or security information and event management (SIEM) tools to appropriately monitor for technical events which could lead to an investigation or incident. It allows for faster recovery from incidents by collecting, storing, and analyzing log and other security-relevant information. These tools are used by security managers and analysts to automate the collection and analysis (to varying degrees) of very large volumes of system-generated event data. One may imagine these automated tools as serving as intelligent pre-filters that allow human responders to more quickly identify and focus on only the most noteworthy events [13]. In addition to incident detection, these tools are also helpful in generating reports which are used for purposes of policy compliance. SIEM solutions typically arrive as either standalone appliances, remotely managed services, or loadable—often open source—software. A list of SIEM capabilities taken from "*Network and System Security"* are as follows:

- Alerting: Automated evaluation of associated events and creation of alerts, used to warn recipients of urgent issues.
- Compliance: SIEM functions are used to automate the collecting of compliance data, generating reports that conform to current security, auditing procedures, and governance.
- Correlation: Searches for common characteristics and connects events together into significant packets. This technology delivers the capacity to execute an assortment of correlation techniques to incorporate diverse sources, for the sake of turning data into beneficial information.
- Dashboards: SIEM/Log Management (LM) tools utilize event data and produce informational charts to help visualize patterns or identify actions which do not form a normal pattern.
- Data Aggregation: SIEM/LM solutions combine data from numerous sources, including databases, servers, security, network, and applications, delivering the ability to consolidate censored data in order to prevent missing critical events.

- Retention: SIEM/SIM solutions use lifelong storage of historical data to enable the correlation of data over a certain period of time and offer the retention of this information needed for compliance obligations [13].

At the most fundamental level, a SIEM system will likely involve some combination of signature-, rule-, and statistics-based correlation mechanisms so as to determine incident-related semantic correlations among multiple logged events. As discussed previously, system "events" are captured (documented) by log messages collected from the various devices that have been configured to provide such. Example log sources are intrusion detection systems, firewalls, routers, hosts, etc. [14]. An example logged event is a user "login" event which would include such useful information as username, hostname, and a timestamp. All events collected by the SIEM system go through a sequence of "rules" called the "rule system." This rule system, or "ruleset," produces "alerts" dependent upon the attributes of the events being processed [14]. The alerts produced signify that an important event or sequence of events has occurred, and which requires attention. Alerts are normally reviewed by security analysts, and will typically be prominently displayed on a SIEM dashboard for quick and easy notice. These alerts, along with the logged events which elicited them, are then saved in the database for tracking and reporting reasons [14].

Incident Response and Computer Forensics believed this ruleset could be considered the heart of a SIEM, as it adds value to the relatively narrow information derivable from the many individual/isolated logs collected. As such, these rulesets, including their format and definitions, are important to the improvement of incident response. SIEM rules are often articulated in the language of indicators of compromise (IOCs). IOCs seek to define the forensic artifacts that constitute various intrusions. These specific artifacts can serve as reliable indicators of incident-related activity. They can take the form of domain names, email addresses, IP addresses of command and control servers, file hashes, file size, name, etc. The intent of creating IOCs for a specific piece of malware, or attack behavior, is to define their related artifacts in a manner that allows an automated system (e.g., a SIEM) to detect the defined artifacts much more quickly than is possible when attempting to do so manually. The creation of IOCs is the method of recording features and artifacts of a security incident or attack in an organized manner.

An important concern in deciding how to represent IOCs, is the capability to use this structure in a standardized, and thus distributable, way within any given organization [11]. Network-based IOCs are typically structured as Snort rules. Snort, as defined by snort.org, is an open source network intrusion prevention and detection system (IDS/IPS) developed by Sourcefire. It is the most commonly deployed IDS/IPS technology globally, and brings together all the benefits of protocol, anomaly-based, and signature inspection into one tool. For host-based IOCs, there is no single, uniformly agreed upon and/or utilized, format. Three nascent standards are, however, currently available for further consideration and development. These are: Mitre's CybOX, Mandiant's OpenIOC, and YARA.

Chapter II will focus on how to detect these cyber incidents and other reportable cyber events through: 1) introducing the DOD Cyber Incident Handling Program, 2) enumerating the types of logged system events of most interest to attack sensing and warning (AS&W), and 3) discussing the merits of event correlation to improve attack sensing and warning.

THIS PAGE INTENTIONALLY LEFT BLANK

II. DETECTING CYBER INCIDENTS AND OTHER REPORTABLE CYBER EVENTS

A. THE DOD CYBER INCIDENT HANDLING PROGRAM (CJCSM 6510.01B)

The *Chairman of the Joint Chiefs of Staff Manual 6510.01B* is the manual that governs the Department of Defense (DOD) Cyber Incident Handling Program [10]. The purpose of this manual is to describe this program in detail and specify its main processes, employment requirements, and other related U.S. government interactions. This program promotes a unified ability to constantly improve the DOD's capability to quickly recognize and respond to cyber incidents that negatively affect DOD information systems (ISs) [10]. The scope of this document is based on the fact that the DOD is comprised of numerous military commands, organizations, agencies, and functions that must direct, coordinate, and answer to technology attacks, threats, and incidents. Without appropriate controls to manage, protect, and detect their effects, these attacks, threats, and incidents could negatively affect DOD information systems and networks. The document provides overarching direction that promotes a shared and in-depth understanding of how the DOD's local, regional, and global organizations synchronize attempts to positively affect response activities [10]. Direction contained within covers the high-level procedures associated with the Monitor, Analyze, Detect, Protect, and Respond stages of the Computer Network Defense (CND) life cycle. It provides the rudimentary framework necessary to structure a DOD-wide cyber incident-handling program.

The activities described in this manual are intended to be hierarchically distributed across three levels—referred to as tiers—of organization: Tiers I, II, and III. Global organizations (Tier I) offer DOD-wide CND operational support or direction. USCYBERCOM is a Tier I entity. Regional (Tier II) organizations offer DOD component-wide operational support or direction and answers to Tier I. Combatant Command/Service/Agency/Field Activity Computer Network Defense Service Providers (CNDSPs) are Tier II entities. Local organizations (Tier III) offer local operational

support or direction, and answer to direction from their appropriate Tier II entity. Bases, stations, and posts are examples of Tier III entities [10].

The basic method for DOD cyber incident management can be arranged into the following phases [10]:

(1) Detection of events

(2) Preliminary analysis and identification of incidents

(3) Preliminary response actions

(4) Incident analysis

(5) Response and recovery

(6) Post-incident analysis

Phases 1 and 2 are of the greatest importance for this thesis as they focus on detection and preliminary investigation. These phases are where the SIEM can prove so valuable. Through the centralization of an organizations security events, via the logging capabilities of the SIEM, attacks can be detected that may not have otherwise been discovered. This is accomplished via the correlation rules within the SIEM. These rules will allow the SIEM to search for multiple individual indicators within a single rule. Basically, they help connect the dots on logically-related (to incident activity/behavior), yet otherwise unrelated data. Due to the vast number of sources that generate security log entries, the incident "first responders" need a single "console-type view" that would allow them to view, analyze, and report on the content found within these log entries. The automation the SIEM provides is also invaluable not only due to the large volume of data that can be ingested and analyzed, but also due to how rapidly this data can be analyzed.

The SIEM is also crucial in the preliminary investigation of an incident. It can provide a "first responder" with the details necessary to make an intelligent, well informed decision regarding the appropriate response action for the threat. If the rules are well-written and configured properly with strong IOCs, then false positives and false negatives should be few, and only true positives will be presented to the SIEM operator. This will help ensure faster response times to the most severe incidents, possibly early enough in the attack phase so as to prevent (or limit) any actual damage to the system.

The preliminary investigation may also reveal new IOCs that are worth considering to be added to the SIEM's ruleset.

The purpose of minimum reporting timelines, in Table 3, are for organizations to report incidents as quickly as possible so that appropriate actions can be taken to stop or diminish harm [10]. This is made possible because the SIEM allows streamlined reporting via its customizable reporting and centralized logging capabilities. With good IOCs, and therefore stronger rules, the "first responder" will be able to collect many, if not all, of the pertinent details of the incident, thus producing more accurate and timely information for the initial report.

Table 3. Reporting Timelines for Cyber Incidents. Source: [10].

Category	**Impact**	**Initial Notification to Next Tier**	**Initial Report to Next Tier**	**Initial Submission to JIMS**	**Minimum Reporting**
1 Root Level Intrusion (Incident)	High	Within 15 minutes	Within 4 hours	Within 6 hours	Tier I
	Moderate	Within 2 hours	Within 8 hours	Within 12 hours	Tier I
	Low	Within 4 hours	Within 12 hours	Within 24 hours	Tier I
2 User Level Intrusion (Incident)	High	Within 15 minutes	Within 4 hours	Within 6 hours	Tier I
	Moderate	Within 2 hours	Within 8 hours	Within 12 hours	Tier I
	Low	Within 4 hours	Within 12 hours	Within 24 hours	Tier I

(continued on next page)

Category	Impact	Initial Notification to Next Tier	Initial Report to Next Tier	Initial Submission to JIMS	Minimum Reporting
4 Denial of Service (Incident)	High	Within 15 minutes	Within 4 hours	Within 6 hours	Tier I
	Moderate	Within 15 minutes	Within 4 hours	Within 6 hours of discovery	Tier I
	Low	As directed by CC/S/A/ FA Guidance	As directed by CC/S/A/ FA Guidance	As directed by CC/S/A/ FA Guidance	Tier I
7 Malicious Logic (Incident)	High	Within 15 minutes	Within 4 hours	Within 6 hours	Tier I
	Moderate	Within 2 hours	Within 8 hours	Within 12 hours	Tier II
	Low	As directed by CC/S/A/ FA Guidance	As directed by CC/S/A/ FA Guidance	As directed by CC/S/A/ FA Guidance	Tier II

The cyber incident reporting format in Table 4 provides a format for reporting preliminary incidents by secure telephone, fax, or by some other electronic channels. Initial reports may not be complete. The reporting organization must balance the requirement of judicious reporting against comprehensive reports [10]. Timely reporting is critical, and complete information must be given as new details occur. The report format consists of the cyber incident tracking information, reporting information, categorization information, technical details, sites involved, impact assessment, and additional reporting or coordination. The section most relevant to this thesis is the technical details section as seen in Table 4.

Table 4. Technical Details Section of the Cyber Incident Report Format. Source: [10].

Field	Description
Technical Details	
Event/Incident Description	Provide a narrative description of the incident with technical details. Include DTGs of significant events (start, stop, or change of activity). State the use of the targeted IS and whether the IS is online or offline. Indicate whether the incident is ongoing.
Root Cause(s)	Identify the IS specific cause(s) of the incident. The root cause expands upon the identified delivery vector(s) and IS weaknesses by precisely identifying the sets of conditions allowing the incident to occur. Indicate whether the DAA or CIO had accepted a risk that led to the incident.
Source IP and Port	Provide source IP with resolution data identifying owner and country of source IP machine. Note: The source IP could be a DOD IP. If the intruder is known, provide all identifying information to include the intruder's objective, if known. Source IP is not necessarily indicative of true origin. Footnote the source of resolution/attribution data (i.e., ARIN.org). Insert "Not Applicable" for incidents that do not involve source IP or port.
Intruder(s) (if known)	Identify the intruder or group responsible for the incident, if known.
Origin (Country)	Identify the source IP's country of origin.
Target IP(s) and Port	Provide target IP with resolution identifying responsible command and physical location of target IP machine (e.g., B/ C/P/S, etc.). Footnote the source of resolution/attribution data (i.e., DDD NIC, nslookup, and whois). If machine is behind a network address translation enabled (NAT'ed) router or firewall then also provide the wide area network (WAN) routable address (i.e., the internet/SIPRNET routable IP address).
Technique, Tool, or Exploit Used	Identify the technique, tool, or exploit used.
Operating System (OS) and OS Version	Record the OS and version number of the OS where the incident occurred.
Use of Target (e.g., Web Server, File Server, Host)	What the intruder/attacker used the target IS for, after it was exploited, if applicable.
Method of Detection	Identify how the intrusion was detected (e.g., external notification, log files, network monitoring, IDS, user).

For the cyber incident analysis framework, the type of analysis performed will vary based on what type of incident is being examined [10]. Incident analysis is a logical series of steps that an incident response team must go through in order to determine what occurred during an incident. The reason behind this evaluation is so the incident response team can fully understand the technical details; including what is often considered the most important aspect: root cause(s) [10]. Good IOCs can aid in providing the technical details required in an incident report. They can help determine the root cause of an incident by identifying what the attack vector was, based on the identified behavioral attributes. These behavioral attributes can be changes made to the registry keys, services started, a large increase in DNS requests from a specific host, or a process exits and then a privileged event occurs. Source IP and port as well as target IP and port are good IOCs to be included in a correlation rule as well.

The evaluation should also shed light on both actualized and potential effects of the incident on normal system operation. This knowledge will aid in establishing what added information to collect, how to synchronize information distribution with others, and what remediating courses of action (COAs) are most appropriate for response activities. If the possibility exists the incident will require legal action to be taken, the appropriate authorities must be contacted to ensure the proper legal actions are followed during the incident examination [10]. The type of incident, along with the technical details and operational impacts, will determine the extent to which an incident must be examined. It will also depend on what information and resources the incident response team has at their disposal [10].

From the CJCSM 6510.01B, “A delivery vector is defined as the primary path or method used by the adversary to cause the incident or event to occur” [10]. This information is used in a portion of the incident report to help determine if there is a trend in how often the different vectors occur. By understanding vector methods, techniques and trends, strategic and tactical plans can be drafted or modified in order to advance the defensive stance of DOD information networks. Although the technical specifics of any given delivery vector are often multifaceted and evolving, generalizing their

methodology allows them to be classified into just a few categories. The main groupings and sub-groupings of delivery vectors are described in Table 5.

Table 5. Delivery Vectors Categories. Source: [10].

Delivery Vector Category Number		Description
1	**Sub-category**	**Reconnaissance:** Information was accessible and used to characterize ISs, applications, information networks, and users that may be useful in formulating an attack.
	A	Information Gathering and Data Mining: Activity that seeks to gather information from publicly available sources.
	B	Network Scan: Activity that targets multiple IP addresses. This is referred to as a horizontal scan.
	C	System Scan: Activity that targets a single IP address across a range of ports. This is referred to as a vertical scan.
2	**Sub-category**	**Authorized User:** A user with authorized access took specific actions that resulted in jeopardizing ISs or data.
	A	Purposeful: An authorized user knowingly took specific actions that jeopardized ISs or data.
	B	Accidental: An authorized user took actions that had consequences over and above the intentions and jeopardized ISs or data.
3	**Sub-category**	**Social Engineering:** Human interaction (social skills) or deception used to gain access to resources or information.
	A	Email: Email is the primary vehicle used to deliver a malicious payload or gain access to resources or information.
	B	website: A website is the primary vehicle used to deliver a malicious payload or gain access to resources or information.
	C	Other: A user was deceived or manipulated in a way that is not covered by the other types of social engineering.
4	**Sub-category**	**Configuration Management:** Compromise resulting from the inadequate or improper configuration of an IS.
	A	Network: An IS that provides network-based services was improperly or inadequately configured.
	B	OS: An OS was improperly or inadequately configured.

(continued on next page)

Delivery Vector / Category Number		Description
	C	Application: An application was improperly or inadequately configured.
5	**Sub-category**	**Software Flaw:** A vulnerability in the software that allows for the unauthorized use of or access to an IS in a way that violates the IS's security policy.
	A	Exploited New Vulnerability: This vulnerability was unknown prior to the event or there was no mechanism available to prevent it.
	B	Exploited Known Vulnerability: This vulnerability was known prior to the event and there was a mechanism available to prevent it.
6	**Sub-category**	**Transitive Trust:** Compromise resulting from the implicit or explicit trust relationship between security domains.
	A	Other IS Compromise: Compromise resulting from access previously gained on another IS.
	B	Masquerading: Compromise resulting from the unauthorized use of a valid user's credentials. This may include cryptographic material, account credentials, or other identification information.
7	**Sub-category**	**Resource Exhaustion:** The consumption of IS resources that prevents legitimate users from accessing a resource, service, or information.
	A	Non-Distributed Network Activity: Activity from a single IP address that overwhelms IS or information network resources. This is generally associated with a DoS incident.
	B	Distributed Network Activity: Activity from multiple IP addresses that overwhelms IS or information network resources. This is generally associated with a DoS incident.
8	**Sub-category**	**Physical Access:** The unauthorized physical access to resources.
	A	Mishandled or lost resource: Equipment was stolen, lost, or left accessible to unauthorized parties.
	B	Local access to IS: An unauthorized user was provided local physical access to a DOD information network resource.
	C	Abuse of resources: The physical destruction of an information resource by an unauthorized party.
9	**Sub-category**	Other
	A	New Delivery Vector: The delivery vector is not covered by the listed methods. Description of the delivery vector must be included in the incident comments.
10	**Sub-category**	Unknown.
	A	Unable to Determine: Delivery vector could not be determined with the information available.

The delivery vector categories described above are not all-inclusive. Reasonably, they generally define the main groupings of delivery vectors. Subcategories are used to provide a higher degree of granularity, so as to provide further specificity for any particular identified delivery vector. For example, the delivery vector category "Software Flaw" includes the following subcategories: "Exploited an Existing Vulnerability" or "Exploited a New Vulnerability." This additional specificity is informative to incident responders. If a vulnerability is determined to be one already existing, the most appropriate remediation action would be to patch the (known) vulnerability; while the other (new vulnerability), would require additional investigative effort to obtain sufficient understanding of the vulnerability to facilitate development of a patch, or mitigating security control.

B. EVENTS OF INTEREST TO ATTACK SENSING AND WARNING (AS&W)

"AS&W is defined as the identification, characterization, detection, and correlation of deliberate unauthorized cyber activity with a warning to command and decision authorities in order for a suitable response to be developed" [15]. Achieving an effective AS&W capability entails systematic gathering of intrusion-/attack-associated intelligence [16]. Such a capability benefits from a network of anomaly, intrusion, and misappropriation detection systems, which in turn feeds into a data fusion and evaluation facility that is capable of long-term pattern and trend analysis.

1. Log-Based Data

Because it is becoming more and more challenging to identify malicious cyber activity, it is proving more critical to monitor log data from as many valuable sources as possible [17]. Event logs encompass a huge volume of data; so much so that manually sifting through these logs to uncover indications of malicious activity would be exceedingly time-consuming, if not outright impossible. There are four main kinds of event logs that are beneficial in detecting possible incidents and/or proceeding with their investigation once they have been detected. The four kinds of logs are detailed in Table 6.

Table 6. Main Types of Event Logs. Source: [18].

Type of logs	Examples
Networking logs	• Firewall, Email, Net flow logs, and VPN
Records from logging and cyber security monitoring tools	• Network intrusion detection systems (NIDS) • Malware protection logs • Data loss protection (DLP) • Network intrusion prevention systems (NIPS) • Tools that utilize potential investigation techniques and malware isolation (virtual execution engines or sandboxing)
System logs	• Endpoint logs • System activity logs • Logs from customized and standard applications • Physical security logs • Authentication logs
Technical logs	• Web and SQL server logs • HTTP proxy logs • App flow logs • DNS, DHCP and FTP logs

Evidence of an incident could potentially be captured in numerous logs that all contain different kinds of data, but all of which; nonetheless, provide useful detective or investigative information pertaining to the incident. For example, an application log could provide the username used by an attacker, while a firewall log could provide the destination IP address targeted by the attacker [18].

Log-based data is raw data from numerous network sensor sources. Numerous network-based applications generate events which they write to log files. The source events that populate log files could come from many possible devices and services found on a typical network; e.g., firewalls, Web proxies, and mail server virus scanners. The information provided directly by, or inferable from, these log files, can be correlated by automated systems in order to enhance the speed and reliability at which incidents are detected [19].

Network log-based detection is a method that centers on the examining of audit logs produced by network devices. Audit logs have two main components. One is a collection of audited events, which are considered indicative of "bad" behavior. This can include actions deemed to be unauthorized, obviously malicious, or otherwise suspicious owing to any number of metrics. The second element is an audit trail examination module. These audit trails are derived from a sequential record of actions on a system. The examination module assesses the observed system's audit trail for actions that match activity in the catalog. If a match occurs, the activity is assumed to be intrusive [20].

Logged events are the main records of network and system activity. According to the "SANS Log Management Survey, Shank (2010)," the top reasons for an organization to collect log-based data are based on the data's usefulness and benefits provided. Some of these reasons are listed next in the order of their importance: [10]

- Prevent/detect insider abuse and unauthorized access
- Forensic analysis and correlation
- Ensure regulatory compliance
- Track suspicious behavior
- Monitor user activity

Log management is a critical part of network security and more and more organizations are using it for troubleshooting purposes as well as detecting and analyzing suspicious behavior.

2. Findings from Sebring's and Campbell's Technical Report

This section references Eric Sebring and Shaun Campbell's Applied Cyber Operations CAPSTONE Project Report titled *Enhancing CANES SIEM Performance via Optimized Event Logging* [21]. This project was completed in September of 2016. The intent of their project was to analyze the inputs to the SIEM system and identify the typical system events most likely to be useful as indicators of malicious activity. Their work was conducted for the purpose of assessing the best collection of events to audit for collection and forwarding to an automated SIEM system. Determination of "best," in this context, was based upon various researchers' inputs, and industry-wide logging best

practices. Their effort, in combination with the efforts of other technical reports, including this one, is intended to result in a highly "tuned" SIEM that exhibits few false alerts (i.e., false-positives or false-negatives). This material is important to this thesis, because it identifies the specific (loggable) events deemed most prudent/useful for ingestion by the SIEM. IOCs deployed on a SIEM, are effectively limited by the quality of the raw event data provided to them.

The United States Navy relies upon automated information systems to conduct their everyday operations and complete their missions. CANES is the present shipboard network infrastructure utilized onboard ships. The security suite implemented at the center of the CANES system is the SIEM. One of the key features of the SIEM is the broad platform and application support provided which includes Microsoft Windows. Microsoft Windows comprises most of the network systems onboard U.S. Navy ships. The SIEM collects and analyzes syslog data that is gathered from all organic and non-organic network equipment onboard ships. This allows the SIEM to provide actual intelligence to the overwhelming amount of ostensibly unintelligent data that pours in. Event logs are an integral part of constructing a timeline of what has occurred on a system. For a SIEM to be of most benefit, it is prudent to ensure it is given the most useful log information for consideration. The ideal set of logged events would yield a high signal-to-noise ratio (S/N) with respect to their ability to capture true indicators of malicious activity. This means there will be more "signal" (useful logged events) provided to the SIEM than "noise" (un-useful logged events).

From the findings of Sebring and Campbell, it is suggested that seven current Windows audit policy subcategory settings should be disabled, while thirteen current Windows audit policy settings should be enabled, so as to provide the most effective Windows auditing environment possible. This, they offered, will likely provide the optimal collection of events for the SIEM, and facilitate greater accuracy in identifying true/real events for attack sensing and warning. The actual Windows logging and auditing environment is established via configuration settings, which, with various degrees of granularity, describe the most critical events that should be examined by a SIEM. There are 53 settings which allow one to choose specifically which events to monitor. This

should, then, provide the SIEM with the best collection of logged events for it to run through its correlation engine.

The seven audit policy settings recommended to be disabled were:

1. *Other Account Logon Events* subcategory setting in the Account Logon category
2. Application Group Management and Distro Group Management subcategories under Account Management
3. RPC (Remote Procedure Call) Events under Detailed Tracking category
4. *Network Policy Server* for both success and failure under the Logon/ Logoff category
5. *Other Logon/Logoff Events* under the Logon/Logoff category
6. *Sensitive Privilege Use* subcategory under Privilege Use category
7. *IPsec Driver* subcategory under the System category

The audit policy settings recommended to be enabled were:

1. *Kerberos Authentication Service and Kerberos Service Ticket Operations* subcategory for both success and failure under the Account Logon category
2. Increase the auditing level of the *Process Creation and Process Termination* subcategories to audit both success and failure events under the Detailed Tracking category
3. *Special Logon* should be enabled for success under the Logon/Logoff category
4. *Any event associated with administrator privileges* should be audited for both success and failure under the Logon/Logoff category
5. Add failure to the setting mentioned above in number three
6. *File System* subcategory for both success and failure under the Object Access category
7. *Registry* subcategory for both success and failure events under the Object Access category
8. *Kernel Object* subcategory under the Object Access category

9. *Authorization Policy Change* for success and failure under the Policy Change category

10. *Other Policy Change Events* set to enabled for failure events only under the Policy Change category [21]

These changes to the auditing policy will allow the SIEM to be better able to detect true positives for indicating malicious activity. The SIEM is a crucial tool in facilitating quicker detection and response of malicious activity by systems analysts.

3. Other Sources of Incident Artifacts

There are many security mechanisms that can be utilized for attack sensing and warning, besides the previously discussed log-based data. Intrusion detection systems (IDS), antivirus, firewalls/routers, and vulnerability scanners are but a few examples of such other sources [22].

According to "Survey of Event Correlation Techniques for Attack Detection in Early Warning Systems," IDSs produce security reporting events which are comprised of information concerning both realized, or potentially realized, attacks. These reports vary in the detail they may provide, owing to the accuracy of the underlying intrusion signatures configured into the IDSs. Events produced via these detection systems will have alternating levels of importance, contingent upon the assessment method, and kind of input information and origin. This document also states methods for detection can be separated into two classes: anomaly-based and signature-based. Anomaly-based systems are only able to determine to a certain/limited likelihood, if the malicious event(s)/activity detected is a true-positive indicator of an incident. The results achieved through the use of signature-based systems largely depend upon the quality of the signatures used. Because of this variation in confidence, events are normally prioritized by the combination of an alarm and confidence level. The alarm level defines the gravity of the detected event, and the confidence level indicates how reliable the signatures are [19].

The remainder of this section comes from Spadaro, "Event Correlation for Detecting, Advanced Multi-Stage and Cyber-Attacks." Antivirus software is used on individual computer system for the purposes of mitigating the threat of malware, which

includes Trojans, viruses, keystroke loggers, backdoors, worms, and blended threats, and all other manner of malicious logic artifices. This technology searches for traces of malware in critical components, file systems, and applications, via both signature-based and heuristic-based detection. If a file containing malware is discovered, the antivirus software will attempt to isolate or clean it. Typically, antivirus software and IDSs will complement each other [22].

Firewalls typically depend on basic protocol information, such as destination and source IP addresses, and port numbers, for filtering network traffic. They are intended for blocking unauthorized access efforts and can be reconfigured by certain IDSs to block a specifically identified threat. Certain router models can also monitor network traffic performance and collect data containing statistics and header information for a set of packets with like features; this can, in turn, be used to discover unusual flows [22]. Such unusual flows are often indicators of backdoors, worms, and distributed denial of service (DDoS) activity.

Vulnerability scanners are used to identify vulnerabilities in networks, applications for IT Security Assessment, and computer systems. Malicious actors may use them to gain information about vulnerabilities they can exploit in order to gain unauthorized entrance into a system or network [22]. This is why it is so crucial to utilize these scanners for defense purposes.

File integrity checking software detects alterations made to critical files. Cryptographic checksums can be obtained, via a hashing algorithm, for every file deemed a likely target of attack. Assuming the defender has done this, and thus has known good checksum values, then any alterations would be indicative of potential malicious activity. Through the use of continual recalculation of checksums and their comparison against previous—known good—values, unauthorized alterations to files can be identified [22].

C. EVENT CORRELATION AS ATTACK SENSING AND WARNING (AS&W)

According to the “Survey of Event Correlation Techniques for Attack Detection in Early Warning Systems,” event correlation is the process of integrating the information

inherent in each individual event, across multiple, related events. This integration is technically achieved via identifying Boolean relationships among the events; as those relationships pertain to particular (or typical) incident activity. A simple example is seen in the Boolean operator "AND" being applied to the two individual events of: 1) System firewall was disabled, and 2) Only user "Ben" was logged onto that system at the time that the firewall was disabled. Combined, these two individual events are thus correlated, with—in this example—a logical inference which suggests that further investigation/ follow-up is warranted. Meta data (such as administrative, time, network topology, or location information) may be used to increase the quality of a single or combined event [19]. The survey suggested correlative event evaluation methods can be generally separated into the same two classes as are most detective type technologies; anomaly-based and signature-based.

Tobias states anomaly-based techniques are utilized in order to detect unusual system behavior. This can be accomplished via two different methods. A specification-based method or a data-mining-based method. Data-mining techniques were developed for modeling extraction from huge databases. The first step in this process is to train the algorithm for a system where only "normal" actions take place. During this process, a model that defines the normal environment of the system is created. In the second step of the process, the real system is observed and continuously compared to the generated model. If the difference between the two systems is larger than some pre-defined threshold, then an event is generated that reports the abnormal behavior. There are many different algorithms applicable for anomaly analysis from the field of statistics, databases, learning-based systems, and pattern recognition. Specification-based solutions constantly analyze the behavior of the system and compare it to a known parameter range. If this range is violated, an event is produced which points to the potential attack.

Elsewhere in the document, Tobias notes signature-based techniques are based on pre-defined signatures and events as input data. The signature denotes a kind of filter, which is applied to all incoming events. If the signature matches the incoming information, a certain pre-determined action is implemented, like the notification of the detection to higher layers in the attack detection system. Occasionally, the filter/signature

combination, along with the executed action command(s), is called a rule. Normally, signature-based systems are inclined to generate fewer false positives as compared to anomaly-based systems. This is due to the fact that signatures attempt to identify a certain kind of data inside the input flow, such as a specific kind of exploit in a payload-based IDS. However, the quality of the detection is dependent upon the signatures that were created. Typically, IDSs like Snort offer the ability to create tailored signatures, which can result in a higher degree of true-positive detections and fewer false -positives and false-negatives. Specific attacks can be identified by adding new rules that are specially tailored for their detection. This allows the correlation algorithms to be adapted to changing requirements.

The remainder of this section comes from Spadaro, “Event Correlation for Detecting, Advanced Multi-Stage and Cyber-Attacks.” Spadaro states event correlation can be viewed as operating in a layered security architecture with correlation methods applied across all layers. These layers are: event layer, report layer, and data layer. In the raw data layer, sensor data is gathered and processed, and then directed to the event layer, at which point “correlators” and IDSs prioritize, classify, and dispose of non-relevant data. Next, acquired data is directed to the report layer, to be post-processed. Correlation methods on the raw data layer primarily focus on combining the large volume of data generated by each sensor, removing data features, and detecting simple incidents. In the event layer, lower layer events, where traffic data analysis takes place, are processed in order to combine alerts into meta-alerts in order to gather all possible information for identifying the event that caused the incident. Meta-alerts are usually generated by combining alerts with like qualities caused by different sensors, for the purpose of acquiring high quality information results, reduce redundancy, and decrease the volume of redundant (i.e., related to same behavior that is the underlying cause of an alert) alerts. Examples of this would be alerts that were triggered by the same event, alerts denoting the exact same vulnerability, and alerts connected on a chronological basis. Assessing and evaluating likenesses of traits in alerts is the central focus of probabilistic reasoning approaches. Spadaro also states to identify a security incident, causal associations must be made between events.

With regard to causality, the correlation techniques can be classified into four categories. Statistical-based, similarity-based, multi-stage-based, and scenario-based. Statistical-based approach is where connections among alerts during a certain time frame are analyzed statistically. Similarity-based method is where events are correlated based on a similarity within data features. Multi-stage-based techniques entail the correlation of events based on well-known preconditions and the results of multistage attacks. They are centered on the belief that suspicious events are typically connected to different phases of a multi-stage attack. The Scenario-based method is based on recognized attack scenarios. This method requires expert technical knowledge to define these attack scenarios in advance. According to Spadaro, the report layer is where the final analysis permits the product of the lower layers to be conceptualized, laid out, and post-processed. This is where active countermeasures to attacks/events and event verification take place. Also, manual assessment of statistical data can take place in this layer. Examples of this type of data are average packet sizes and data transfer rates. Event correlation as attack sensing and warning allows a more complete picture to be achieved. Correlation of data is of vital importance in every layer of early warning and attack detection systems. It helps improve the value of detected events and determine if they are linked to real incidents [22]. Individually obscure security events can be correlated via numerous logs, and in the process, produce the advanced level of vision required for precise and prompt intrusion analysis.

Chapter III will focus on what an ontology of cyber Indicators of Compromise (IOCs) looks like. This ontology is developed via analysis of IOCs definitions, structures, and examples, from established (or nascent) IOCs standards initiatives.

III. THE ONTOLOGY OF CYBER INDICATORS OF COMPROMISE (IOCS)

A. IOC: DEFINITION, STRUCTURE, AND EXAMPLE

Indicators of Compromise are defined as forensic artifacts that can be used as signs to denote a system has been compromised by an attack or was otherwise infected by malicious software [23]. The purpose of IOCs is to enable the automated detection of malicious information system (IS) activity [24]. According to Michael Cloppert, who is the lead analyst for Lockheed Martin's Computer Incident Response Team's (CIRT) Intel Fusion Team, IOCs can be classified into three categories based upon the type of compromise indicators used: computed, behavioral, and atomic [25].

Computed indicators are "computed." They are developed from material involved in the incident. A common example of this type of indicator is the hash of a known malicious file [26]. Behavioral indicators combine other indicators in order to create an overall profile of the targeted malicious behavior. Such combinations of indicators can be created from computed indicators, atomic indicators, and specific behaviors of the attacker. These component indicators may be identified during separate incident response actions, and may appear to have little investigative meaning when considered in isolation. However, when considered collectively, these indicators can be correlated to form composite behavioral indicators, which often provide more reliable indications of attacker activity. These behavioral indicators are often referred to as attacker tactics, techniques, and procedures (TTPs) [26].

Atomic indicators are fragments of data that individually, by themselves, indicate adversary activity. Examples of this include fully-qualified domain names (FQDNs), IP addresses, or email addresses. These types of indicators can be a problem because they might or might not indicate adversary activity. For example, the source IP address of the attack could very well be an otherwise-valid site. Atomic indicators frequently require inspection through examination of accessible historical data to decide if they exclusively signify hostile intent [26].

In Jason Luttgens, et al., book, titled *Incident Response & Computer Forensics* [11], 3rd edition, the creation of IOCs is defined as "the process of documenting the characteristics and artifacts of an incident in a structured manner." These "characteristics and artifacts" can include virtually any informative piece of data that flows over the network (inter-host activity) or is generated on individual hosts (intra-host activity). Due to the fact that IOCs are merely a definition, it does not provide the actual mechanism used in finding matches. The technology used to leverage this IOCs language is the SIEM. The format chosen to represent IOCs depends on the organization using them. It can be either network-based indicators, such as Snort rules, or host-based indicators such as YARA, Mandiant's OpenIOC, or Mitre's CybOX. Host-based indicators are discussed in detail in the next section of this chapter, titled Established IOCs Formats [11].

The real power of IOCs is their ability to enable IR teams to uncover maliciousness in an automated manner, either via an enterprise IR program, or via Windows Management Instrumentation (WMI) and visual basic (VB) scripting. "Deploying" IOCs results in a capability to hunt for and report on IOCs throughout the enterprise in an automated fashion. IOCs, which are basically leads generated by IR teams, are used mainly for the scoping of an incident. The idea is that the IR team would first detect some malicious activity, then create IOCs that is tailored to that activity's underlying behavior/signature, and then apply that IOC to both past and future events to see if the behavior is found elsewhere. The IR team will begin to receive notifications called "hits" once the IOCs have been deployed. Hits occur when an IOCs device finds a match for a specific rule or IOC. Validation of each hit is advisable before a response action is generated [11].

Analyses of attacks that have been conducted, whether they were successful or unsuccessful, provides the incident response team with a good "roadmap" for facilitating future discovery. The details uncovered during the analysis may provide indications that the initial attack was only the first step in a series of attacks to come. It may also provide behavioral indicators for an adversary that will allow the IR team to build a profile on that particular adversary. This makes future discovery of attacks by the same adversary possible. Learning all of this detail allows adjustments to be made to existing IOCs as

well as provides new artifacts to be developed into new IOCs. The indicator life cycle is derived from this historical attack analysis. The indicator life cycle depicted in Figure 1 is cyclic, with the discovery and application of indicators, resulting in the discovery of additional indicators.

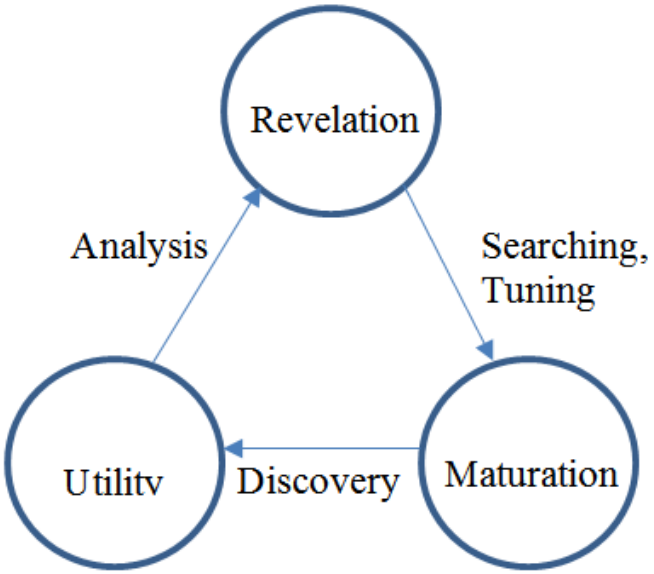


Figure 1. Indicator Life Cycle State Diagram. Source: [26].

The analysis transition from the utility state to the revelation state is where an indicator is shown to not only generate legitimate "hits," but also useful in identifying (i.e., revealing) additional incident-related artifacts that can be considered for inclusion in the growing list of indicators associated with an identified-hostile actor. These leads/ indicators, can come from many places, such as intelligence from partners, internal investigations, the FBI, internet service providers (ISPs), or open source platforms such as IOC Bucket that allows sharing of IOCs. The searching and tuning transition from the revelation state to the maturation state is where analysts articulate the best definition to leverage the newly identified indicators. As the number and quality of indicators grows, detection tools are reconfigured/retuned and collection signatures are scripted or modified as necessary and appropriate. The discovery transition from maturation state to utility state, is where the full potential of the indicator is likely realized, and the result is a

collection of true-positive “hits” that are useful in detecting the extent/scope of the targeted incident [26].

Figure 2 stacks up the numerous indicators that can be used in detecting an adversary’s actions, alongside the relative amount of effort required by the adversary to turn and continue with the intended attack, if indicators at each level are denied [14].

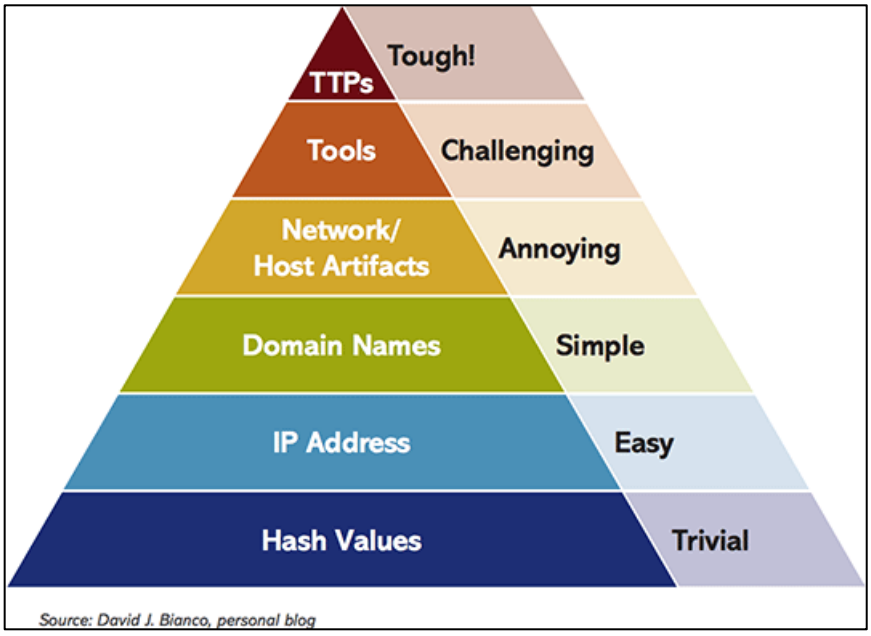


Figure 2. Pyramid of Pain with IOCs. Image from AlienVault Blogs at https://www.alienvault.com/blogs/security-essentials.

Starting with the base of the pyramid is the point where if the adversary were detected and denied, their required effort would be the lowest, are the Hash Values such as MD5 or SHA1. Hash values are regularly used to distinctively identify malicious files or malware involved in an intrusion. The adversary might possibly change an insignificant bit causing a different hash to be produced, hence making our earlier detected IOCs hash ineffective [14]. Next up the pyramid are the IP addresses. Since the adversary can alter IP addresses with minimal effort, it wouldn’t take them long to recover. This indicator would have no effect on the adversary if they used an anonymity proxy service like Tor. In contrast, Domain Names are harder to alter than IP addresses since they must be visible and registered on the internet. It is still achievable, but would

take the adversary more effort and time than IP addresses. Standard examples of network Artifacts are SMTP Mailer values, URI patterns, distinctive HTTP User-Agent, or C2 information embedded in network protocols [14]. Examples of host artifacts might be values or registry keys recognized to have been generated by specific pieces of malware, directories or files using certain names or dropped in certain places, and malicious services or descriptions or names. Uncovering attacks using host/network artifacts can be very difficult for the adversary because they force the adversary to spend a great amount of effort in trying to identify the artifact, which revealed their approach, then revise and relaunch it. Next up the pyramid are Tools, including services intended to create password crackers, malicious documents for spearphishing, or backdoors used to establish C2 communication [14]. Tool indicators could be network alert tools with fuzzy hashes and a distinct communication protocol. If these tools are detected and the breach has been secured, the adversary has to build new tools to accomplish the same purpose which slows their progress. Lastly, at the very top of the pyramid, are the Tactics, Techniques, and Procedures (TTPs). This deals with the adversary's propensities and behaviors. By negating any TTP of the adversary, we compel them to do the greatest time intensive activity; they must learn new behaviors [14].

Next, we examine in more detail the host-based and network-based indicators mentioned above. According to "Incident Response and Computer Forensics," host-based indicators are the way to implement binary classification of endpoints. Indicators are created by compiling a set of observable properties which describe a particular situation known to be suspicious. The usefulness of these observables relies upon the caliber of the members of the set. A valuable host-based indicator is comprised of numerous observables which are specific to a certain activity, yet is common enough to be applied to a spinoff of the activity. The aim of network-based indicators is much the same as that of the host-based indicators. This document points out the intent is to rapidly determine whether a specific network session is pertinent to the investigation. The characteristics and properties chosen are reliant upon the abilities of the monitoring system being used. The majority of indicators are simple such as "if a given set of bytes is present in the first *n* bytes of a session, raise an alert" [14]. These indicators however, might have a reduced

lifespan due to the fact the adversary could make changes to their procedures or tools used in the attack. If the incident response investigation runs for any large interval of time, one will probably have to edit the network signatures created for the malware repeatedly.

Now that we know what IOCs are and their purpose, let's examine where does the IOCs data come from. This data normally comes from many sources including a number of online security associations, industry and commercial groups, and numerous free IOCs-specific sites. Commercially created IOCs are frequently released by various security vendors. Typically, these feeds are very costly and are only sent to paying customers [25]. McAfee, RSA, and Symantec are a few of the security vendors providing these services. Furthermore, there are numerous academic factions such as Information Sharing and Analysis Center (ISAC) groups which distribute such data, usually separated by specific industry. Free IOCs data can frequently be located online on certain IOCs distribution sites. Various security companies utilize IOCs releases to discuss new attacks or malware [25]. There are also web-based tools that can be used for researching and sharing IOCs. IOC Bucket is one such tool that provides an easy, quick system for searching the uploaded data for a specific indicator. This tool also has a Twitter feature that sends out tweets as new IOCs are added. Lastly, custom IOCs may be developed. They may be constructed based on in-depth knowledge and analysis of our networks [25].

There are many different tools available for creating and editing IOCs. Typically, an editor will be able to create new IOCs and easily edit current ones. IOC-EDT is one such open-source and free web-based tool. Because it is web-based, it is easy to access, use, and doesn't need to be installed. It also works equally well across multiple platforms, i.e. Windows, Linux, Mac, Unix, etc. [25]. IOC Editor is another free but not open source tool that can be used for working with IOCs. It is a Windows-only GUI that allows you to import current IOCs, build your own from scratch, and shows you the difference between two IOCs records. Lastly, PyIOC is a free and open source tool that is Python-based. It is an attempt to make a fully featured editor minus the closed-source restraint that inhibits the security community from being able to really benefit from it [25].

We now examine a host-based indicator scenario. This IOC is based primarily on the artifacts created by execution of a file or the properties of the file itself. Let's say we have a Portable Executable file that is a Win32 EXE file for the Windows command line subsystem. A simple indicator we could use to describe or identify the file consists of a solitary, high-assurance check is the MD5 hash of the file as shown here [11].

```
if
{
(file MD5 hash != "e2bf42217a67e46433da8b6f4507219e")
}
then
        raise alert
```

This IOC has some extremely good attributes. It is looking for only a single, definite property which is the MD5 hash. This offers a high degree of confidence that if we get a match, we have found precisely what we were searching for [11]. The MD5 hash has a very low false positive rate so we would seldom ever get a match for something that wasn't the file we were actually searching for [11]. This IOC has a limited lifespan however because if only a single bit in the file is changed, the MD5 hash will have changed and the IOC will no longer be equivalent.

There are numerous data structures within a Windows Portable Executable file that can be examined and used to create IOCs. The header of the PE file has a compile timestamp. This is a time and date inserted by the compiler when the file is compiled. Attackers will often compile their binary and then go back and manually make alterations to it [11]. Sometimes, the compile timestamp alone will be unique enough to do a search for. Typically, it is combined with something else like the size of the file in order to reduce the chance of getting a false positive. The previous IOC we mentioned would be updated to incorporate these new conditions, as follows. This IOC validates for the MD5 hash as well as inspects the file size and compile-time stamp [11].

```
if
{
        (file MD5 hash != "e2bf42217a67e46433da8b6f4507219e")
                OR
        (
                (PE header Date/Time != "2010/08/24 01:00:23 UTC")
                        AND
                (file size != "25076")
        )
}
then
        raise alert
```

This IOC can be improved further by additional analysis of the binary executable of the file being examined. The binary has the ability to perform numerous actions on the system. For example, it can connect to sites on the internet or install a certain Windows service. These facts can be used to continue improving our IOCs by examining the artifacts associated with them. These artifacts are created on the host after the binary is executed and are not direct properties of the file itself. These attributes are good to have when the binary is no longer present on the system. Examples of this would be a DNS cache artifact pertaining to the host name the malware joins to and the exact service name created by the binary [11]. Another way to improve our IOCs is to define what the binary can accomplish. This is usually done through an examination of the import table. Our example binary has numerous imports. Any single import may not be unique because most malware uses tasks common to several other kinds of software. What would be unique, however, is the subset of the tasks normally not found jointly in a single binary. We would have to create our IOCs with many loosely or indistinct attributable properties.

Another factor to consider is that often times there is no malware associated with the attack and the IOCs would need to describe what the adversary does [11]. These IOCs would be used in detecting a normal sequence of actions that could be seen from an attacker. It is an anomaly-based indicator and can include property-based indicators with data on artifacts the active attacker left behind. The property-based indicators describe a set of known recognizable characteristics of malicious actions or software such as an MD5 hash or a registry key. Of course, we can continue to improve our IOCs if we spend additional time examining all the unique features of the binary. If we are not careful though, the indicator rapidly becomes unmanageable [11]. This is why we must pursue a compromise between too little and too much.

Lastly, we will examine a network-based indicator scenario to provide an example of a network-based IOCs. We will use the malicious binary we examined in the previous section and continue analysis to identify network signatures which could be used to recognize the appearance of that malware. For our example, we consider a malicious binary that searches for the host name practicalmalwareanalysis.com. Network monitoring can effortlessly detect this DNS lookup but if the attacker deploys different mechanisms in the malware, relying on this DNS lookup alone could be insufficient [11]. Let's assume we collected the network traffic on a live network and observed the DNS standard query via monitoring UDP port 53, which contains the main fields

DNS Query flags: 0x0100

Query Type: A

Query Class: IN

Query String: "practicalmalwareanalysis.com"

We could use this to create a signature for the data structure used in the packet itself [11]. We could refer to RFC 1035 to see the relevant excerpt that describes what we should see during the query. We would see QNAME, QTYPE, and QCLASS. As stated in *Incident Response & Computer Forensics*, QNAME is a domain name characterized by a series of labels, each label consisting of a length octet then followed by that number of octets. The domain name is terminated by the zero-length octet denoting the null label

of the root. QTYPE is a two-octet code specifying what the query type is. It includes all codes acceptable for a TYPE field. Looking at the depiction for the QNAME section of the query, we can see searching for the simple string “practicalmalwareanalysis.com” would not work [11]. The payload includes a null-terminated series of strings, each with a specific octet set aside for the length of the string. The QNAME section of the query would have the following information: [11]

Length: 0x18

String: practicalmalwareanalysis

Length: 0x03

String: com

Terminating octet: 0x00

The Snort manual would provide us with the material needed to create a signature that would signal when the sensor detected this specific query: [11]

```
alert udp $HOME_NET any - > any 53 (
        msg: “Lab03-03.exe Malware:practicalmalwareanalysis.com”;
        content: “|18|practicalmalwareanalysis|03|com|00|”;
        nocase; threshold: type limit, track by_src, count 1, seconds 300;
        classtype:bad-unknown; sid:1000001; rev:1;
)
```

Using a search that is not case-sensitive, the signature will alert when the UDP traffic includes the following content “|18|practicalmalwareanalysis|03|com|00|.” However, any lookup will initiate this alert so a notification limit is incorporated to reduce duplicate events. By isolating the malware and letting it execute in a safe environment, we can capture packets received and sent between the remote site and the malware itself. This will provide us with additional detailed data that can be used in creating our network-based IOCs [11]. An example of this is network signatures which detect the payload sent from the remote site. When the server responds with an extended

status/error message or an actual file, this particular portion of the communication will be far less likely to cause a false positive. This simple example provides a general idea of the development of a network-based lead [11]. Almost all of our incident response investigations will cover countless systems and reach countless numbers of endpoints. With so many endpoints to scan, we have to be aware of how detailed we make our indicators and the quantity of data generated by the results in order for them to be most effective.

B. ESTABLISHED IOCs FORMATS

While Snort has been recognized as the predominant standard for network-based IOCs, there is no such widely accepted standard for host-based IOCs. The three-leading host-based IOCs definitions are YARA, Mandiant's OpenIOC, and Mitre's CybOX [11]. YARA offers a language and a tool which is mainly focused on identifying and classifying malware. Mandiant's OpenIOC standard is more all-inclusive and has an IR collection tool that is publicly available called Redline. Also, Mitre's CybOX standard is all-inclusive but the only tool offered, is IOCs format conversion scripts. No complete or enterprise-grade solution is easily available for any of these three options [11].

1. YARA

According to Dias [26], YARA is an open source tool designed to assist malware researchers in identifying and classifying different malware samples. It is used to generate free form signatures which can be used to connect indicators to actors, and allows security analysts to go beyond the simple indicators of IP addresses, domains and file hashes. YARA also helps identify commands generated by the C2 infrastructure [27]. It also provides, as Dias further explains, the ability to generate descriptions of malware families founded on binary or textual patterns. Each description contains a Boolean expression and a set of strings which determines its logic. YARA has a simple and flexible rule syntax along with the following engine scanning abilities: external variables, file objects, and processes. It allows the development of custom modules as well as an extension of its engine's abilities. Yara is a multiplatform tool that can be used via user-written Python scripts or its command-line interface [27].

The syntax of YARA rules resembles, in plain text format, the C language and is comprised of condition, metadata, and strings sections. Dias [26], states each rule starts with the keyword “rule” followed by a unique rule identifier. Also, rule tags can be identified following the rule identifier. Dias points out these identifiers have to follow the same lexical conventions as the C programming language. He also states they can contain the underscore character and any alphanumeric character, but the initial character must not be a digit. Rule identifiers cannot exceed 128 characters and are case sensitive. Yara uses a number of reserved keywords that cannot be used as identifiers as seen in Table 7 [27].

Table 7. YARA Reserved Keywords. Source: [28].

all	and	any	asci	at	condition	contains
entrypoint	false	filesize	fullword	for	global	in
import	include	int8	int16	int32	int8be	int16be
int32be	matches	meta	nocase	not	or	of
private	rule	strings	them	true	uint8	uint16
uint32	uint8be	uint16be	uint32be	wide		

The information in this paragraph is taken from “Intelligence-Driven Incident Response with YARA.” The metadata section of YARA rules includes descriptive information about the rule. It has value/identifier pairs defined by the keyword “meta.” The strings definition section of the rule can be omitted if the rule does not rely on any string. This section is where strings that are a part of user-defined rules for pattern matching are defined by code sequences [27]. Each string is composed of a $ character followed by underscores and a series of alphanumeric characters. Strings can be defined

as regular expressions, or hexadecimal or text form. The condition section of the rule holds the Boolean expressions which define the logic of the rule. It tells under what conditions a process or file satisfies the rule or not. Unlike the strings definition section, the condition section is always required. Conditions will typically refer to strings that have been defined earlier in the signature by using their identifiers. This identifier will act as a Boolean variable that will equate to true if that particular string has been identified in the file, otherwise it will indicate false [27].

YARA's scan engine is offered for multiple operating systems such as Linus, MacOS X and Windows. This engine is typically invoked by Python with the YARA-Python addition or command line. The engine will compare a parsed process' memory or file object with a signature file that contains rules formatted in YARA [27].

YARA delivers wide-ranging features that go far beyond just the simple file object parsing. The most significant of these features are discussed in the rest of this section. Portable executable file parsing is available via the PE module and enables the development of fine-grained rules. This module reveals almost all of the fields existent in a PE header and allows the user to write more targeted and expressive rules [26]. The process scanning feature lets YARA scan the whole memory of a process, which results in the engine being impervious to file obfuscation techniques. This feature allows YARA rules to uncover indicators which are integrated in the install stage of the intrusion kill chain. The "yarascan" plugin scans the process address area against the YARA rules [27].

The metadata section of rules is one of the most ignored elements of YARA. In this section, the analyst can define an assortment of details concerning the rule [27]. This section will not affect the logic of the rule but will allow it to be used in tasks after processing. Metadata identifiers can be an integer, Boolean, or string. YARA rules can also support tags which are used in rule management, and output filtering.

External variables allow rules to receive data from outside. This is an extremely beneficial feature when building rules dependent upon computed indicators like file hashes. Using these external operators permits the insertion of variables for the purpose of influencing the rule match. Executing YARA inside the framework of scripting

languages allows the fetching of values from numerous sources so they can be used when invoking the scan engine [27].

YARA permits the making of rules that reference other rules. This supports the design of behavior-based signatures that contain both computed and atomic indicators. You can also create private rules in YARA that do not give any output when a match is detected [28]. When these rules are mixed with the rule referencing, they become useful. They can act as building blocks for additional rules, as well as prevent cluttering the output with irrelevant material [28]. Global rules have an effect on every rule within the same file while eliminating the need for referencing by each individual rule. They enable the ability to impose restrictions in each one of the rules at the same time.

"Intelligence-Driven Incident Response with YARA" states YARA must initially compile its defined working rule set before it can start parsing data. When managing repositories that contain thousands of rules, overall scan time can be greatly enhanced by precompiled rules. Compilation of rules takes place via two different methods. The first method this document discusses takes place at the time of conventional same time as the YARA scan engine performance. The compiled version of the parsed file from the scan engine will be stored in memory [27]. The second method discussed in this document utilizes YARAC binary to transform clear text formatted rules into binary object structure. This allows the rules to be utilized by the scan engine [27]. This provides a benefit with regards to scanning speed and keeps the rule structure concealed from prying eyes.

The aforementioned document illustrates how YARA was devised to be a very simple and fast engine and thus leaves out features like the ability to control the output. YARA's input features offer file objects and memory parsing capabilities that will output the scan engine results to the standard output for viewing. This can be a problem if any post-processing is necessary. In order to overcome this, the YARA Python extension was developed [27]. This extension permits building Python scripts containing all the YARA core features which benefits from the huge variety of Python modules.

In order to implement and execute the scan engine throughout the enterprise, a package managing platform needs to be in place. Typically, this type of platform works with architectures that are agent-based and provides central implementation management for applications [27]. The YARA scan engine can be implemented on numerous clients by encapsulating it into an adaptable package.

Now we will look at a few examples of YARA rules and their meanings in Figures 3, 4, and 5.

```
rule silent_banker : banker
{
        meta:
                description = "This is just an example"
                thread_level = 3
                in_the_wild = true
        strings:
                $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
                $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
                $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
        condition:
                $a or $b or $c
}
```

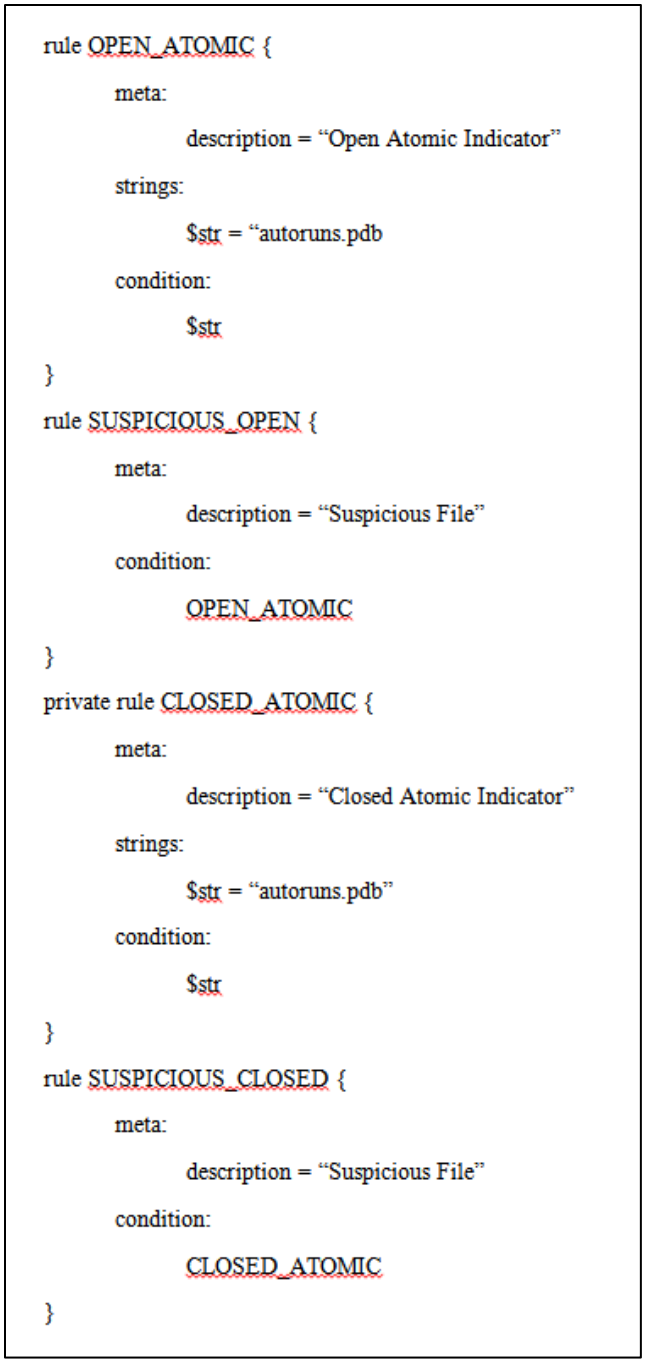
Figure 3. Simple YARA Rule. Source: [28].

The simple rule in Figure 3 tells YARA that any file that includes one of the three strings should be reported as silent_banker.

```
rule RULE_EVILMD5
{
        meta:
                description = “Suspicious MD5 Hash”
        condition:
                md5 matches /^3bb34a700e8d21acfdfe0f09208a7c01$/
}
rule RULE_EVILPATH
{
        meta:
                description = “Suspicious File Path”
        condition:
                path contains “/appdata/roaming/”
}
rule RULE_SUSP_BHV
{
        meta:
                description = “Suspicious Behavior”
        condition:
                RULE_EVILMD5 and RULE_EVILPATH
}
```

Figure 4. YARA Rule Featuring Rule Referencing. Source: [27].

Figure 4 is about rule referencing and shows the parent rule along with the referenced rules that are all reported when a match is detected. RULE_SUSP_BHV is the parent rule in the above example.



```
rule OPEN_ATOMIC {
        meta:
                description = "Open Atomic Indicator"
        strings:
                $str = "autoruns.pdb
        condition:
                $str
}
rule SUSPICIOUS_OPEN {
        meta:
                description = "Suspicious File"
        condition:
                OPEN_ATOMIC
}
private rule CLOSED_ATOMIC {
        meta:
                description = "Closed Atomic Indicator"
        strings:
                $str = "autoruns.pdb"
        condition:
                $str
}
rule SUSPICIOUS_CLOSED {
        meta:
                description = "Suspicious File"
        condition:
                CLOSED_ATOMIC
}
```

Figure 5. YARA Rule Referencing Private Rules. Source: [27].

Figure 5 from "Intelligence-Driven Incident Response with YARA" demonstrates private referencing within the condition segment. Even though the "Atomic" rules are alike, they will be guarded against possible reverse engineering because the "closed" rule is maintained private [27].

YARA is a very flexible tool that blends perfectly within the cyber threat intelligence model. It is a very simple, open source tool for creating indicators. The engines core qualities and rule flexibility, joined with a management structure are a great first step towards an intelligence driven response [27].

2. MANDIANT'S OpenIOC

Mandiant's OpenIOC is a system for sharing, recording, and defining threat data information. It enables the flexibility of modifying data on the fly as additional intelligence becomes available so that input from human subject matter experts can be incorporated [29]. The OpenIOC standard allows you to group artifacts in a logical manner. This data can then be transferred in a format that is machine readable. "Fighting Back Malware with IOC & YARA" defines IOCs or indicators as a logically categorized group of terms that describe a precise threat, whereas the language used in describing those precise groups is referred to as OpenIOC [29].

OpenIOC is an extensible XML framework to build and ingest IOCs. XML offers a well-ordered standard structure of encoding data turning it into machine-readable format [30]. This allows it to be used in numerous, standardized ways for sharing data. XML delivers numerous benefits to OpenIOC consumers. It can extend the rather small and lightweight base schema of OpenIOC with indicator sets (written in XML). Custom indicators can also be created that suit a specific setting or threat. It is also simple to build utilities that can parse or convert OpenIOC to other formats [30].

"Sophisticated Indicators for the, Modern Threat Landscape: An Introduction to OpenIOC" notes indicator terms are the specific kinds of data elements which are incorporated in IOCs. They are often categorized into an XML document. When an investigator creates IOCs, they can use as few or as many terms as needed from however many sources are necessary.

MANDIANT presently provides indicator terms for OpenIOC which detail over 500 kinds of evidence that can be collected in an organization [30]. This factor in combination with the nested logical structures of OpenIOC, have led to far greater functionality than the typical static signature based tools.

Indicators begin in complexity with merely searching for signatures of explicit artifacts. These artifacts can be registry keys, MD5 checksums, compile times, or the size of a file. They may also contain elements obtained via advanced forensic procedures like exports used by an executable or artifacts that are considerably difficult for an attacker to modify. Multiple kinds of precise indicators can be combined into one IOC allowing one specific IOC to apply to several groups of complex signatures [30].

"Sophisticated Indicators for the, Modern Threat Landscape: An Introduction to OpenIOC" points out there are other ways IOCs can be used besides just a direct query against a host. They state when probing against collected data sets, logical operators may be utilized to omit whole domains of the network or hosts being scanned. Instead of just searching for an exact file based on terms that must precisely match, IOCs may be used for matching every file that ought to reside on a specific segment of a system. An incident response investigator would gather data that is unfiltered from a system, and subsequently run IOCs against the gathered data to search for any file that stands out [30].

Examples of OpenIOC simple use cases that permit probing for forensic artifacts are [30]:

- Searching for a particular entry or set of entries in the Windows Registry
- Searching for a particular file via an MD5 hash, create data, file name, size, or additional file characteristics
- Searching for a particular object in Memory
- Joining together the entries above in assorted combinations offers improved matching and reduces the number of false positives.

These difficult methods can be joined together to permit more depth to the IOCs [30]:

- Rather than tracking down a particular file known to be bad, an incident responder can create a whitelist of files that should reside within a particular directory. This will allow them to catch any file not listed that should not be a part of that directory.

- OpenIOC logic can be used to combine collections of artifacts together to generate matches on artifacts that are from the same author or shared throughout malware families.

Describing an attacker's methodology could possibly be the most powerful method of making an Indicator. Indicators that try to detect methodology do not concentrate on a certain piece of forensic evidence directly connected to compromise or malware. Rather, they concentrate on the commonality of techniques that an attacker might use [30]. Indicators based on methodology do not necessarily illustrate a particular occurrence of compromise, but they do illustrate the result of recurrent tactics repeated by a particular group of adversaries. Essentially, making them the most difficult to write, but if done correctly, they can capture indication of a behavior that is done by only the adversary as opposed to valid users of a system [30].

For OpenIOC, unlike several other data standards used to define threat information, there is no element by element diagramming of an instantiation of a threat. The best IOCs possess the following properties [30]:

1. The IOCs are very costly for the attacker to elude. That is, to elude the IOCs, the attacker must radically change their approach, tactics, or tools.
2. The IOCs recognizes only attacker actions.
3. The IOCs must be simple and reasonable to assess. It analyzes data that is not costly to gather.

At the heart of Mandiant's incident response methods are IOCs. This is made possible by the machine-readable nature and adaptability of the OpenIOC format. The following framework taken directly from the "Sophisticated Indicators for the Modern Threat Landscape: An Introduction to OpenIOC" demonstrates how OpenIOC and IOCs make the steps of an incident response investigation possible [30].

- Initial Evidence: Responders examine and identify evidence of a compromise on either the network or host which is a solid forensic indicator of an intrusion.
- Construct IOCs for Network and Host: After the preliminary discovery of this forensic proof, the incident response investigator will construct IOCs from the current data. The particular kind of IOCs constructed will change

based on the environment, evidence, and the comfort and proficiency level of the investigator. The adjustability of OpenIOC permits an unlimited number of permutations on the way an indicator can be constructed, which gives the investigator employing OpenIOC many options to follow.

- Deploying IOCs in the Organization: After an IOC or group of IOCs have been constructed, the investigator will run these in the SIEM to look for the presence of these IOCs on other portions of the network or on other systems. For the Mandiant workflow, the IOCs are fed into the Mandiant Intelligence Response (MIR) applications, which will then connect with MIR Agents on hosts, or monitor network traffic.

- Identify Added Questionable Systems: After the IOCs have been deployed to the SIEM, other systems that have been compromised will be identified, except in the case where the initial host was the only endpoint compromised.

- Evidence Collection: Supplementary evidence is obtained from the additional systems that were identified.

- Analyze Evidence: Supplementary data is collected and analyzed. This helps detect additional intrusions, added intelligence for investigators, or false positives. These assessments permit the investigator to enhance their searches and return to the beginning of the workflow.

- Refine and Build New IOCs: Based on all of their assessments and findings, investigators can generate new IOCs as needed for the task at hand.

OpenIOC is an open source tool released by Mandiant. It includes two tools to create, edit, and use OpenIOC. The first tool is Mandiant IOC Editor. This tool permits the easy design of IOCs by utilizing a graphical interface instead of having to edit raw XML [30]. IOCs built with this editor can then be distributed to other responders within or outside of the organization. The second tool is Mandiant IOC Finder. This tool can be used to gather data from a host once the IOCs have been built. Once the data has been gathered, IOC Finder could be employed to check the IOCs against the selection of data to determine if the host corresponded to the conditions exhibited in the IOCs. Based on these results, the IOCs can be refined or used to look for additional endpoints [30].

IOCs constructed in OpenIOC let organizations describe fragments of threat intelligence in a consistent, logically structured manner. They also capture the knowledge and proficiency of human subject matter experts to a machine-readable form, which can

be quickly transferred across their organization [30]. Below is an example from “Sophisticated Indicators for the Modern Threat Landscape: An Introduction to OpenIOC” [30] of what IOCs written in Mandiant’s OpenIOC signature format would look like.

```
<?xml version=“1.0” encoding=“us-ascii”?>
<ioc xmlns:xsi=“http://www.w3.org/2001/XMLSchema-instance”
xmlns:xsd=“http://www.w3.org/2001/XMLSchema”
id=“9121b0f5-d268-479a-aa73-aa89688b0741”
last-modified=“2017-01-15T15:05:03”
xmlns=“http://schemas.mandiant.com/2010/ioc”>
<short_description>Kernel32.dll malware</short_description>
<authored_by>Authors Name</authored_by>
<authored_date>2017-01-15T15:00:45</authored_date>
<links />
<definition>
<Indicator operator=“AND” id=“5865f42f-fe4f-4b1e-87e3-ca31702f6935”>
<IndicatorItem id=“e2394125-efd8-4d46-a909-45866af9946c”
condition=“contains”>
<Context document=“FileItem” search=“FileItem/FullPath” type=“mir” />
<Content type=“string”>C:\Windows\System32\kernel32.dll</Content>
</IndicatorItem>
<IndicatorItem id=“92a2028b-81c6-40c3-b520-af7a35cb9d57”
condition=“is”>
<Context document=“FileItem” search=“FileItem/SizeInBytes” type=“mir” />
<Content type=“int”>1161216</Content>
```

</IndicatorItem>

<IndicatorItem id=“90fc74ee-7ee2-4060-bcb2-24fb240d06bb”

condition=“is”>

<Context document=“FileItem” search=“FileItem/Md5sum” type=“mir” />

<Content type=“md5”>d8973e71f1b35cd3f3dea7c12d49d0f0</Content>

</IndicatorItem>

</Indicator>

</definition>

</ioc>

3. MITRE’S CybOX

“CybOX, A Structured Language for Cyber Observables” implies CybOX is a free structured language for capturing, characterizing, specifying, and communicating events that are discernable within the domain of operation [31]. They note how an extensive array of complex cyber security use cases depends on this data to include sharing of indicators, intrusion detection, event logging, and attack pattern characterization. It also states CybOX offers a common configuration for denoting cyber observables throughout and amongst these different use cases thus improving total situational awareness, consistency, interoperability, and efficiency for the organization.

The idea of observable attributes or events in the operational cyber domain is an essential underlying component of many of the distinctive activities incorporated in cyber security. Every use case, every activity area, and frequently every associate tool vendor practices its own unique method that impedes interoperability, total situational awareness, consistency, and efficiency [31]. CybOX attempts to help with this standardization by being flexible and not targeting just an individual cyber security use case. Its purpose is to provide an adaptable solution that can be used for every cyber security use case that has to handle cyber observables. Also, it is designed to permit very clear quality definitions of cyber observable occurrences within an operational setting [31]. By

identifying a common structured representation tool for these cyber observables, CybOX facilitates meticulous automated sharing, analysis heuristics, mapping, and detection.

CybOX is aimed at supporting a broad range of important cyber security domains to include [31]:

- Digital Forensics
- Threat characterization and assessment
- Operational event management
- Cyber situational awareness
- Malware characterization
- Logging
- Indicator Sharing
- Incident response
- Etc.

By using the CybOX language to capture and share relevant observable properties or events, or define rules and indicators, logical pattern constructs can be tied to real-world evidence of their presence or occurrence for attack characterization and detection. Incident response personnel can benefit from all these abilities to investigate occurring incidents, improve imminent attack prevention, detection, and response, and improve overall situational awareness [31].

There is a wide assortment of cyber observable use cases [32]:

- Improved distribution between each cyber observable stakeholder
- Possible capability to analyze data from all kinds of tools and all vendors
- Identify new attack patterns
- Detection of malicious activity though attack patterns
- Facilitate automated signature rule generation
- Etc.

CybOX has two fundamental XML schemas to deliver the essential functionality and structure of CybOX: CybOX Common and CybOX Core. CybOX objects, which are detailed in separate schema files, are accurate characterizations of specific kinds of observable cyber entities, such as a DNS query, HTTP session, and a Windows Registry Key [33]. XML namespaces deliver a means of preventing naming conflicts for attributes and elements. Every CybOX XML Schema describes a distinct namespace, permitting the integration of data types and fields among and within schemas. When constructing a CybOX Object, it is necessary for an author to describe their own schema fields and types namespace to exist within [33].

The following steps show how to use and create a new CybOX Object:

1. Decide what needs to be represented in CybOX.
2. Determine what attributes/fields could be used to characterize that CybOX Object.
3. Map those field data types to existing CybOX Object Property Types.
4. Review existing CybOX Objects to see if the capabilities defined in steps 1–3 are already supported by an existing CybOX Object. Identify capability gaps if an existing CybOX object supports a subset of desired capabilities.
5. Define a namespace for the object.
6. Create the object schema.
7. Add documentation to the schema.
8. Use the newly-created object in CybOX content. [34]

```
<xs:schema elementFormDefault="qualified"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:cyboxCommon="http://cybox.mitre.org/common-2"
 xmlns:IPAddressObj="http://example.com/objects#IPAddressObject-1"
 targetNamespace="http://example.com/objects#IPAddressObject-1"
 version="1.0">

 <xs:import namespace="http://cybox.mitre.org/common-2" schemaLocation="http://cybox.mitre.org/XMLSchema/common/2.1/cybox_common.xsd"/>

 <xs:element name="IP_Address" type="IPAddressObj:IPAddressObjectType"/>

 <xs:complexType name="IPAddressObjectType">
  <xs:complexContent>
   <xs:extension base="cyboxCommon:ObjectPropertiesType">
    <xs:sequence>
     <xs:element name="IP_Address_Value" type="cyboxCommon:StringObjectPropertyType" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="category" type="IPAddressObj:CategoryTypeEnum" use="optional"/>
    <xs:attribute name="is_source" type="xs:boolean" use="optional"/>
    <xs:attribute name="is_destination" type="xs:boolean" use="optional"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>

 <xs:simpleType name="CategoryTypeEnum">
  <xs:restriction base="xs:string">
   <xs:enumeration value="ipv4"/>
   <xs:enumeration value="ipv6"/>
  </xs:restriction>
 </xs:simpleType>
</xs:schema>
```

Figure 6. CybOX Object Schema Characterizing IP Address Information. Source: [34].

"How to Create a CybOX Object" [34] explains the XML snippet in Figure 6. The portion highlighted in yellow, describes the root xs:schema element and is used to establish the necessary schema imports and namespace definitions. The first line is used to denote that every element must be namespace qualified, if it is used by an XML instance document in this schema [34]. The next segment explained by this document, highlighted in blue, presents two new global XML structures: implementation of "IPAddressObjectType" of ObjectPropertiesType and "IP_Address" element. Extension

of the first new global XML structure provides fields like object_reference and Custom_Properties. The “IP_Address_Value” add category attribute is used for declaring IPv6 or IPv4. If the IP address is a source address, the “is_source” attribute is used. If the IP address is a destination address, the “is_destination” attribute is used [34]. The last segment, highlighted in green, allows the enumeration of category values to express if the designated IP address is IPv6 or IPv4.

Below are the steps of a Theoretical Flow for a Modern Security Incident [32].

1. There is an attack on an information system that involves a vulnerability exploit, malware plus command and control, and social engineering.

2. Operational sensors enabled by CybOX detect anomalous activity and then report it in CybOX/CEE (Common Event Expression) formats. CEE is a set of specifications to define parsing information, transport, logging recommendations, taxonomy, and syntax about event records.

3. Automated analysis rules & tools try to correlate anomalous activity against CybOX-enhanced CAPEC (Common Attack Pattern Enumeration and Classification) attack methods but find no correlating methods. CAPEC is a catalog of attack patterns with a comprehensive schema and classification catalog built to assist in the making of secure software that is available to the public.

4. Incident has been reported – Incident Management Response procedure is started.

5. IR staffs acquire a discovered fact of an incident in CybOX-compatible formats, plus CEE.

6. IR staffs discover malware as part of the current attack.

7. Malware is subjected to automated analysis (static and/or dynamic) and the outcome is captured in Malware Attribute Enumeration and Characterization (MAEC) (CybOX- incorporated) language. MAEC is a standardized language for sharing malware information that is based upon attributes such as artifacts, attack patterns, and behaviors.

8. Malware analysts are capable of associating the existing malware instance with a wide array of pre-existing malware samples and examine data from MAEC-enabled repositories.

9. Malware analysts acquire a new discovered fact about the malware in MAEC format, as well as the Common Weakness Enumeration (CWE) or Common Vulnerabilities and Exposures (CVE) exploited. CWE is a software community project whose objectives are to create a catalog of software weaknesses and vulnerabilities. Common Vulnerabilities and Exposures (CVE) as defined by cve.mitre.org, "is a list of information security vulnerabilities and exposures that aims to provide common names for publicly known cyber security issues."

10. Sample and examined data from the existing malware instance are inserted into applicable malware repositories.

11. CybOX observables pertaining to malware effects on hosts are separated from MAEC content to produce Open Vulnerability and Assessment Language (OVAL) checks to decide if there have been any host affected/infected by the existing malware instance. Open Vulnerability and Assessment Language (OVAL) is an information security standard accepted internationally and used for promoting security information that is openly available to the public.

12. OVAL checks are disseminated and run against additional regions of the organization to define the extent of the compromise.

13. IR/IM staff employ suitable remediations/mitigations to negate the consequences of the attack.

14. A new CAPEC attack pattern is composed to define this new perceived attack behavior, and is enhanced as suitable with CybOX content detected for this pattern in the operational environment.

15. IR/IM staff issue appropriate alerts for the detected incident with the new MAEC bundle, CAPEC pattern, and associated CEE/ CybOX content.

16. Secure expansion takes advantage of this new CAPEC pattern to: structural threat analysis, define/refine appropriate security requirements; security testing and secure code review, guide control selection; identify applicable CVE vulnerabilities & CCE structure issues, CWE weaknesses; prioritize applicable CAPEC patterns based on real-world observed frequency described through automated observation of CybOX patterns in the operational environment.

CybOX makes it easier and faster to share information within and outside of an organization. It permits the entire information security community to add to and extend the context of threat information and threat intelligence.

C. SUMMARY

There is currently no commonly accepted data format standard for incident response teams to utilize for sharing Indicators of Compromise (IOCs). This causes the processing and sharing of IOCs to be a manual process which impacts participation. Vendor agreement on any standard has been extremely limited.

Mandiant's OpenIOC uses an XML scheme for defining its signatures making them easy to create and use by analysts [35]. XML format can be parsed easily and used to hide unnecessary code from its users. This design's advantage is based on XMLs extensive use and ease of processing with tools [35]. XML users even have the added ability of writing their own custom indicators that are more suitable for their specific environment. This format also allows users to whitelist files, in order to determine which files were already present and safe, and therefore exclude them from the search. However, OpenIOC has limited commercial adoption, viewed as a "vendor" solution, and provides no support for describing Tactics, Techniques, and Procedures (TTPs) [35].

For YARA, the rules are recognized by .yara extensions and are defined in a plain text structure for technology independence. YARA supports many conditions that employ these rules by means of Boolean logic to create more formidable detection and conditional operations [35]. *Analysis of Malware Classification Schemas* points out YARA is an open source tool that anyone can access and adapt to suit their needs. It is a system for users who don't need large, complex frameworks like CybOX. Many tools are capable of easily taking a standard YARA scanner output and processing and formatting them in numerous ways due to their plain text format. However, processing, categorizing, and managing a large number of rules could be more difficult to do since the parsing of plain text format is not a trivial task and requires a lot of custom code [35]. Although, if working with a standardized language such as XML, a huge number of tools can be used virtually right out of the box. Nonetheless, YARA rules could significantly reduce the number of required declarations that would otherwise produce an enormous element count, making YARA very readable and maintainable [35].

CybOX, a vendor neutral tool, provides an XML schema that can be used to define "objects." One can use a consistent XML-based schema to describe the files that are most tempting for an adversary to go after [36]. CybOX has the added capacity to represent "Events" or behaviors. This allows the possibility for full range transcription of an adversary's actions within its framework. Because CybOX is very versatile, we have the ability to not only describe the observables we plan to seed our network with, but we can additionally convert simple intrusion data into multilayered Indicators of Compromise. For instance, if we built a file called *banking.txt* and then observed it for unauthorized access, we would be able to not just identify that access, but also log information about what additional system files it accessed, what process on the system opened its file handle, and what network ports it is currently utilizing [36]. The logging of all this data is supported by CybOX making it a perfect format to enable future incident response actions after a compromise is identified. Also, CybOX provides a comprehensive list of elements to build IOCs and can be integrated with other tools like CAPEC and MAEC, under STIX, for robust IOCs development [36].

Comparing the formats, each format deals with the task of identifying a file of a given size, name, path, and hash. OpenIOC and CybOX both have an out-of-box provision for these requirements whereas YARA needs a little assistance from an external tool, which is a Python script [35]. The strength of YARA lies in its lightweight, flexible design that provides users the option of using regular expressions during scanning. Its disadvantage comes from the fact it is not a supported format but is only advanced through contributions made by users. Also, because it is not a simple format like OpenIOC which is XML based, its functions cannot be extended very easily [34]. The optimal format for users would be a combination of the CybOX expression strength and objectives, the flexibility of YARA, and the scanning capabilities and tools of OpenIOC.

YARA can be used on all platforms because it is written in Python and its signatures are in plain text. OpenIOC can only be used on Microsoft Windows but it does provide a memory forensic tool for Mac OS called Memoryze [35]. CybOX can be used on all platforms because it is XML based, and there are no platform requirements for XML files. OpenIOC is proprietary because of paid support and indicator releases but

CybOX and YARA are free for public use. Since they all have their pluses and minuses and cannot meet 100% of the users' needs, there is really no "best" IOCs format [35]. It all depends on the specific needs of an individual organization and what they are trying to accomplish [37]. A more concise comparison of the three can be seen in Table 8.

Table 8. Comparison of YARA, CybOX, and OpenIOC. Source: [35].

Properties	YARA	CybOX	OpenIOC
Signatures	plain text	XML	XML
Default scanning capabilities	Yes	No	Yes
Platforms	All	All	Microsoft Windows
Proprietary	No	No	Yes

Chapter IV will present real-world examples of Indicators of Compromise (IOCs). This will be accomplished by describing and presenting IOCs examples for ZeuS (Trojan Horse), ATM malware attacks, and NetIQ in detail.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. REAL-WORLD EXAMPLE

A. ZEUS

The real-world example that will be discussed in this chapter is ZeuS, also known as Zbot. ZeuS is a Trojan horse malware kit that runs on versions of Microsoft Windows. It steals IDs like online banking accounts by using Web injection [38]. It was first identified in 2007 when it was used to steal data from the U.S. Department of Transportation. In 2009, it was discovered that ZeuS had compromised over 74,000 FTP accounts on the following companies' websites: NASA, Oracle, Bank of America, Amazon, Cisco, ABC, Monster.com, Play.com, and BusinessWeek. The source code for ZeuS was leaked in 2011 and several variants have since been discovered [38].

ZeuS is made up of two main components. The first is a panel that allows Web command and control giving its operators the ability to execute and monitor payloads on host that have been compromised. The second is the generation of the Zeus bot executable which can be delivered via a website or the panel.

There are two versions of Zeus, each one providing different functionality. The earlier version had multiple hidden files located in a hidden directory. This hidden directory included the encrypted payload, configuration file, and log file. It was placed in the system32 directory. These files were obscured utilizing the NtQueryDirectoryFile API within the Windows API hooks. Therefore, if a victim of Zeus were to use Windows Explorer to search their system directory, they would not be able to see these hidden files.

When Zeus is executed, thread injection is used in an effort to infect some of the other processes resident on that system. Explorer.exe, winlogon.exe, and svchost.exe are the processes most often infected. The code located in the memory of these processes will be found in high memory segments as a result of obscure thread injection.

ZeuS malware behavior/characteristics include: gathering data of infected machines, anti-forensics, code injection, and Web injection. ZeuS obfuscates important strings (as seen below). The strings are decoded upon their execution. De-obfuscated

strings are good indicators of a Zeus malware infection [38]. These de-obfuscated strings in Figure 7 are necessary for recognizing the payload code.

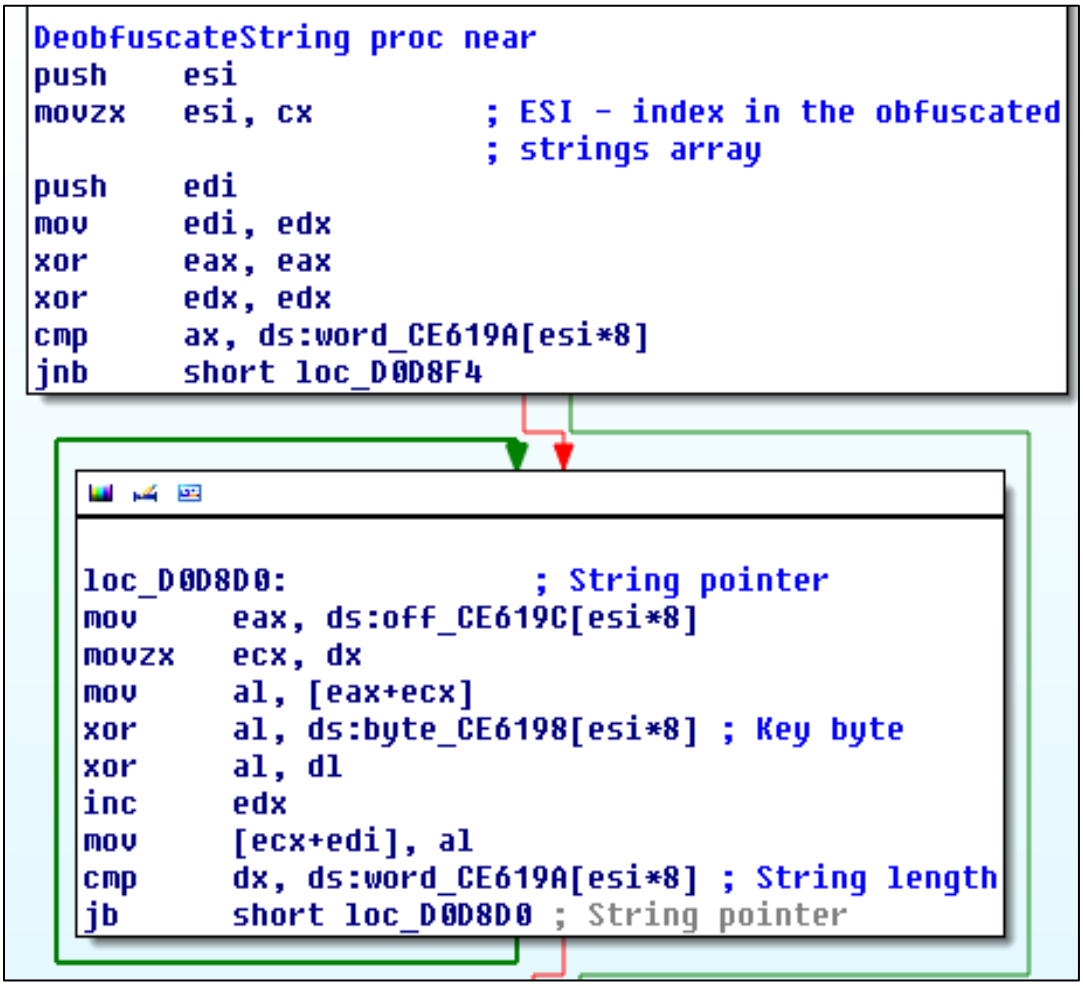


Figure 7. De-obfuscated Zeus String Algorithm. Image from Bromium at https://labs.bromium.com.

Figure 8 presents some indicators of ZeuS variants. In the “Imodule” variant, top diagram, many obfuscated strings are added. The Citadel variant, depicted in the bottom diagram, is used for detecting sandboxes for anti-analysis [38].

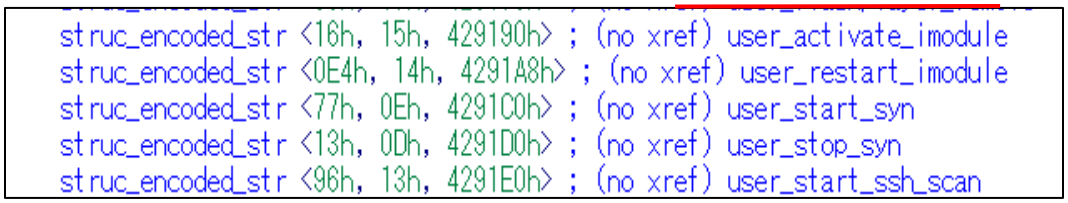


Figure 8. Indicators of ZeuS Variants. Source: [38].

Figure 9 illustrates IOCs that can be used for detecting the Zeus malware using OpenIOC.

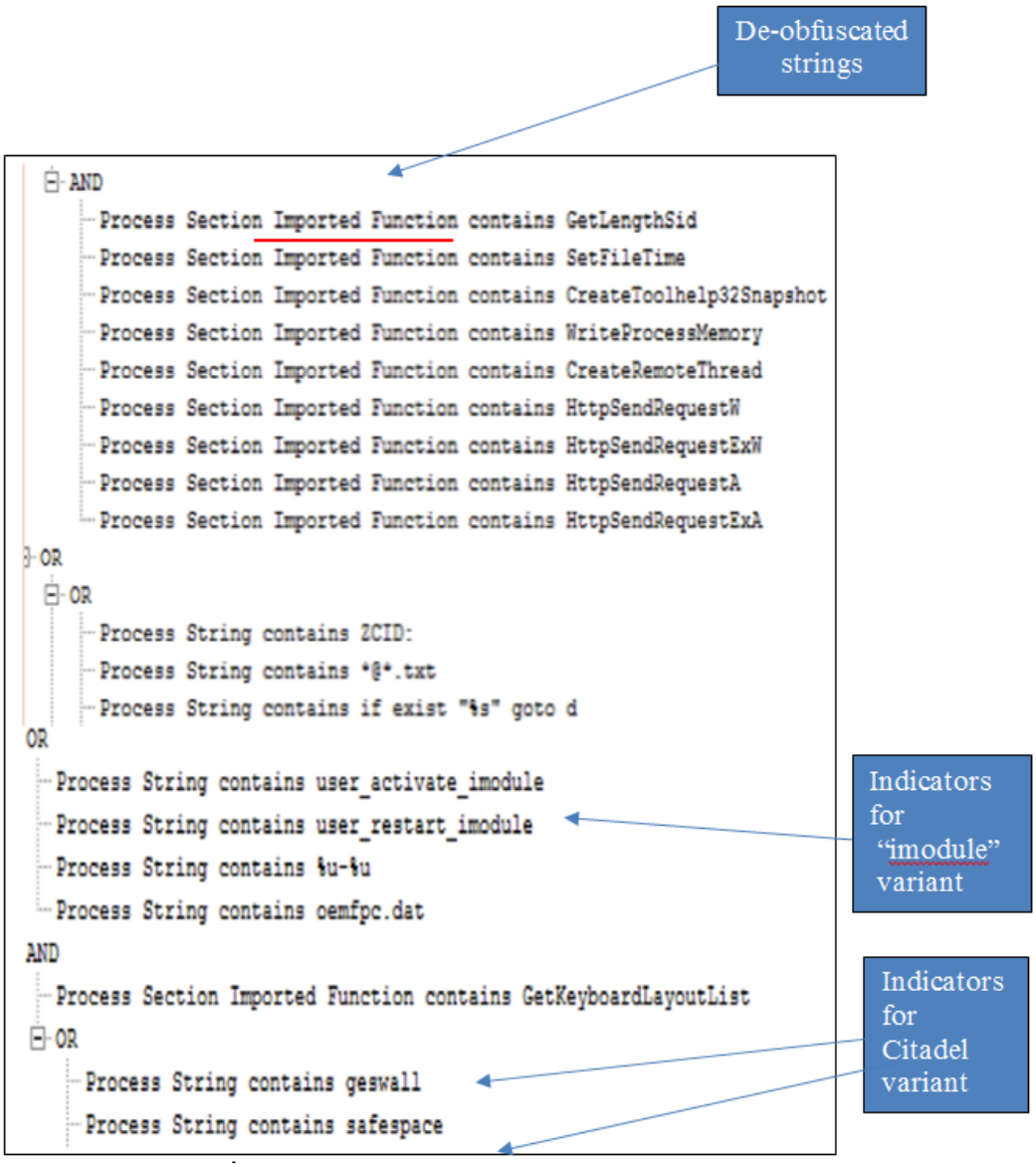


Figure 9. Mandiant's OpenIOC Indicators of Compromise for ZeuS [38].

B. ATM MALWARE ATTACKS

The next real world example to be discussed is ATM malware attacks. In recent years, new cyber-attacks targeting ATMs have been discovered. The latest techniques do

not need skimmers or the other usual physical tools. However, these attacks utilize malicious code on operating systems that are unsupported with unpatched vulnerabilities [39]. This code will be uploaded to the terminal in one of two ways. It can be uploaded directly into the terminal or via remote system access. The spoils resulting from these activities range from large cash withdrawals to sensitive data being exposed. Frequently, the attacks mix the cyber and physical realms, using partners who physically collected the money once the terminal was infected. There are four malware subfamilies that specifically target ATMs [39]: Backdoor.Ploutus, Backdoor.Tyupkin, Backdoor.ATM.Suceful, and Backdoor.GreenDispenser.

Backdoor.Ploutus was one of the first ATM malware variants to be publically disclosed and is typically installed via USB. A mobile phone tied to the ATM is used to control the malware via SMS messages. This allows the attacker to control the operating system of the ATM. There are two separate SMS commands [39]:

- The first includes an activation ID to allow the Ploutus malware on the ATM.
- The second includes a command to distribute the money.

The phone forwards authorized SMS messages as UDP or TCP packets to the ATM OS. The ATM network packet module collects the UDP/TCP packet and (if valid) executes them, potentially resulting in the machine immediately dispensing cash. The amount of cash distributed is frequently pre-configured into the malware, and the cash is often collected in person by an associate of the attacker [39].

Backdoor.Tyupkin is a malware family that can be installed through physical access to the ATM terminal by means of a bootable CD, or by means of RDP from an alternative device on the network. This malware family manipulates the NCR Persona chain of ATM machines which run Microsoft Windows 32-bit OS. As Maccaglia and Myers explain, irrespective of the means used to infect the ATM system, two files are copied onto the ATM machine:

- An executable (the malicious binary itself)
- A debugging file (responsible for imbedding the malware in the registry before it is deleted) [39]

The malware offers the attacker (or the assistant) direct access to the terminal by means of the ATM's keypad upon entry of the correct passcode and session key. The terminal then prompts the attacker to select which cash box to dispense the cash from. The amount of cash is limited by how much is physically available in the machine [39]. The malware also restricts the ATM's communication on the local area network, almost certainly to interrupt remote monitoring or troubleshooting. This malware could also be configured to operate during specific time windows [39].

In September 2015, a new ATM malware variant, named Backdoor.ATM.Suceful, was discovered. Originally the sample was uploaded to VirusTotal (VT) from a Russian IP address. According to the timestamp, it was likely composed on August 25, 2015 [39].

This malware family is possibly still being developed and being tested by its authors. Nevertheless, the sophisticated capabilities of this malware signify the authors are evolving and planning to steal data that hasn't previously been harvested by any other ATM malware. Presently documented capabilities of Backdoor.ATM.Suceful include [39]:

- Suppressing ATM sensors to avoid detection
- Reading all the debit/credit card track data
- Control of the malware via ATM PIN pad
- Retention or ejection of a card inserted into the ATM
- Reading data from the EMV chip4 of the card

Backdoor.ATM.GreenDispenser is the latest entry onto the ATM malware scene and was initially discovered in Mexico in September 2015. Initial analysis suggests this malware must be installed manually. This malware is similar to the Tyupkin family in functionality, but does display some unique functionality [39]:

- Two-factor identification composed of a hardcoded PIN, and a second one obtained by decoding a QR code
- Malicious code will only run on a system whose time and date is post-September 2015

It communicates with the hardware of the terminal, such as the cash dispenser and the PIN pad. The malware could be constructed to show a message to the potential ATM user, written in Spanish or English, indicating the machine is out of service. While routine cardholders may walk away when seeing this error, the attackers merely type in an access code to contact the malware's menu and gain access to the system [39]. Figure 10 shows how the YARA rules match the ATM sample analyzed [39]. Figure 11 shows the YARA rules used for describing the malware variants.

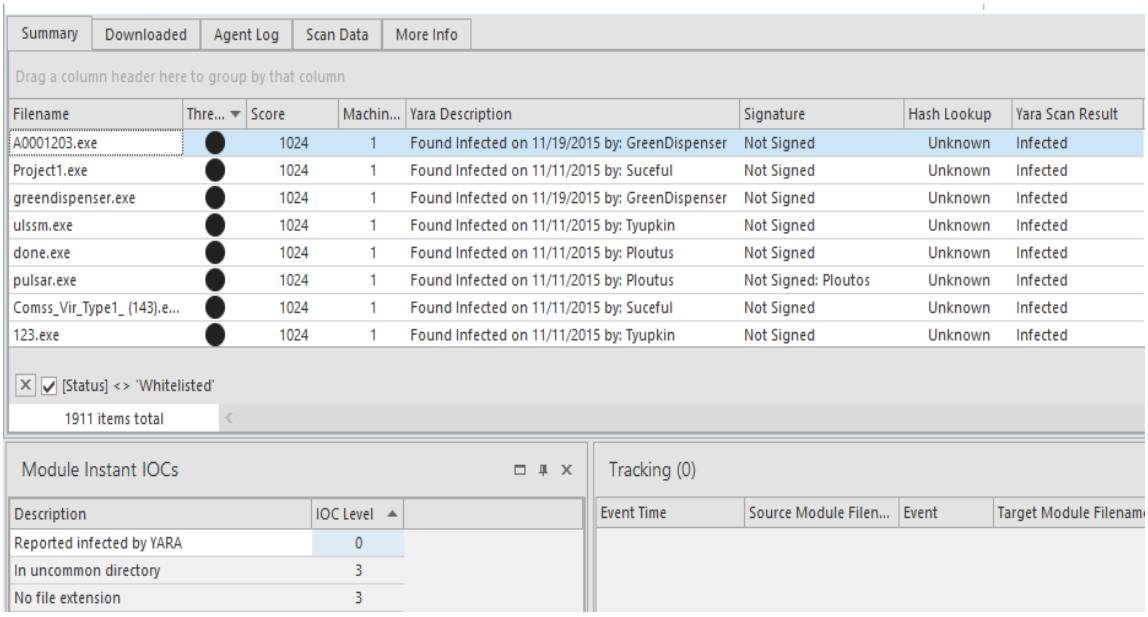


Figure 10. YARA Rules for ATM Sample. Source: [39].

```
rule Ploutus {
strings:
$s1 = "Confuser v1.9.0.0" wide ascii
$s2 = "Ploutos.exe" wide ascii
condition:
all of them and
//MZ signature at offset 0 and ...
uint16(0) == 0x5A4D and
//... PE signature at offset stored in MZ header at 0x3C
uint32(uint32(0x3C)) == 0x00004550
}
rule Tyupkin {
strings:
$s1 = "\\ulssm.exe" wide ascii nocase
$s2 = "\\AptraDebug.lnk" wide ascii nocase
$s3 = "AptraDebug" wide ascii nocase
$s4 = "ulssm.Form1.resources" wide ascii nocase
$s5 = "MSVCR80.dll" wide ascii nocase
$s6 = "MSXFS.dll" wide ascii nocase
condition:
all of them and
//MZ signature at offset 0 and ...
uint16(0) == 0x5A4D and
//... PE signature at offset stored in MZ header at 0x3C
uint32(uint32(0x3C)) == 0x00004550
}
rule Suceful {
strings:
$s1 = "vcl60.bpl" wide ascii nocase
$s2 = "Project1.exe" wide ascii nocase
$s3 = "SUCEFUL"  wide ascii nocase
$s4 = "msxfs.dll " wide ascii nocase
condition:
all of them and
//MZ signature at offset 0 and ...
uint16(0) == 0x5A4D and
//... PE signature at offset stored in MZ header at 0x3C
uint32(uint32(0x3C)) == 0x00004550
}
rule GreenDispenser {
strings:
$s1 = "dispenserprogm" wide ascii nocase
$s2 = "del.exe" wide ascii nocase
$s3 = "sdelete.pdb" wide ascii nocase
$s4 = "MSXFS.dll" wide ascii nocase
$s5 = "sdelete" wide ascii nocase
condition:
all of them and
//MZ signature at offset 0 and ...
uint16(0) == 0x5A4D and
//... PE signature at offset stored in MZ header at 0x3C
uint32(uint32(0x3C)) == 0x00004550
}
```

Figure 11. YARA Rules for Describing the Malware Variants. Source: [39].

C. CANES

This last example is included because this thesis is part of a larger "CANES CDOSS" initiative aimed at introducing a CDOSS IR capability to CANES-outfitted Navy vessels. NetIQ SM is the current SIEM deployed onboard U.S. Navy ships. At the time of this thesis, the underlying IOCs for the 1066 rules within CANES, were not available. However, some examples of how rules are written in NetIQ will be presented and provide insight into how the 1066 rules were constructed.

CANES rules are built into the NetIQ security manager (SM). Rules are entered into the security manager via scripts using VBScript or JScript. VBScript and JScript are both Microsoft scripting languages. The scripts are written in XML utilizing a combination of Boolean logic and regular expressions. NetIQ does not specifically use YARA, CybOX, or OpenIOC for writing their rules, but instead utilizes its own proprietary format for them.

All of the rules presented in this section come directly from the *NetIQ Sentinel User Guide* [40]. The first NetIQ example is for detecting a spreading attack. This type of rule is considered a correlation rule because a comparison must be made between a current event and a past event. The expression for this rule is as follows: filter(e.TaxonomyLevel1="Attack") flow window(w.dip=e.sip, filter(e.rv51="Attack"), 15m). The second example is a rule that will detect whether or not the source IP address of a current event matches one in an event that occurred 60 seconds ago. The past events would be limited to those containing source IP addresses within the specified subnet. The rules is as follows: window(w.sip = e.sip, filter(e.sip match subnet (10.0.0.10/22),60). The third example is a rule that can be considered a "domino effect" kind of rule. This type of rule would be appropriate for an attacker who has exploited a vulnerable system and then used that as an attack platform against other systems. The rule is expressed as follows:

> filter(e.XDASTaxonomyName = "XDAS_AE_IDS_PROBE" OR e.XDASTaxonomyName = "XDAS_AE_IDS_PENETRATE") flow window((e.sip = w.dip AND e.dp = w.dp AND e.evt = w.evt), filter(e.XDASTaxonomyName = "XDAS_AE_IDS_PROBE" OR e.XDASTaxonomyName = "XDAS_AE_IDS_PENETRATE"), 1h).

The fourth example is a rule that identifies if your system was subject to a potential security breach following a denial of service attack. This rule will cause an alert if a service within the destination of the attack stopped within 60 seconds of the attack. This rule is expressed as: filter(e.rv51="Service" and e.rv52="Stop") flow window (e.sip = w.dip, filter(e.XDASTaxonomyName = "XDAS. The last example is a rule that will detect whether an attack came from outside your firewall. It will check whether or not an IDS attack event detected inside your network passed through your firewall within the past 10 seconds. The rule is expressed as follows: filter(e.TaxonomyLevel1="Attack") flow window(w.dip=e.sip, filter(e.rv32="FW"), 10).

D. SUMMARY

This chapter covered three real-world examples of IOCs and how they can be used to write good rules for detecting malicious activities on a network. Each example used a different IOC format. This provided a distinct perspective on how each IOC's format could be used in writing rules based on the type of malicious activity and the needs and preferences of the user.

Chapter V, the final chapter, summarizes the main points of this thesis research and proposes recommendations for future research.

V. CONCLUSIONS AND FUTURE WORK

The Navy has placed almost all of its warfighting abilities onto mission-essential cyber systems. These systems provide high-speed automation, but bring with them potential vulnerabilities. These vulnerabilities can be mitigated through leveraging the current SIEM technology and the incident "first responders" not only onboard ships, but within any organization that is concerned with security and protection of their networks. The SIEM's ability to accurately collect, detect, correlate, and alert on possible security-related events is mission essential. The SIEM's effectiveness is limited by the quality of the rules that are "fed" to it. The "ruleset" is the heart of the SIEM and the underlying foundation of this "ruleset" are good IOCs. Good IOCs will result in SIEMs that provide more reliable and rapid incident detection, and provide better data with which to respond to those incidents. Well defined IOCs will, in the final analysis, result in a SIEM that detects more true positives, and suffers fewer "falses"—whether negative or positive.

A. CONCLUSIONS

This thesis provides an ontology of indicators of compromise (IOCs). An ontology, in the information science context, "is a formal naming and definition of the types, properties, and interrelationships of the entities that really, or fundamentally, exist for a particular domain of discourse" [41]. In this research, the domain of discourse is cyber IOCs. In Jason Luttgens et al., book, *Incident Response & Computer Forensics*, 3rd edition, IOCs creation is defined as "the process of documenting the characteristics and artifacts of an incident in a structured manner." The text goes on to state that "the goal of IOCs is to help you effectively describe, communicate, and find artifacts related to an incident." It is noteworthy that IOCs are only definitions. To actually *affect* detection of an incident, these definitions must ultimately be actualized via technology. The current state of the practice regarding this technology, is the security control (tool) typically referred to as Security Information and Event Management (SIEM). In order for the SIEM to be effective, theses definitions must be turned into a high-quality ruleset and

supplied to the SIEM. They play a vital role in the detection, as well as the investigation phases, of the incident life cycle.

The purpose of this thesis is to explore the existing space of the IOCs domain of study (i.e., its current ontology), to summarize it, and to—if efficacious—suggest "extensions" or alterations that would best benefit the incorporation of well-defined cyber IOCs into a target SIEM solution. The current U.S. Navy SIEM solution for the CANES environment is NetIQ.

There is currently no preferred or accepted standard for representing IOCs. The three nascent IOCs standards examined in this thesis are CybOX, OpenIOC, and YARA. The definition, structure, and examples of each standard were examined and then compared in Chapter III. Each standard has its own pluses and minuses. CybOX and OpenIOC offer their users the ability to create signatures in XML whereas YARA offers this ability in plain text. YARA and CybOX can be used on any platform but OpenIOC is specific to Microsoft Windows only. No one standard is better than the other. It depends on the organization and their security needs. The Navy currently uses Snort and Microsoft scripting languages built into NetIQ to write their rules. When compared to the three nascent standards in this thesis, OpenIOC seems to be the closest match.

A research objective for this thesis has been to help enhance the overall quality of the cyber incident response capability, by informing would be SIEM operators and developers of the structure and semantics of the IOCs that lie at the heart of a SIEM's functionality.

B. FUTURE WORK

Follow on work to this thesis would be identifying the IOCs specific to the shipboard environment and using these to generate rules specific to NetIQ onboard U.S. Navy ships. This can be done through the creation of new rules along with the modification of the existing 1066 rules currently built into the SIEM. This would allow the SIEM's ruleset to be fine-tuned producing fewer false positives and more accurate results. Examining the IOCs used in creating the current NetIQ ruleset will provide a

better understanding of the current rules, along with their purpose, and allow them to be adjusted or deleted as necessary.

Another focus for future research is the training and education of Sailors/analysts that are part of the incident response process. This training would be NetIQ-specific and provide a quick reference guide to what IOCs are, what good IOCs looks like, and an example format. It would also provide examples of strong rules and the IOCs used in building them. This would provide the analyst creating or modifying these rules with a quick reference guide as to what the best and most fruitful IOCs are. This could also benefit the intelligence analyst who is providing these IOCs to be used in rule creation. It would give them a better understanding of what data to look for in their daily intel searches. It would allow them to act quicker and with greater confidence on any piece of information they believe is IOCs worthy.

A combination of these suggestions will provide the best overall enhancement to the SIEM onboard Navy ships, resulting in a hardening of naval networks. It will provide a more fine-tuned system and better trained personnel. It will provide the skillsets necessary for the incident "first responders" to quickly and more accurately identify possible security-related incidents or events. This will help identify and stop future system compromise.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Joint Chiefs of Staff, Joint publication 3–12 (R), *Cyberspace Operations*, 2013. [Online]. Available: http://www.dtic.mil/doctrine/new_pubs/jp3_12R.pdf.

[2] ICS-CERT, cyber threat source descriptions. (2005). [Online]. Available: https://ics-cert.us-cert.gov/content/cyber-threat-source-descriptions.

[3] The Department of Defense. (2015). The Department of Defense cyber strategy. [Online]. Available: https://www.defense.gov/Portals/1/features/2015/0415_cyber-strategy/Final_2015_DOD_CYBER_STRATEGY_for_web.pdf.

[4] A. C. Henning and J. Rollins. (2009, Mar.). *Comprehensive National Cybersecurity Initiative: Legal Authorities and Policy Considerations*. Congressional Research Service. CRS R40427. [Online]. Available: https://fas.org/sgp/crs/natsec/R40427.pdf.

[5] P. Singer and A. Friedman, *Cybersecurity and Cyberwar, What Everyone Needs to Know*. New York, NY: Oxford University Press, 2014.

[6] S. Bucci, P. Rosenzweig, and D. Inserra. (2013, March 28). *A Congressional Guide: Seven Steps to U.S. Security, Prosperity, and Freedom in Cyberspace*. Backgrounder No. 2785. [Online]. Available: http://www.heritage.org/defense/report/congressional-guide-seven-steps-us-security-prosperity-and-freedom-cyberspace.

[7] R. Kissel, *Glossary of Key Information Security Terms*, 2nd ed. Gaithersburg, MD: National Institute of Standards and Technology, 2013, p. 58.

[8] *Framework for Improving Critical Infrastructure Cybersecurity*, 1st ed. Gaithersburg, MD: National Institute of Standards and Technology, 2014, pp. 1–41.

[9] J. Eykyn et al., "Applied cyber operations," Capstone Project Report, Dept. Computer Science, Naval Postgraduate School, Monterey, CA, 2016.

[10] Department of Defense, Chairman of the Joint Chiefs of Staff Manual, *Cyber Incident Handling Program*, J-6 CJCSM 6510.01B, 2012. Department of Defense, Arlington, VA.

[11] J. Luttgens, and M. Pepe, and K. Mandia, *Incident Response & Computer Forensics*, 3rd ed. New York, NY: McGraw-Hill Education, 2014.

[12] P. Cichonski *et al*. (2012, August). "Computer security incident handling guide." *International Journal of Computer Research*. [Online]. 20(4). Available: http://search.proquest.com/docview/1623314374.

[13] J. Vacca. *Network and System Security*, 2nd ed. Elsevier, 2014.

[14] B. Ramachandran. (2014, Dec. 30). Anything connected, exploring connected technologies, trends, businesses and ecosystems. [Online]. Available: https://connectedtechnbiz.wordpress.com.

[15] R. Schaeffer, JR., *National Information Assurance (IA) Glossary*, CNSS Instruction No. 4009. Fort Meade, MD: Committee on National Security Systems, 2010.

[16] *Support to Computer Network Defense (CND).* DOD Instruction O-8530.2. DOD Chief Information Officer, Washington, DC, 2001.

[17] National Security Agency/Central Security Service, Information Assurance Directorate. (2013, Dec.). *Spotting the Adversary with Windows Event Log Monitoring*. Revision 2. [Online]. Available: https://www.iad.gov/iad/customcf /openAttachment.cfm?FilePath=/iad/library/ia-guidance/security-configuration/applications/assets/public/upload/Spotting-the-Adversary-with-Windows-Event-Log-Monitoring.pdf&WpKes= aF6woL7fQp3dJirMdSa8DC96UXYyqLaHTWS7vR.

[18] J. Creasey. (2015). Cyber security monitoring and logging guide. CREST (GB). [Online]. Available: https://www.crest-approved.org/wp-content/uploads/2015/ 05/Cyber-Security-Monitoring-Guide.pdf.

[19] T. Limmer and F. Dressler, *Survey of Event Correlation Techniques for Attack Detection in Early Warning Systems*. Rep. 01/08. Computer Networks and Communications Systems, University of Erlangen, Germany.

[20] T. Kondo and G. Kanyenze. (2011). Beyond the enclave. Available: http://lib.myilibrary.com?ID=323828.

[21] S. Campbell and E. Sebring, "Enhancing CANES SIEM performance via optimized event logging" Capstone Project Report, Dept. Computer Science, Naval Postgraduate School, Monterey, CA, 2016.

[22] A. Spadaro, "Event correlation for detecting, advanced multi-stage and cyber-attacks," M.S. thesis, Dept. Information and Communication Technology, Delft University of Technology, 2013.

[23] O Catakoglu, M. Balduzzi, and D. Balzarotti, "Automatic extraction of indicators of compromise for web applications," *International World Wide Web Conference Committee (IW3C2)*, Montréal, Québec, Canada, 2016, pp. 1–11.

[24] T. Proffitt and C. Roberstson. (2013, Feb.). Indicators of compromise in memory forensics. SANS Institute InfoSec Reading Room. [Online]. Available: https://www.sans.org/reading-room/whitepapers/forensics/indicators-compromise-memory-forensics-34162.

[25] J. Andress. (2015, May). Working with indicators of compromise. *ISSA Journal*. [Online]. Available: www.issa.org.

[26] M. Cloppert. (2009). Security Intelligence: Attacking the Cyber Kill Chain. SANS Digital Forensics and Incident Response Blog. Available: /blog/2009/10/14/ security-intelligence-attacking- the-kill-chain#.

[27] R. Dias. (2014, Oct. 24). Intelligence-driven incident response with YARA. SANS Institute InfoSec Reading Room. [Online]. Available: https://www.sans.org/reading-room/whitepapers/forensics/intelligence-driven-incident-response-yara-35542.

[28] V. M. Alvarez, Yara Documentation, 3.5.0. YARA, 2016.

[29] S. Kadhi, *Fighting Back Malware with IOC & YARA*. Paris, France: OSSIR, 2012.

[30] Sophisticated indicators for the modern threat landscape: An introduction to OpenIOC. (n.d.). [Online]. Available: www.openioc.org.

[31] Cyber observable eXpression – CybOX, a structured language for cyber observables. (n.d.). [Online]. Available: https://cybox.mitre.org.

[32] S. Barnum and R. Struse, *IT Security Automation Conference. A (Very) Brief Introduction to the Cyber Observables eXpression*. CybOX. Crystal City, VA: 2011.

[33] CybOX language frequently asked questions (FAQs). (2016). The MITRE Corportion. [Online]. Available: https://cybox.mitre.org.

[34] CybOX. (2016). How to create a CybOX object. The MITRE Corporation. [Online]. Available: https://github.com/CybOXProject/schemas/blob/master/ cybox_common.xsd.

[35] Bc. P. Nemcek, "Analysis of Malware Classification Schemas," M.S. Thesis, Department of Computer Science, Masarykova Univerzita Fakulta Informatiky, 2014.

[36] M. Toussain. (2014, Sep. 25). Home-field advantage using indicators of compromise to hunt down the advanced persistent threat, SANS Institute InfoSec Reading Room. [Online]. Available: https://www.sans.org/reading-room/whitepapers/detection/home-field-advantage-indicators-compromise-hunt-down-advanced-persistent-threat-35462.

[37] C. Harrington, “Sharing indicators of compromise: an overview of standards and formats,” presented at RSACONFERENCE2013, February 27, 2013, DSP-W25A.

[38] T. Haruyama. (n.d.). Volatile IOCs for fast incident response. Internet Initiative Japan Inc. [Online]. Available: https://digital-forensics.sans.org/summit-archives/DFIR_Summit/Volatile-IOCs-for-Fast-Incident-Response-Haruyama.pdf.

[39] S. Maccaglia and J. Myers, *RSA Incident Response Report: Threat Detection Techniques – ATM Malware*. St. Paul, MN: EMC Corporation, 2016.

[40] NetIQ, *NetIQ Sentinel User Guide 7.3.4*. Houston, TX: Novell, 2015.

[41] Ontology (information science). (n.d.). *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Ontology_(information_science)#Examples_of_applications. Accessed Jan. 3, 2017.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California