Chapter 4
Perception of Cyber Threats

Kevin Kornegay, Kofi Nyarko, Jeffrey S. Chavis, and Ahmad Ridley

1 Introduction

An important aspect of cyber threat perception is reducing the uncertainty level represented by the large volumes of cyber events collected from host-based and network-based sensors (Shakut et al., 2020). By automating the collection, filtering, and aggregation of these events in real-time, threat perception can be improved. In addition, cyber alerts generated from such context can enable the prioritization of threat alerts and, ultimately, efficient and effective responses to threats. Autonomous Intelligent Cyber-Defense Agents (AICA) can identify and prioritize cyber threats faster, and in an increasing number of scenarios, better than human cyber defenders, motivating their inclusion in the cyber threat analysis process (Muser & Garriott, 2021).

K. Kornegay (✉)
Cybersecurity Assurance & Policy (CAP) Center, Morgan State University, Baltimore, MD, USA
e-mail: kevin.kornegay@morgan.edu

K. Nyarko
Electrical and Computer Engineering, Morgan State University, Baltimore, MD, USA
e-mail: kofi.nyarko@morgan.edu

J. S. Chavis
Asymmetric Operations Sector, The Johns Hopkins University Applied Physics Laboratory, Laurel, MD, USA
e-mail: Jeffrey.Chavis@jhuapl.edu

A. Ridley
Laboratory for Advanced Cybersecurity Research, National Security Agency (NSA), Laurel, MD, USA
e-mail: adridle@uwe.nsa.gov

A. Kott (ed.), *Autonomous Intelligent Cyber Defense Agent (AICA)*, Advances in Information Security 87, https://doi.org/10.1007/978-3-031-29269-9_4

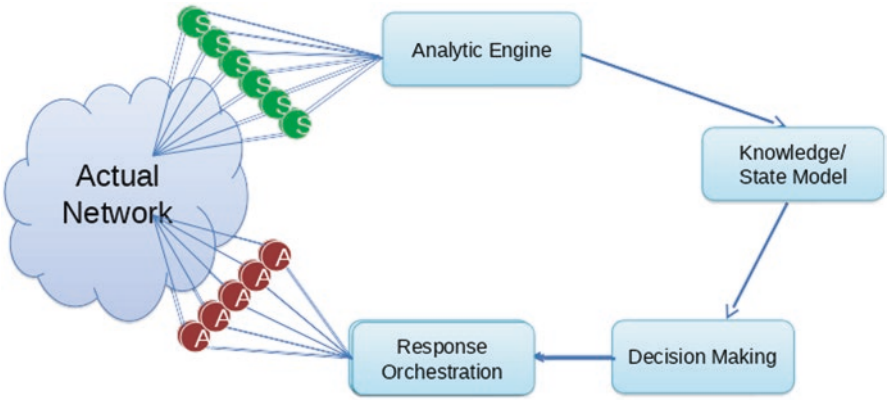

Fig. 4.1 Cyber-defense monitoring and decision-making feedback loop

Cyber threats can vary depending on the type of network being monitored. Figure 4.1 below provides an abstract view of a closed-loop monitoring and decision-making system. The sensors, S, collect cyber data from hosts and network devices. The data is processed by fixed, rules-based, or machine learning-based analytics, producing results such as cyber alerts about potentially compromised network devices. The analytic engine results are aggregated to create a current overall network state, which provides a context about the security or availability of the network. Based on this state of information, AICA decide how to respond to potential threats. Any response is implemented on the network using actuators, A. The sensors continue collecting data, initiating the next phase of the monitoring loop, providing feedback to the agents about the impact of their decisions.

The type of threats against an Internet-connected, open network will differ from a closed network with no internet access. Agents monitoring these open networks must continuously detect threats originating from internal and external sources, while agents monitoring closed networks are probably more concerned about threats originating from internal sources. Another example of cyber threat diversity involves the homogeneity of hardware and software on a network. The cyber threats to the homogenous enterprise networks of Windows workstations and servers vary from threats to an industrial control system (ICS) network. Furthermore, the spectrum of threats encountered by an agent monitoring a heterogeneous enterprise network containing a combination of Internet of Things (IoT) devices, Windows- and Linux-based workstations, servers, routers, and iOS- and Android-based mobile devices can be extremely large, and complex. Nguyen and Reddi (2021) provide specific examples of these types of network environments that can be monitored by AI-based agents. AICA have the ability to either quickly detect threats or adapt to various threats encountered across these diverse network environments.

One specific type of automated agent is a defensive cyber deception agent. A decoy environment consists of realistic, lightweight decoy agents that appear to be real systems running real services from an attacker's perspective performing actions,

such as IP address scanning, on the network (Walter et al., 2021). These decoy agents are deployed on a real network alongside real systems to maximize the probability of detecting and quickly mitigating threats from cyber-attackers. A high number of false systems helps provide an asymmetric advantage for cyber defenders by distracting an attacker from the real assets. This forces an attacker to take additional actions, increasing the likelihood of revealing themselves and the defender detecting the threat in its early attack stage. Leveraging deceptive AICA for improved threat perception can be effective in reducing the inherent cyber-attacker advantage (Muser & Garriott, 2021).

Since authorized network users usually do not interact with decoys, these agents can provide an early warning, high-confidence threat signal to defenders about an attacker's presence (Walter et al., 2021). In contrast to light-weight baits, honeypots are a different type of high-fidelity deception system. These high-interaction fake systems are connected to but located outside the real network. Once an attacker enters a honeypot, defenders can gain insight about cyber-attack threats, such as goals and severity level, through further attacker interactions with the deceptive honeypot agents. Ferguson-Walter et al. (2019) describe how the impact of cyber deception can be extended further, leading an attacker towards a specific incorrect belief.

However, there exists a full spectrum of AICA, from automated (i.e., static, expert-driven rules) agents to autonomous (i.e., adaptive, Artificial Intelligence (AI)-based rules) agents, which can be used to improve cyber threat perception. Although automated agents created by human expert-based rules and logic can be beneficial, autonomous, artificial intelligence (AI)-based agents evolve. Muser and Garriott (2021) describe the potential short- and long-term benefits of AICA, and how AICA can adapt to detect changing cyber threats. For example, as network administrators add new machines to a network, new hardware and software attack surfaces are introduced. The time required for administrators to manually update the security rules/policies that guide automated agents would potentially leave the network vulnerable to the speed and scale of new and existing cyber threats. AI-based agents that efficiently learn to defend the new attack surface autonomously can mitigate this speed and scale challenge better than automated agents.

AICA have been employed to defend against cyber-attacks. Such agents have been trained using supervised and unsupervised methods to perform automated and autonomous cyber-defense tasks, such as intrusion detection, malware detection, and data privacy protection (Shakut et al., 2020). Recently, reinforcement learning (RL) has been increasingly used to autonomously detect (and respond) to cyber-attacks (Nguyen & Reddi, 2021). Unlike other ML methods, like supervised learning from labeled input-output examples, an RL-based agent learns its behavior from interacting directly with the environment. It is a trial and error approach that attempts to imitate the basic manner in which humans learn. Given a state of the environment and a reward signal indicating how good or bad an action is, the agent learns a sequence of good actions to achieve a goal (Shakut et al., 2020). For cyber threat perception, an RL agent can learn from which host or network device to gather additional data to reduce the uncertainty of cyber threats. These RL-based AICA

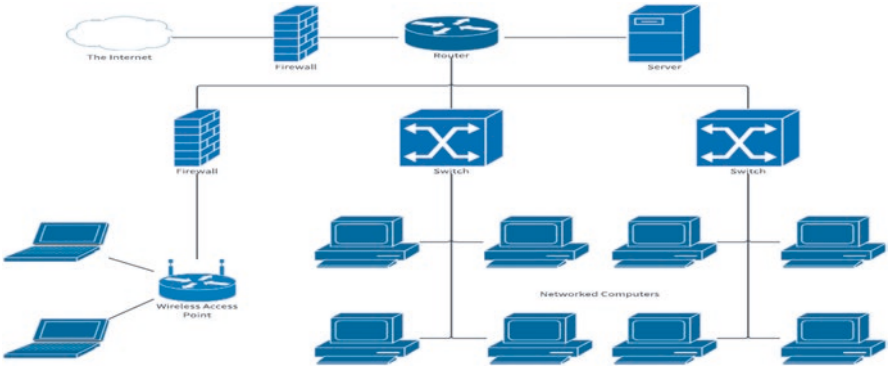

Fig. 4.2 Sample enterprise computer network

can also be leveraged to adapt the capabilities of decoy agents to maintain the effects of cyber-deception (Ferguson-Walter et al., 2019). Both ML- and RL-based AICA provide increased ability to detect threats hidden among large amounts of cyber event data.

Different AICA agent hierarchies can be implemented for improved cyber threat perception. For example, a single agent can be effective in monitoring a small, homogenous enterprise network. The single agent would collect cyber data from each devices or aggregated data across each device to perform malware detection. For larger, heterogenous network monitoring, this agent structure would be inefficient. A decentralized hierarchy of multiple agents is more practical and generalizes across multiple cyber environments, such as Internet of Things, Cloud Computing, and 5G Networks (Nguyen & Reddi, 2021).

Individual agents located on each network device, e.g., the switches, routers, and computers shown in Fig. 4.2, provide local monitoring and report their results to higher-level agents performing global monitoring. The higher-level agents use these results to monitor different network segments, e.g., the wireless and wired segments. Teaming and cooperation among agents can provide additional benefit in reducing uncertainty about the nature of a cyber threat.

Finally, in the remaining sections, we will discuss the potential impact of AICAs in a hierarchical, decentralized agent structure in perceiving complex cyber-attacks within various, dynamic cyber environments.

2 Simplified Hierarchical Cyber-Defense Agents for Threat Perception

As discussed in previous chapters, in general terms, a software agent can be defined as a software entity that functions continuously and autonomously in a particular environment and can carry out activities flexibly and intelligently that are

responsive to changes in the environment (Bradshaw, 1997). Ideally, an agent that functions continuously would be able to learn from its experience and inhabit an environment with other agents and processes collaboratively and cooperatively, moving from place to place as needed (Bradshaw, 1997). In this chapter, the hierarchical agent architecture (Palau et al., 2019) is further explored for the purpose of conceptualizing the implementation of threat perception in AICA. More specifically, cyber-defense agent may be considered as software processes that perform specific monitoring and offensive and defensive functions via individualized configurations that may be duplicated or migrated across multiple operating environments. Hence, these agents are autonomous because they are independently-running entities, individuated by their configuration profiles that govern how they sense, adapt, and affect their local environment. Due to the agent's independent nature, they can be added, removed, and reconfigured without altering other components of the operating environment.

In general, a Cyber-defense agent system should provide the following characteristics: (1) continuous operation, (2) fault tolerance, (3) ability to resist subversion, (4) minimal overhead, (5) dynamic reconfigurability, (6) adaptability, (7) scalability and (8) graceful degradation of service (Spafford & Zamboni, 2000). Regarding continuous operation, a collection of agents may form a group that performs simple or complex coordinated functions that the individual agent can not achieve. The collective agent system can be designed to run continuously if some agents are taken off-line, purposely or through malicious intent, thereby providing continuous cyber-defense functionality.

When agents are deployed hierarchically, they can capture higher-level system states and be able to adapt to changes in global behavior. This hierarchical structure enables agents to be inherently scalable. One bottleneck that agents deployed in this fashion may face lies in the agents' communication mechanism. But there are various methods of circumventing these bottlenecks by minimizing communication between components (Cen et al., 2014). If the service for one or more agents is disabled, the damage is restricted to just those sets, and perhaps those directly depend on their service. Thus, if the agents are correctly organized in mutually independent sets, service degradation will be gradually proportional to the number of agents that stop functioning (Spafford & Zamboni, 2000). The ability to start and stop agents independently enables the possibility of reconfiguring dynamically. This, in turn, allows other agents or processes to migrate agents by overwriting current configurations with configurations from other agents that have demonstrated improved effectiveness at a task in a given environment. Because an agent can be reconfigured arbitrarily, it can obtain its data from an audit trail, probing the system it is running, capturing packets from the network, or capturing data through physical sensors. Thus, cyber-defense activities can be supported across traditional boundaries between the physical system, the operating system host, and networks. Furthermore, since agents are implemented as separate processes on a host, each agent can be implemented in the programming language best suited for the task and the host (e.g., light-weight drone vs. enterprise system).

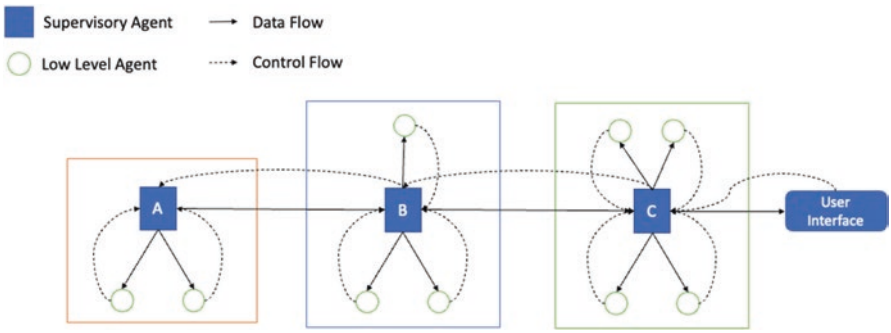

Fig. 4.3 Simplified hierarchical agents

As illustrated in Fig. 4.3, a simplified agent system architecture could consist of generic agent profiles that can be adapted for various functionalities based on their configuration. Besides specific task-based functionalities, agents can be configured in one of 2 primary types: low-level and supervisory. Low-Level agents are distributed over any number of hosts, where they either monitor for specific events or perform task-specific actions. Each agent can persist in its state for a certain period of time defined by its configuration, which enables the detection of long-term attacks. Each low-level agent is configured to monitor for one or more events and report detected events to a supervisory agent, where one such agent exists per host. The supervisory agent oversees all operations of the low-level agents on that host. These agents can start/stop low-level agents and send new configurations as needed. Supervisory agents exist in a hierarchy in which each one may communicate with several parent supervisory agents, where each one monitors and controls several child supervisory agents. This architecture provides redundancy and resistance to the failure of one or more supervisory agents. These agents have access to network-wide data and can thus perform higher-level correlations and detections across several hosts. By combining reports from multiple agents, they can build a unified picture of the status of their host. Supervisory agents at the highest level of the architecture will employ capabilities to interface with users; this may be through a graphical user interface, terminal commands, or physical input/output interfaces on embedded systems. There are several methods by which agents can communicate securely and in a distributed manner, such as through asymmetric encryption over TLS/SSL with publish/subscribe (pub/sub) messaging (Farmer et al., 1996).

Low-level agents consume host/network data via another kind of process called filters (Spafford & Zamboni, 2000). These filter processors are responsible for acquiring specific host/network data types and feeding the filtered data to one or more agents. One efficient process by which agents could receive these data streams is through a pub/sub messaging implementation. A low-level agent would generate a notification when an event is detected on the subscribed data provided by one or more filters based on its current configuration. The agent doesn't have the authority to trigger an alarm or action directly. Hence it sends its event to one or more

supervisory agents on the same parent hierarchical level. These agents combine events received across child agents and communicate across sibling supervisory agents to determine the appropriate course of action. Depending on the deployed system architecture, the right course of action may be to generate alerts sent further up the hierarchy to parent supervisory agents or to directly trigger an alarm or action event. Action events are sent across or down the hierarchy to agents capable of executing the actions, where the activities may change the host's communication, state information, or physical posture.

Agents may evolve over time based on their function and information obtained from their environment. For example, suppose the agent utilizes reinforcement learning to detect malicious activity on its local host. In that case, it may evolve to better detect behavior based on communication with, and feedback from, supervisory hosts. This evolution would be captured in its configuration parameters, such as current learned weights of its neural network. If the agent successfully detects desired events, supervisory agents may clone its configuration to other agents on other hosts.

3 Autonomous Hierarchical Agents for Anomaly Detection

Anomaly detection in a complex system of systems can be performed on two primary levels, communication network and application, including environment and system state sensing. These systems often have some aspect of mobility where network nodes wander freely and can join and leave a given network arbitrarily. These network dynamics impose further complications on effective anomaly detection. With traditional centralized solutions, the scale of these types of networks would be an issue since anomaly detection solutions would have to factor in load-balancing and fault tolerance. However, an agent architecture inherently addresses these factors with intelligent autonomous agents.

In the AICA architecture, low-level agents disbursed across fixed and mobile network nodes detect anomalies by analyzing the events on the systems where a data instance designates each event. The data instance possesses defining features (i.e., attributes) (Xie et al., 2011), such as a packet's source/destination address, length, and time at which it was sent for the case of network-level anomaly detection. Features are crucial for distinguishing normal behavior from anomalous behavior. A given communication network typically provides many features per a given data instance, yet they are not necessarily all equally informative (Bhuyan et al., 2014). Low-level agents can be configured independently to observe varying features based on their location within the network and on the systems in which they reside. For example, some agents can be configured to use information-theoretic approaches to help distinguish informative features (Cen et al., 2014; Ham & Choi, 2013; Mas'ud et al., 2014). In this approach, Information Gain (IG) (Mitchell, 1997) and chi-squared (Sharma, 2005) methods can be utilized to select the most informative features.

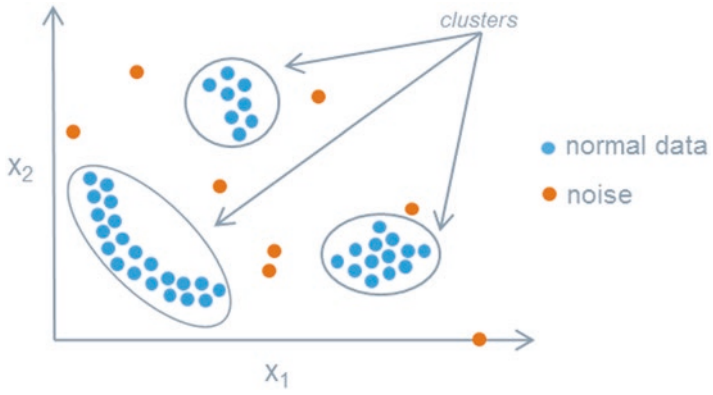

Fig. 4.4 Example of a point anomaly

Similarly, other agents could be configured to choose features and perform anomaly detection based on the Fisher score, the ratio of inter-class variance to the intra-class variance for a given feature, assuming that normal instances form a class and anomalous instances form another class (Crowley et al., 2003). Agents can also be configured to perform a similar method through machine learning (Guyonand & Elisseeff, 2003). In the simplest incarnation, agents can be configured based on hand-pick informative features to help detect anomalies (Al Marakeby et al., 2010).

Low-level agents can detect anomalies by observing data instances (i.e., point anomalies). However, point anomalies do not fit the situations where anomalous behavior is an aggregate of data instances, or when anomalies are associated with given contexts, as shown in Fig. 4.4. In some cases, the anomalous behavior is defined within a context; thus, a given data instance is not anomalous unless it happens within a predefined context (Chandola et al., 2009). In this case, Low-level agents would not conclusively detect anomalies since they may lack the broader context of the state of relevant parts of the network or system. Hence, when the right conditions are met, they send alerts to the supervisory agents that consider all incoming alerts to facilitate the detection of anomalous contextual activity. In contextual anomalies, the data instance has to have some features about the context, whether temporal (i.e., time-relevant), spatial (i.e., location-relevant), or a different kind of context per the problem domain, as shown in Fig. 4.5.

Agents can be configured to detect anomalies through methods that are either signature-based (Migliardi & Merlo, 2013) or behavioral (Bhuyan et al., 2014). Signature-based solutions operate by applying a set of hardwired patterns, signatures, or rules against given behavior(s). An anomaly is detected if a given behavior matches either one of the hardwired signatures. Otherwise, the agent will not tell whether or not the designated behavior is anomalous. While these agents are potentially more efficient in terms of computational cost, which is a practical consideration when deployed on power-constrained platforms, these solutions fail to identify new or previously unseen anomalies.

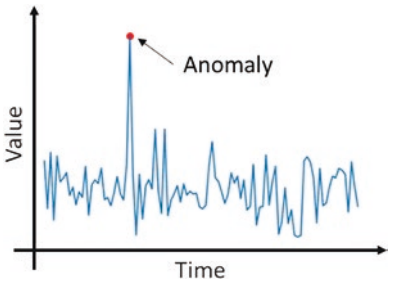

Fig. 4.5 Example of contextual anomaly

On the other hand, behavioral agents can learn the normal and/or anomalous behavior(s) of a network and, thus, have the potential to identify whether new or previously unseen behavioral patterns are anomalous (Mitchell, 1997). These agents are typically supervisory and may experience longer computational times and utilize more computational resources. Hence, its best for these agents to be deployed on near-edge platforms or systems.

Individuated agent configurations enable hybrid methods to detect anomalies and activities across an entire network. The following are some common approaches that an agent can be configured to utilize:

The spectral approach: In some situations, the dimensions of the data instances (i.e., features) are inherently dependent. Thus, combining the dependent dimensions both improves the classification accuracy and reduces the computational complexity; the application of such a combination transforms the original data instances into new instances with only the independent dimensions, formally referred to as dimensionality reduction (Wang, 2012). One popular dimensionality reduction technique is the Principal Component Analysis (PCA) algorithm which gets applied to a matrix of the original instances and generates a set of orthogonal vectors. The first k vectors capture the highest variance and designate normal activity, while the last m vectors represent anomalous activity. Hence if a data instance is projected into the anomalous subspaces, it can be considered anomalous (Chandola et al., 2009)

The Information-Theoretic Approach: Data instances can be a set of symbols generated by the network or system, whereas each instance is generated independently with a certain probability. Thus, one would seek to measure the average amount of information conveyed by each instance. This approach utilizes the concept of entropy that assumes anomalies distort the information content of the network's data instances. Hence, the anomaly detection technique needs to split the data instances into subsets that minimize the entropy (Cen et al., 2014; Ham & Choi, 2013; Shabtai et al., 2012; Cuadra-Sanchez et al., 2014)

The Machine Learning Approach: Machine learning (ML) agents improve their ability to distinguish normal behavior from anomalous behavior with experience (Mitchell, 1997). These agents typically provide a mapping that adapts to unseen network anomalies (Wang, 2012) by utilizing a set of data instances that resemble the instances within a given system network; this set is referred to as a training

dataset. Supervised ML algorithms learn the mapping function by utilizing labeled training sets. On the other hand, Unsupervised ML algorithms utilize training sets of totally unlabeled instances. The two approaches are mixed into the semi-supervised learning hybrid approach in some cases. The algorithm is trained with most unlabeled instances and a minority of labeled instances. A machine learning algorithm starts by learning the mapping function from the training dataset, then proceeds to the testing phase. It examines "other" data instances collectively referred to as the testing set and computes the label for each instance using the mapping function learned. Once trained, an agent's configuration would include the desired ML architecture and associated trained weight vectors. ML algorithms can be further categorized into: (1) Classification-based, (2) Nearest-neighbor algorithms, (3) clustering.

The main goal of classification-based ML algorithms is to assign each data instance to either one of pre-set classes based on their features. Some examples include:

Classification-oriented neural networks: A neural network loosely mimics the human neuronal structure and comprises a set of highly interconnected processes that operate asynchronously on their local data (Chandola et al., 2009). A neural network is trained on normal data instances. After that, it is presented with unseen cases. Here, the network applies a test on the test data instance; it gets accepted as a typical instance if it passes. Otherwise, it is considered anomalous. Feed-forward networks are neural networks typically used in classification, like multilayer perceptron networks (Cuadra-Sanchez et al., 2014). Depending on the labeling of the data, neural networks can be used for both supervised and unsupervised learning.

Bayesian networks: A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest (Thottan et al., 2010). Bayesian networks are supervised learning algorithms based on the well-known Bayes Theory (Mitchell, 1997). They operate by estimating the posterior probability of an event given some pre-condition. A particular class of Bayesian networks is referred to as Naïve Bayesian networks used for univariate categorical data instances (Mitchell, 1997). Here, for a given data instance, the network estimates the posterior probability of detecting a class label from a set of normal and anomalous class labels. The class label with the most considerable posterior probability is selected as the class to which the data instance belongs. Multivariate data instances are handled via generalizing the univariate model, as the posterior probability for each attribute is estimated. The estimated probabilities get combined to assign the data instance to a given class (Chandola et al., 2009)

Support Vector Machines (SVM): SVMs are supervised learning algorithms that represent the training data instances in a multi-dimensional plane and then determine a hyperplane that splits the data instances into two disjoint groups while maintaining the maximum margins around the separating hyperplane (Nigrin, 1993). One-class SVM algorithms are trained only with normal data. Thus, upon receiving a test data instance, they predict whether it belongs to the normal data class or not. SVMs are well-defined as they stem from a

solid mathematical background in statistical learning theory (Nigrin, 1993). An SVM algorithm is considered a linear classifier when it uses a line to split the data instances into normal and anomalous. To perform non-linear classification, SVM algorithms use kernel functions (Gardner & Dorling, 1998).

Rule-based machine learning algorithms: These supervised learning algorithms learn the rules that capture the expected behavior of a data instance. Thus, it is considered anomalous when all the rules fail to capture a data instance during testing. Decision trees and Association Rule Mining (ARM) techniques, among other rule-based methods, are used to learn the rules from the training data instances (Crosbie & Spafford 1995; Hofmeyr 1999). Each rule is assigned a weight proportional to the ratio of the number of training data instances the rule classified correctly to the total number of training instances covered by the rule. The rule that best captures the test instance for a given test data instance is sought. Here, the anomaly score is the inverse of the weight associated with the best rule. Random forests are constructed from several decision trees; a random forest reports the mode of classifying all individual decision trees as the overall classification result (Heckerman, 2008).

Nearest-neighbor algorithms use distance-based or density-based functions to measure the distance between a given data instance and its nearest neighbor (Chandola et al., 2009) This distance designates the anomalous score of that instance. The assumption is that normal instances occur in dense groups, unlike anomalous instances. These algorithms can operate in supervised or unsupervised fashions based on whether labels are used in training data instances.

Clustering algorithms are unsupervised learning algorithms that operate by trying to identify groups (i.e., clusters) of closely located (or similar) training data instances. Anomalies may form sparse clusters or belong to no cluster at all. Self-Organizing Maps (SOM) (Karnin et al., 2012), Expectation-Maximization (EM) (Kecman & Brooks, 2010), k-means clustering (Elbasiony et al., 2013), and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithms (Kohonen, 1990) are classical clustering algorithms.

Based on the anomaly detection techniques discussed, agents can be imbued with the ability to perform network-level and application/system level detection effectively. For network-level detection, low-level agents analyze the headers and/or the payloads of the messages exchanged in the network and send filtered alerts to supervisory agents for correlation and detection. For application/system-level detections, low-level agents analyze the application or system specification and/or examines its behavior during runtime and then report significant events to supervisory agents for further action. Generally speaking, agents can tackle anomaly detection using several techniques belonging to different disciplines. One approach to anomaly detection uses either parametrized or non-parametrized statistical methods to model the network and/or devices behavior and measure the deviation of anomalous behaviors from normal ones. Alternatively, machine learning techniques can be used to learn the normal and/or abnormal behaviors and then try to classify or cluster unseen behaviors accordingly. In addition, information-theoretic and spectral techniques provide different perspectives to help measure how an anomalous behavior differs from normal behaviors.

4 Honeypot Agents

Another class of general agents is one where each agent is designated to collect crucial, accurate, concise high-quality information about malicious activities (Yegneswaran et al., 2005). These agents facilitate the configuration of computer resources to serve as honeypots whose value lies in being probed, attacked, or compromised (Ester et al., 1996). With the help of these resources, agents can spot zero-day attacks and give insights into attackers' actions and motivation. Supervisory agents can receive log data from honeypot agents for analysis from the systems on which they are deployed. The general objective of a computer resource configured as a Honeypot is either to distract attackers from their actual target or to gather information about the attackers and attack patterns, such as a set of popular target hosts and the frequency of requests-responses (Yegneswaran et al., 2005). For the AICA architecture, the primary purpose of honeypot agents is to configure systems to gather information about the attackers. Once a system has been configured and deployed, these agents are responsible for actively monitoring log data (e.g., applications, services, design) for known text patterns or anomalous patterns for essential events. When these events are detected, honeypot agents send alerts to supervisory agents for further analysis, resulting in alerts sent up the agent hierarchy or actions sent down the order to appropriate agents for execution.

While the AICA architecture includes general agents configured to detect malicious activities through various techniques, this task is often complicated on production systems because the attacks are submerged in vast amounts of production or mission-critical activity. Honeypot agents can simplify the detection process since the systems they configure have no production activity, and thus all connections to the honeypot system are suspect by nature. Therefore, unauthorized probes, scans, or attacks are easily detected with fewer false positives and negatives (Yegneswaran et al., 2005).

Reaction to attacks can be accelerated with the help of honeypots. Because attack data is not mingled with production activity data, supervisory agents' analysis of potential attacks is greatly simplified. In addition, honeypot systems can be taken offline entirely for further forensic examination. Insights from this analysis can be used to reconfigure the honeypot agents for increased effectiveness and help supervisory agents develop appropriate countermeasures against threats.

For example, an organization that deploys the AICA architecture can redirect incoming traffic to unused IP addresses to a virtual machine (VM) configured as an SSH honeypot and spun up by a honeypot agent. Agents on the honeypot VM identifies the attackers by IP addresses and then send this information to supervisory agents that develop filters to block the access to mission-critical systems. The functionality of the honeypot agents on the VM can be limited, as it only has to recognize the traffic and its source. A more complex honeypot agent configuration might help determine which dictionaries were used to guess the passwords. This information would be sent to a supervisory agent, which would then use the information to update a rule relating to password strength, which would then be sent to the human

analyst as an alert. This type of agent configuration would require the analysis of extensive log records with SSH credentials.

In general, honeypot agent configurations may be classified in terms of the level of interaction (Leita et al., 2008) or the direction of interaction (Spitzner, 2003). In terms of the level of interaction, the honeypot agents consider low-interaction, medium-interaction, or high-interaction configurations. Low-interaction configurations simulate only a small set of services, such as SSH or FTP, and do not allow the attacker to access the operating system. These configurations would be suitable for recognizing peaks in the number of requests. Medium/High-interaction configurations provide more simulated services with increasing sophistication that offer higher levels of attacker interactions but may still limit access to the operating system. These systems would produce reasonable replies to attackers in the hope of triggering follow-up attacks. The difference between medium and high levels of interactions is based on the levels of risk of compromise, information levels, and level of access to the operating system. For honeypot agents configured based on the direction of interaction, they fall primarily into server or client-based configurations. Server-based honeypot configurations are entirely passive; therefore, all incoming requests form an anomaly and are, by definition, an attack. Client-based honeypot configurations actively search and contact communication partners. Thus, client honeypots must discern which communications comprise an anomaly. Heuristics usually verify this by looking after uncommon modifications.

In summary, honeypot agents enable data collection that is not polluted with noise from production activities and is usually of high value. This makes the data sets they process smaller and less complex, which reduces their workload and, by extension, the supervisory agents to which they communicate their findings. Furthermore, honeypot agents deployed on configured VMs only need to process traffic directed at them or originates from them. This means that they are independent of the workload of their parent process. Additionally, these agents capture everything used against them, which means unknown strategies and zero-day-exploits will be identified. It should be noted that any activity with server-honeypot configurations is an anomaly, which should be considered an attack. On the other hand, client-honeypot configurations verify attacks by detecting system state changes, reducing false positives and false negatives (Yegneswaran et al., 2005).

5 Perception of Threat Applications

One approach to applying automation to the cybersecurity problem is Integrated Adaptive Cyber Defense (IACD). IACD is a research effort jointly funded by the US Department of Homeland Security (DHS) and the US National Security Agency (NSA), in collaboration with The Johns Hopkins University Applied Physics Lab (JHU/APL) and industry. Integrated Adaptive Cyber Defense (IACD) aims to shorten the timeline and effectiveness of cyber defense via integration, automation, orchestration, and sharing of machine-readable cyber threat information. IACD

defines a strategy and framework to adopt an extensible, adaptive, commercial off-the-shelf (COTS)-based approach (IACD, 2016).

Since 2014, IACD has been a jointly sponsored government, industry, and Trusted agent (JHU/APL) initiative. IACD is an effort to get humans from 'in the loop' to 'on the loop' (Sparrell, 2019). Human-in-the-loop aspects of cybersecurity include disgruntled employees, human errors, awareness and training, access controls and certifications. Human-on-the-loop, deals with the lack of Situation Awareness (SA) or a Common Operating Picture (COP), increased cognitive load and stress that contribute to lower attention span, and the difference in speed between technology and human cognition processes (Sundararajan et al., 2018).

Automated Cyber systems like IACD seek to create an ecosystem to alter the timeline and efficacy of cyber defense through integration, automation, and information sharing. IACD seeks to decouple functions and standardize interfaces between functions to and defines the following security functions:

- Sensing: gathering all the data
- Sense-making: correlating and analyzing data, transforming it into information, knowledge, and intelligence
- Decision-making: deciding what to do
- Acting: sending the actual commands.
- Socializing: Sharing threat data among interested, trusted parties.

One of the more prevalent forms of attacks that are particularly suited for Automated cyber defense is effects-based courses of action. Effects Based Operations (EBO) are "actions taken against enemy systems, designed to achieve specific effects that contribute directly to the desired military and political objectives" (Caroli et al., 2004).

Effects-Based Courses of Action Cybersecurity attacks increase volume, scale, and complexity. To address the growing threats, cybersecurity solutions are also becoming more complex. To help manage this complexity, Security Orchestration, Automation, and Response (SOAR) technology can be used to coordinate the actions of multiple security tools. SOAR technology seeks to create a need to ensure that the correct information is exchanged between products to provide the necessary context to achieve a coordinated response. SOAR platforms enable the Observe and Act functions of cyber defense required to Observe, Orient, Decide, & Act, more commonly known as the OODA loop, for decision-making and operations.

Now, many security vendors are adding artificial intelligence (AI) and/or machine learning (ML) capabilities to their products, which could be used to address and improve decision-making functions for cyber security. This division of labor between AI/ML solutions and SOAR platforms could help manage cybersecurity solutions' complexity, speed, and scale: AI/ML solutions can be applied to find patterns and decide faster and at scale. In contrast, SOAR can be applied to act faster and at scale. Integrated Adaptive Cyber Defense (IACD) demonstrated how to bridge these technologies while maintaining human control using effects-based courses of action (COAs).

An effects-based COA is a set of response actions to a cyber-attack, selected based on the desired high-level cyber affect – the goals of the response – rather than having to specify the exact steps to be executed via a course of action. In a traditional COA workflow, a SOAR platform starts the workflow, gathers additional data/evidence, selects an appropriate COA, and performs its execution – covering all the functions of the OODA loop within that single platform. With an effects-based COA workflow, an AI capability can be used to gather additional data/evidence and select an appropriate COA based on that data and the desired cyber effect. Then the SOAR platform can be used to automate and orchestrate the actions required of various security products to achieve that desired response and outcome.

6 Experimentation

IACD conducted an experiment to demonstrate the benefits of combining AI and SOAR technologies using effect-based COAs. The experiment used the DarkLight AI expert system to provide sense-making and decision-making capabilities, corresponding to the Orient and Decide functions of the OODA loop. IACD used the Cortex XSOAR platform to control response actions, corresponding to the Act function. The figure below depicts the workflow for an effects-based COA, where DarkLight performed the first few decision-making functions, and Cortex XSOAR performed the remaining response actions.

In the experiment, The automated system successfully demonstrated the combination of DarkLight AI and the Cortex XSOAR platforms to select and execute effects based COAs in the face of different attack scenarios, all with a human monitoring "on the loop" instead of a human having to decide and act "in the loop." DarkLight made sense of two different attack scenarios – malware conducting data exfiltration versus ransomware – and selected an appropriate effects-based COA response for the attack. DarkLight then triggered Cortex XSOAR to execute the proper COA for the attack, and Cortex XSOAR orchestrated the response actions of the enterprise security products.

Throughout the process, a human monitored the performance of both the AI and SOAR components via metrics and summaries of the actions taken. The human had specific criteria defined for situations where he/she would take over control. The human was available for decision escalation in cases where the AI could not decide with a certain threshold level of confidence. The effects-based COA experiment successfully demonstrated the ability to coordinate the activities of an AI/ML product and a SOAR platform, allowing each to perform functions of the OODA loop to which each is best suited while enabling human monitoring and control.

Leveraging automation in IT systems has been shown to provide measurable improvements in cyber security. Several organizations have used automation with their systems, as shown below:

- JHU/APL did studies on their network comparing various automation scenarios with their current manual scenarios. The most significant finding was that the attacks were stopped two orders of magnitude faster, resulting in significantly less damage (Peters, 2017).
  - Phantom Cyber, a security orchestration vendor, published similar savings in combating phishing. Their customer reduced phishing incident response costs by 98% and saved $1.06 M annually (Royer, 2016)
- Zepko, a managed security service provider in the United Kingdom, used OpenC2 to increase the efficacy of their Security Operations Centre (SOC) by 25–30% (Bradbury, 2016).

Automation solutions like IACD are not a cure-all for solving cybersecurity challenges, but they provide a mechanism to respond to the threat at the speed of the threat and not at human speed. Moving forward, much work is needed to evolve automation systems to recognize, react and respond to threats as they evolve and to deploy solutions to systems in a timely and effective fashion.

7 In Conclusion: Further Research Areas

This chapter aimed to discuss the potential benefits of applying AICA to improve the perception of cyber threats for effective mitigation of cyber-attacks. Research on applications of different types of AI/ML algorithms and architectures has been performed to evaluate AICA. As mentioned in other chapters, the practical use of AI-based AICA is still relatively new. More research experiments are required to answer the following questions and motivate AICA adoption into commercial cyber-defense solutions:

- What combination of cyber data, e.g., host, network, cyber threat intelligence, is required for optimal AICA performance?
- How do we curate representative cyber datasets for AICA training and testing?
- How do we design and evaluate more realistic simulated and emulated cyber environments to train AICA for real-world threat perception?
- What cooperative and competitive multi-agent AI methods can be leveraged to evaluate AICA architectures for cyber threat perception?
- How do we mitigate adversarial AI/ML attacks on AICA reasoning and decision-making processes?

References

"2021 Trends Show Increased Globalized Threat of Ransomware | CISA." https://www.cisa.gov/uscert/ncas/current-activity/2022/02/09/2021-trends-show-increased-globalized-threat-ransomware. Accessed 5 Mar 2022.

Al Marakeby, H., Zaki, M., & Shaheen, S. (2010, November). A generalized object detection system using automatic feature selection. In *Proceedings of the 10th international conference on intelligent systems design and applications (ISDA'10), Cairo, Egypt* (pp. 839–844).

Bhuyan, M., Bhattacharyya, D., & Kalita, J. (2014). Network anomaly detection: Methods, systems and tools. *IEEE Communication Surveys and Tutorials, 16*(1), 303–336.

Bradbury, A. (2016, September 29). *OpenC2 and OrchID – Using OpenC2 is a managed security services provider*. OpenC2 Forum.

Bradshaw, J. M. (1997). Chapter 1: An introduction to software agents. In J. M. Bradshaw (Ed.), *Software agents* (pp. 3–46). AAA1 Press/MIT Press.

Cen, L., Gates, C., Si, L., & Li, N. (2014). A probabilistic discriminative model for android malware detection with decompiled source code. *IEEE Transactions on Dependable and Secure Computing, PP*(99), 1–1.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 15:1–15:58. [Online]. Available http://doi.acm.org/10.1145/1541880.1541882

Crosbie, M., & Spafford, E. (1995, October). Defending a computer system using autonomous agents. In *Proceedings of the 18th National Information Systems Security Conference*.

Crowley, J. L., Piater, J. H., Vincze, M., & Paletta, L. (Eds.). (2003, April). *Proceedings of the 3rd international conference on computer vision systems (ICVS'03), Graz, Austria*. Springer.

Cuadra-Sanchez, A., Aracil, J., & Ramos de Santiago, J. (2014, June). Proposal of a new information-theory based technique and analysis of traffic anomaly detection. In *Proceedings of the 2014 international conference on smart communications in network technologies (SaCoNeT'14), Vilanova i la Geltru, Spain* (pp. 1–6).

Elbasiony, R. M., Sallam, E. A., Eltobely, T. E., & Fahmy, M. M. (2013). A hybrid network intrusion detection framework based on random forests and weighted k-means. *Ain Shams Engineering Journal, 4*(4), 753–762. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2090447913000105

Ester, M., Peter Kriegel, H., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 1996 knowledge discovery and data mining conferences (KDD'96), Portland, Oregon, USA* (pp. 226–231). AAAI Press.

Farmer, W. M., Guttman, J. D., & Swarup, V. (1996, October). Security for mobile agents: Issues and requirements. In *Proceedings of the 19th national information systems security conference* (Vol. 2). National Institute of Standards and Technology.

Ferguson-Walter, K. J., Fugate, S. J., Mauger, J., & Major, M. M. (2019, March). Game theory for adaptive defensive cyber deception. In *ACM hot topics in the science of security symposium (HotSoS)*.

Gardner, M., & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron) – A review of applications in the atmospheric sciences. *Atmospheric Environment, 32*(14–15), 2627–2636. Available http://www.sciencedirect.com/science/article/pii/S1352231097004470

Guyonand, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

Ham, H.-S., & Choi, M.-J. (2013, October). Analysis of android malware detection performance using machine learning classifiers. In *Proceedings of the 2013 international conference on ICT convergence (ICTC'13), Jeju Island, Korea* (pp. 490–495).

Heckerman, D. (2008). A tutorial on learning with Bayesian networks. In D. Holmes & L. Jain (Eds.), *Innovations in Bayesian networks* (Studies in computational intelligence) (Vol. 156, pp. 33–82). Springer. [Online]. Available https://doi.org/10.1007/978-3-540-85066-33

Hofmeyr, S. A. (1999, May). *An immunological model of distributed detection and its application to computer security*. PhD thesis, University of New Mexico.

"IACD Spirals 1 to 22 graphic". H. B. J. Caroli, D. Fayette, N. Koziarz, and T. Stedman, "Tools for effects based course of action development and assessment."

Karnin, Z., Liberty, E., Lovett, S., Schwartz, R., Weinstein, O., Mannor, S., Srebro, N., & Williamson, R. C. (2012). Unsupervised SVMs: On the complexity of the furthest hyperplane problem. *Journal of Machine Learning Research, 23*, 1–18.

Kecman, V., & Brooks, J. (2010, July). Locally linear support vector machines and other local models. In *Proceedings of the 2010 international joint conference on neural networks (IJCNN'10), Barcelona, Spain* (pp. 1–6). IEEE.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE, 78*(9), 1464–1480.

Leita, C., Pham, V., Thonnard, O., Ramirez-Silva, E., Pouget, F., Kirda, E., & Dacier, M. (2008). The leurre.com project: Collecting internet threats information using a worldwide distributed honeynet. In *Information security threats data collection and sharing, 2008. WISTDCS'08. WOMBAT workshop on* (pp. 40–57). IEEE.

Mas'ud, M., Sahib, S., Abdollah, M., Selamat, S., & Yusof, R. (2014, May). Analysis of features selection and machine learning classifier in android malware detection. In *Proceedings of the 2014 international conference on information science and applications (ICISA'14), Seoul, Korea* (pp. 1–5).

Migliardi, M., & Merlo, A. (2013). Improving energy efficiency in distributed intrusion detection systems. *Journal of High Speed Networks, 19*(3), 251–264.

Mitchell, T. M. (1997). *Machine learning* (1st ed.). McGraw-Hill, Inc.

Micah Muser and Ashon Garriott (2021) Machine learning and cybersecurity: Hype and reality. Center for Security and Emerging Technology (CSET), Georgetown University. https://cset.georgetown.edu/wp-content/uploads/Machine-Learning-and-Cybersecurity.pdf

"NCCIC CYBER INCIDENT SCORING SYSTEM", "Integrated adaptive cyber defense, IACD." https://www.iacdautomate.org/. Accessed 5 Mar 2022.

Nguyen, T. T., & Reddi, V. J. (2021). Deep reinforcement learning for cybersecurity. *arXiv:1906.05799v4 [cs.CR]*. https://arxiv.org/pdf/1906.05799.pdf

Nigrin, A. (1993). *Neural networks for pattern recognition*. MIT Press.

Peters, W. (2017, March 23). *IACD overview and IACD framework*. IACD Community Day, Laurel, Maryland.

Royer, P. (2016, September 29). *Orchestration and automation*. OpenC2 Forum.

Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). Andromaly: A behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems, 38*(1), 161–190. [Online]. Available https://doi.org/10.1007/s10844-010-0148-x

Shakut, K., Luo, S., Varadharajan, V., Hameed, I. A., & Xu, M. (2020). A survey on machine learning techniques for cyber security in the last decade. *IEEE Open Access Journal*. https://doi.org/10.1109/ACCESS.2020.304195

Sharma, A. K. (2005). *Text book of chi-test and experimental designs* (1st ed.). Publishing House.

Spafford, E. H., & Zamboni, D. (2000). Intrusion detection using autonomous agents. *Computer Networks, 34*(4), 547–570.

Sparrell, D. (2019). Cyber-safety in healthcare IoT. In *11th academic conference ITU kaleidoscope: ICT for health: Networks, standards and innovation, ITU K 2019*. https://doi.org/10.23919/ITUK48006.2019.8996148

Spitzner, L. (2003). The honeynet project: Trapping the hackers. *IEEE Security and Privacy, 1*(2), 15–23. [Online]. Available: https://doi.org/10.1109/MSECP.2003.1193207

Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Wee, C., Yip, R., & Zerkle, D. (1996, October). GrIDS: A graph-based intrusion detection system for large networks. In *Proceedings of the 19th national information systems security conference* (Vol. 1). National Institute of Standards and Technology.

Sundararajan, A., Khan, T., Aburub, H., Sarwat, A. I., & Rahman, S. (2018). A tri-modular human-on-the-loop framework for intelligent smart grid cyber-attack visualization. In *SoutheastCon 2018* (pp. 1–8). https://doi.org/10.1109/SECON.2018.8479180

Thottan, M., Liu, G., & Ji, C. (2010). Anomaly detection approaches for communication networks. In G. Cormode & M. Thottan (Eds.), *Algorithms for next generation networks* (Computer communications and networks) (pp. 239–261). Springer. [Online]. Available https://doi.org/10.1007/978-1-84882-765-311

"US-CERT Year in Review 2012", "Battle against cybercrime continues." https://blog.checkpoint.com/2021/10/06/as-battle-against-cybercrime-continues-during-cybersecurity-awareness-month-check-point-research-reports-40-increase-in-cyberattacks/. Accessed 5 Mar 2022.

Verizon. *2016 data breach report*. Available https://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf

Walter, E. C., Ferguson-Walter, K. J., & Ridley, A. (2021). Incorporating deception into CyberBattleSim for autonomous defense. In *IJCAI 2021 international workshop on adaptive cyber defense. arXiv:2108.13980v1 [cs.CR]*. https://arxiv.org/pdf/2108.13980.pdf

Wang, J. (2012). *Geometric structure of high-dimensional data and dimensionality reduction*. Springer.

White, G. B., Fisch, E. A., & Pooch, U. W. (1996). *Cooperating security managers: A peer-based intrusion detection system* (pp. 20–23). IEEE Network.

Xie, M., Han, S., Tian, B., & Parvin, S. (2011). Anomaly detection in wireless sensor networks: A survey. *Journal of Network and Computer Applications, 34*(4), 1302–1325. Advanced Topics in Cloud Computing. [Online]. Available http://www.sciencedirect.com/science/article/pii/S1084804511000580

Yegneswaran, V., Barford, P., & Paxson, V. (2005). Using honeynets for internet situational awareness. In *Proceedings of the fourth workshop on hot topics in networks (HotNets IV)* (pp. 17–22). Citeseer.