Integrating Collaborative Cognitive Assistants into Cybersecurity Operations Centers

Steven Meckl, Gheorghe Tecuci, Dorin Marcu, Mihai Boicu

Learning Agents Center, Volgenau School of Engineering, George Mason University, Fairfax, VA 22030, USA
smeckl@masonlive.gmu.edu, teuci@gmu.edu, dmarcu@gmu.edu, mboicu@gmu.edu

CEUR-WS.org/Vol-2269/FSS-18_paper_28.pdf

Abstract

This paper presents current work in integrating cognitive agents, trained to detect advanced persistent threats (APTs), into cybersecurity operations centers (CSOCs). After introducing APTs, the cognitive APT detection model, and the training of the agents, it overviews how the collection agents required to gather evidence for cognitive agents are selected, how abductive triggers are generated using collected data and threat intelligence, how the Collection Manager software is used to integrate cognitive agents with selected collection agents, and how results of searches are added to the knowledge base as evidence. These concepts are illustrated with an example of how the system uses these components to search for evidence required to detect an APT attack.

Introduction

The science and art of intrusion detection and prevention has evolved over the last decade, largely due to the shift from cyber vandals and pranksters to multi-billion-dollar criminal enterprises and state-sponsored APT intrusion methodology (Zimmerman, 2014). What was once the realm of criminals with a small collection of easily discovered automated tools is now ruled by well-funded and highly sophisticated sets of hackers carefully orchestrating intrusions as a means to advance their criminal enterprise or intelligence collection mission.

State-sponsored intrusion sets such as the People's Republic of China's (PRC) APT1 have demonstrated that organization, funding, and lack of consequences can be more effective than use of sophisticated intrusion tools (Mandiant, 2013). An APT is an adversary that leverages superior resources, knowledge, and tactics to achieve its goals through computer network exploitation and are notorious for their persistence and ability to adapt to efforts of network defenders in order to gain persistent access to victim networks (Mandiant 2013). APT groups have become a major area of security research over the past several years as threat intelligence companies began tracking their specific tools, techniques, and procedures (TTPs), attributing those TTPs to threat actors, and publishing reports on the groups. FireEye/Mandiant has published reports on 30 APT groups since 2013, naming them simply APT1 through APT30 (FireEye 2015). APT1, the most infamous of the APT groups, was attributed to Unit 61398 of China's People's Liberation Army (Mandiant 2013).

The responsibility of a CSOC's analysts is to monitor alerts and log information from available sources, each having differing levels of credibility, and use them to make decisions about the presence or absence of intrusion activity. However, modern detection technologies are error-prone, because log information can be ambiguous. As a result, each alert must be carefully examined and investigated by a human analyst (Zimmerman 2014). In a large enterprise, thousands of alerts can be reported daily, and most organizations report they are able to investigate less than 50 in a typical work week (Ponemon Institute 2017), leaving most alerts uninvestigated and increasing risk to the enterprise. Improving the efficiency and accuracy of analysis and threat intelligence tasks could drastically lower the cost of running a CSOC and reduce risk to the organization.

One approach to increasing CSOC efficiency is to employ collections of agile cognitive assistants, able to capture and automatically apply the expertise of cybersecurity analysts, to automate detection of APTs and other sophisticated threats (Meckl et al. 2017). These cognitive agents use abductive, deductive, and inductive reasoning to learn from cybersecurity experts how to autonomously respond to CSOC alerts, capture digital evidence related to the alert, and analyze it to detect threats.

A significant architectural challenge to this approach is integration of the agents into real-world CSOCs, which have a wide variety of security infrastructure. There are thousands of commercial and open source security products available for CSOC managers to choose from. For this approach to be successful, cognitive agents must be able to

Copyright © 2018 by the papers authors. Copying permitted for private and academic purposes. In: Joseph Collins, Prithviraj Dasgupta, Ranjeev Mittu (eds.): Proceedings of the AAAI Fall 2018 Symposium on Adversary-Aware Learning Techniques and Trends in Cybersecurity, Arlington, VA, USA, 18-19 October, 2018, published at http://ceur-ws.org

seamlessly interact with security sensors in place in the CSOC with minimal re-architecture of the system. This paper builds on previous research to present current results on researching the automation of CSOCs with agile cognitive assistants. Specifically, we focus on how our agents interact with real-world security infrastructure to detect attacks.

We start with an overview of our approach to teaching a learning agent shell how to detect APTs. Then, we discuss how the search/collection agents required to gather evidence for cognitive agents are selected, how abductive triggers are generated using collected data and threat intelligence, how the Collection Manager software is used to integrate cognitive agents with selected search/collection agents, and how their results are added to the knowledge base as evidence.

APT Detection: Teaching and Learning

For many years we have researched the Disciple theory, methodology, and tools for the development of knowledge-based cognitive assistants that: (1) learn complex problem-solving expertise directly from subject matter experts; (2) support experts and non-experts in problem solving; and (3) teach their problem-solving expertise to students (Tecuci 1988; 1998; Boicu et al. 2000; Tecuci et al. 2005; 2016a).

In the following we summarize how the Disciple-EBR learning agent shell is taught to detect APT intrusions. In essence, an expert cybersecurity analyst teaches Disciple-EBR in a way that is similar to how the expert would teach a student or an apprentice, by explaining APT detection examples to it. The agent learns general rules from these examples and applies them to new situations. Then the expert critiques the attempted detections by the agent, and the agent improves its learned rules and its ontology, to more accurately simulate the detection activity of the expert. Fig.1 is an abstract illustration of this process.

First, the expert specifies an event of interest, or trigger (T_1 in Fig.1), that should alert the agent of a potential intrusion, for example, an alert generated by the BRO IDS (Paxson, 1999) in the case of an intrusion by the Auriga malware of APT1. The problem is that such an alert may be generated by BRO also in cases when there is no intrusion, called false positives.

Thus, once such an event is generated by BRO in a real situation, the expert's question is: What hypotheses would explain it? Therefore, the next step in the teaching process is to specify the abductive steps of generating alternative hypotheses that may explain this alert, as abstractly shown in the left part of Fig.1. Some of these hypotheses are APT1 intrusion hypotheses (e.g., H_1), but others are false positive hypotheses (e.g., H_q). From these abductive reasoning steps, the agent will learn trigger, indicator, and question rules. These rules will enable the agent to automatically generate alternative hypotheses that could explain similar triggers, as will be discussed later and illustrated in Fig.6.

Once the hypotheses that can explain the trigger are generated, the expert would need to assess which of them is most likely, and thus to determine whether there is an intrusion or not. For this, however, the expert would need more evidence. The expert will put each hypothesis to work to guide him/her in the process of collecting this evidence, as abstractly illustrated in the right-hand side of Fig.1. In particular, the expert will decompose H_1 into simpler and simpler hypotheses, down to the level of hypotheses that show very clearly what evidence is needed to prove them. For example, H_1 would be true if H_1^1 and … and H_n^1 would be true. Then, to determine whether H_n^1 is true, one would need to invoke a search procedure S_n^2 that may return evidence E_n^2, if present. If E_n^2 is found, its credibility and relevance determine the probability of H_n^1 (Tecuci et al. 2016b). Once the probabilities of the sub-hypotheses are assessed, the probabilities of the upper-level hypotheses are determined, and one may conclude whether there is an intrusion or not.

From a decomposition tree like that in Fig.1, the agent will learn both hypothesis analysis rules and collection rules. These rules will enable the agent to automatically decompose similar hypotheses and search for evidence, as will be discussed later and illustrated in Fig.7.

Each of the rules mentioned above is initially partially learned as an ontology-based generalization of one example and its explanation. They are then used in reasoning to discover additional positive and negative examples and are further incrementally refined based on these new examples and their explanations. The Disciple approach is based on methods for integrating machine learning with knowledge acquisition (Tecuci and Kodratoff 1995), and on multi-strategy learning (Tecuci, 1988; 1993; Tecuci et al. 2016a).

Agent teaching and learning is a continuous process resulting in the customization of Disciple-EBR into an agent

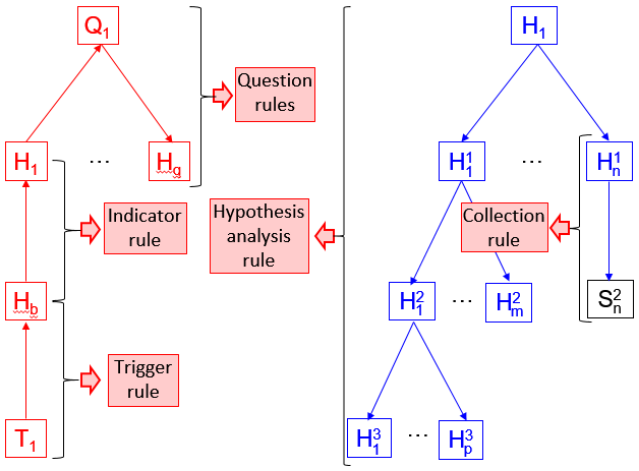

Figure 1: Agent teaching and learning.

that not only has reasoning modules for all the phases of the APT1 intrusion detection process, but also a knowledge base (KB) with a developed ontology and reasoning rules for all these phases. This agent is used to generate several autonomous agents, each specialized for a specific phase of APT intrusion detection in a CSOC, as discussed next.

Cognitive Assistants for APT Detection

Fig.2 shows an overview of the architecture of the Cognitive Assistants for APT Detection (CAAPT).

The ***Trigger Agent*** receives alerts from a variety of sources, such as network IDSs or endpoint protection systems, uses a matching trigger rule to represent the alert in ontological form in a KB, and places that KB into the *hypothesis generation queue* from which KBs are extracted by the Hypothesis Generation Agent.

The ***Hypothesis Generation Agent*** uses indicator and question rules to generate alternative hypotheses that could explain the trigger and places the KB into the *hypothesis analysis queue* from which KBs are extracted by the Automatic Analysis Agents.

The ***Automatic Analysis Agents*** use hypothesis analysis rules to decompose the hypotheses from such a KB, as much as possible, down to the level of evidence collection requests. Then they place the KB into the *evidence collection queue* from where each KB is extracted by the Collection Manager.

The ***Collection Manager*** uses collection rules to generate search requests and invokes specialized collection agents to search for evidence on the network in response to these requests. Then it uses matching collection rules to represent the retrieved evidence into the corresponding KB and places the KB back into the hypothesis analysis queue for further analysis (if needed), until the hypothesis with the highest probability is determined.

Selection and Integration of Collection Agents

Abstract searches requested by the analysis agents require evidence from multiple types of data sources available on a typical network. There are hundreds of security appliances, log source, and data store combinations in real-world networks. Therefore, a comprehensive set of corresponding collection agents is required. Industry research has determined that the most critical security technologies are a Security Incident/Event Management (SIEM) system, a network detection/collection solution, and a host detection and query solution (Chuvakin, 2018), so our selection of agents focused on those areas. We have chosen to use those critical technologies, as well as others, as needed, broken down into the following categories from the ontology in Fig. 3.

Passive network monitors are responsible for passively watching data as it moves across the network and either recording it in full or recording metadata in the form of logs,

Passive host monitors, which watch operating system and application activity on a host computer and record metadata as logs.

On-demand host agents, which allow for collection and analysis of raw forensic artifacts, including disk and memory data, from workstations and servers. They can also be used to retrieve log data generated by *passive host monitors* in an on-demand fashion.

For CAAPT, collection agents were chosen based primarily on their ability to collect and query data required for detecting sophisticated attacks. Based on the requirements for modeling detection for APT1 malware, we chose a collection of agents for netflow (network connection) data, full packet

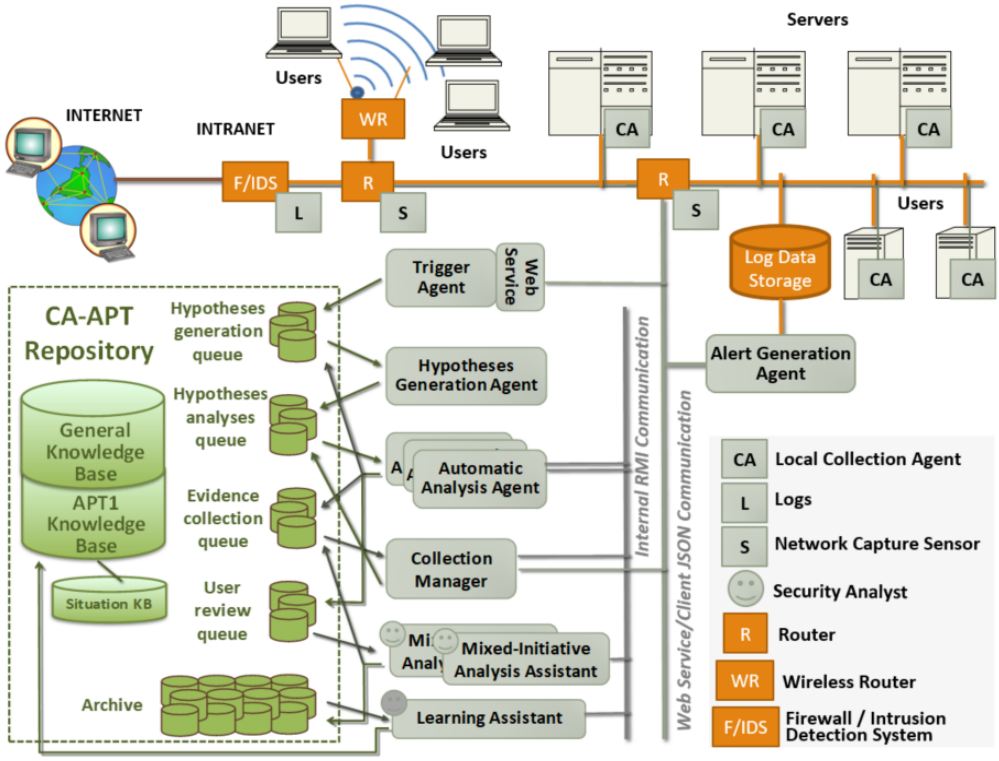

Figure 2: CAAPT architecture overview.

capture, DNS logs, volatile memory artifacts, Windows Registry keys and values, file-based artifacts, endpoint logs, domain controller logs, EDR logs, and passive DNS data. Next, free or open source solutions were prioritized to eliminate cost as a barrier to adoption. Lastly, we chose tools supporting a RESTful API (MuleSoft, 2016) for uniformity of integration.

GRR is used as our sole on-demand host agent and is comprised of an agent which must be run on each host computer on the network and a server responsible for managing search requests. GRR's collection functions are managed using its RESTful API.

The output of passive collection agents is log data which must be stored in a SIEM based on tools such as Elasticsearch or Splunk. For CAAPT, we use Elasticsearch as a SIEM. As shown in Fig.4, all passive collectors used by the system send log data, in the form of JSON documents, to Elasticsearch for storage and indexing.

For collection agents having a non-JSON log format (such as SYSMON and BRO) Elasticsearch Beats (Beats, 2018) are used to convert the logs to JSON before sending them to Elasticsearch. This includes Filebeat for collecting BRO logs, Winlogbeat for collecting SYSMON and Windows system logs, and Packetbeat for collecting raw network data.

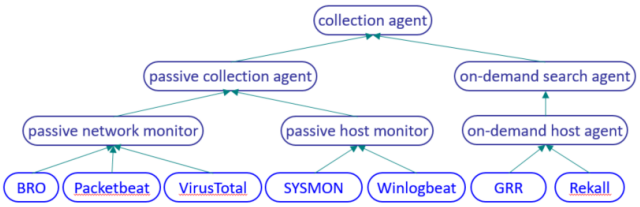

Figure 3: Ontology of search and collection agents.

VirusTotal provides a free passive DNS database which is used for historical domain/IP resolution queries, and Rekall (2017) is used by GRR to retrieve raw memory artifacts from hosts.

Generating Abductive Triggers from Threat Intelligence

The cognitive assistants in CAAPT respond to and investigate security alerts generated by a CSOC's security infrastructure, where an analyst is typically required to conduct follow-on analysis to determine whether a threat was accurately identified. The first step in this process is to use available detection technologies to identify potential threats based on threat intelligence and use the resulting security alerts to trigger the abductive reasoning process. This section describes the process CAAPT uses to generate abductive triggers from threat intelligence.

At its core, security alerts are created by applying threat intelligence to data collected by security sensors, which include endpoint security such as anti-virus software or network-based firewalls and intrusion detection systems. Threat intelligence data is distributed in the form of *indicators of compromise* (IOCs), which include file hashes, anti-virus signatures, and the fully-qualified domain names (FQDNs) or IP addresses of known malicious servers. Security alerts come in a variety of formats but are normalized and sent to the SIEM.

Fig.5 shows an overview of the process by which a BRO alert log entry becomes an alert message sent to the CAAPT Trigger Agent. BRO was chosen as the IDS because of its ability to easily consume network threat intelligence and efficiently apply it to identify threats, it also generates logs for network metadata for use in follow-on analysis.

The first step in the process is to convert logs from the CSV format to a JSON message and transport the log entry to our Elasticsearch database. This is done using FileBeat.

Next, a process is required to search Elasticsearch for new alerts and send them to the Trigger Agent. We developed a custom Windows program called the *Alert Generation Service* to perform this task. This service polls Elasticsearch on a specified interval, looking for alert log entries generated by BRO. A new Trigger Agent message is created for each new alert, using relevant information from the BRO alert, and sent to the Trigger Agent to start the analytic process.

In the example in Fig.5, an alert was generated by BRO because a computer it was monitoring made a DNS request to resolve a domain known, via threat intelligence, to be associated with APT1. The message sent to Elasticsearch contains several extemporaneous data elements added by FileBeat. For the purposes of threat detection, the system is primarily concerned with the information contained in the *message* data element. The Alert Generation Service parses that field, converting the relevant data fields into appropriate data elements required for the JSON message on the right.

When the message is received by the Trigger Agent, it is added to the knowledge base in the form of an ontology fragment, as shown on the far right of Fig.5. Each field in the JSON message is ingested as an instance of a fact in the ontology. In this example, knowledge of the connection (source and destination IPs and ports) is captured directly

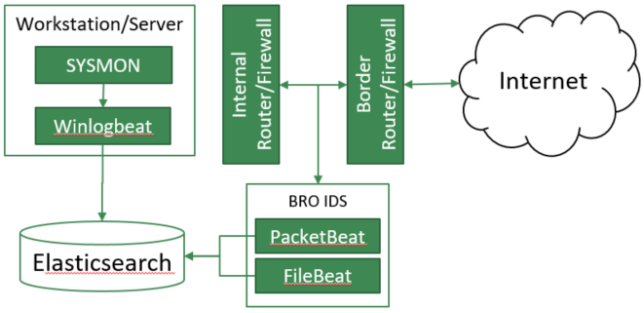

Figure 4: Passive collector integration architecture.

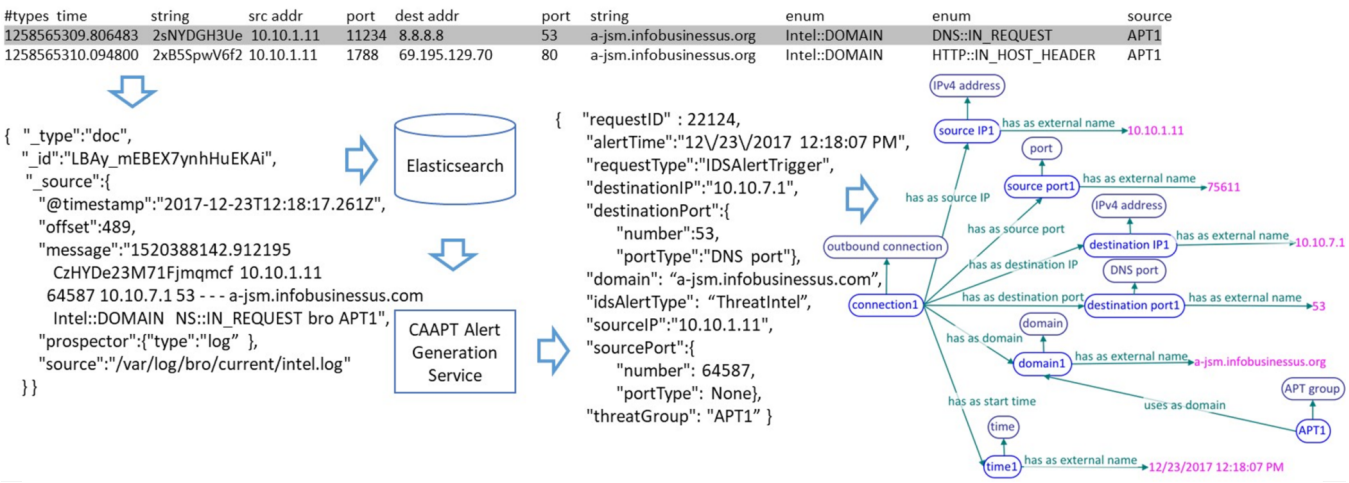

Figure 5: How security alerts become abductive triggers.

from the BRO alert. The Trigger Agent can further specify, based on a learned rule, that the destination port is for DNS because it uses standard port 53. The domain *a-jsm.infobusinessus.org* is associated with the connection. Because the knowledge base includes modeled knowledge of adversary TTPs, the domain's association with APT1 is automatically recognized. The alert information is added to the knowledge base using ontology actions associated with the learned trigger rule that matched the BRO alert.

When the process described in Fig.5 is complete, the Trigger Agent places a new knowledge base to be used for further analysis into the hypothesis generation queue. The Hypothesis Generation Agent then uses the new knowledge base for the abductive reasoning process, using learned rules to create a set of multiple competing hypotheses which could explain why the alert was generated. First, an indicator rule is matched, which generates a hypothesis from the suspicious connection that there is an active APT1 intrusion

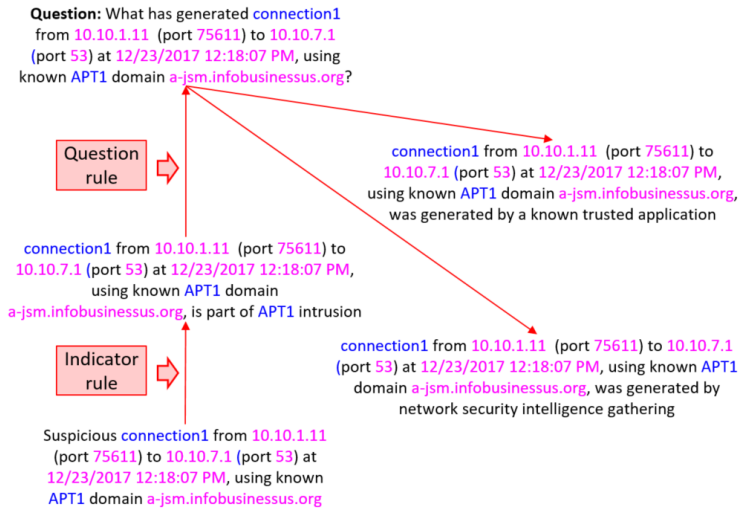

Figure 6: Hypotheses generation from abductive trigger.

on the network. Then a question rule is matched, to generate a question which the previously generated hypothesis could answer. From the question rule, multiple competing plausible hypotheses are generated, which also could answer the question. Generation of the set of multiple competing hypotheses is called *abductive reasoning* and completes the first phase of the theoretical model of APT detection.

Fig.6 shows an example of the abductive reasoning process. Using an indicator rule, the agent generates the hypothesis that the connection is part of an APT1 intrusion. However, there are multiple hypotheses which could explain the connection, including both those generated based on modeled adversary knowledge and those that would indicate a false positive. In this example, we offer two plausible false positive hypotheses. The first is that connection was generated as part of security intelligence gathering. Security operations or research personnel often accidentally trigger security alerts performing their duties. The second hypothesis is that a trusted application made the connection. Security tools often perform DNS lookups as part of their data enrichment features. Unless the IDS is configured to exclude applications performing this type of activity, false positive alerts can be generated. It should be noted both of these types of false positives were accidentally triggered during the course of this research.

Hypothesis-driven Search for Evidence

Once a set of hypotheses are generated, the next phase is the deductive reasoning process, where each top-level hypothesis is decomposed into one or more sub-hypotheses. The process, executed by an Automatic Analysis Agent, continues until a set of leaf hypotheses are generated requiring one or more searches for evidence. This overall process is called hypothesis-driven search.

Fig.7 shows an example of the initial hypothesis decomposition tree for the detection of an APT1 intrusion, based on modeled adversary knowledge. At the top level, we decompose the hypothesis stating the network connection causing the BRO alert is the result of an APT1 intrusion, into two sub-hypotheses. The sub-hypothesis on the left states the connection involves an active C2 server. This hypothesis is further broken down to its two components: the domain *a-jsm.infobusinessus.org* was active at the time of the connection and was registered using a dynamic DNS provider. These are typically true when there is an active APT1 attack. The sub-hypothesis on the right states the program used in the attack is APT1 malware.

The leaf nodes of the decomposition tree result in three different searches for evidence. All three searches will eventually lead to evidence being added to the KB for this security alert investigation. The search for the program that made the network connection will result in that branch of the decomposition tree being further decomposed, asking more detailed questions about the behavior of the malware.

Using Search Results as Evidence

The abstract searches from Fig.7 must be turned into concrete searches for real evidence on the network. The Collection Manager is responsible for that process. Let's consider, for example, the left-most search from Fig.7, which is looking for the IP address to which a domain was mapped at a specific point in time. Using the JSON request template associated with the collection rule that generated that leaf, the JSON-formatted search message at the top center of Fig.8 is created and sent to the Collection Manager.

When the Collection Manager receives this message, it will call the function **GetDomainIPResolution**, which is one of the programmed search functions supported by the Collection Manager, mapping data elements from the search into function parameters. **GetDomainIPResolution** uses a passive DNS database, such as the one maintained by VirusTotal, to determine the IP address mapped to APT1 domain a-jsm.infobusinessus.org at the time specified in the *timeStamp* field in the JSON request.

Modeling Search Results as Evidence

When the Collection Manager completes the **GetDomainIPResolution** search, it will respond to the calling agent with the JSON-formatted response from the upper right part of Fig.8.

The response message contains the input parameters from the search request, a *requestID* used by the Collection Manager and calling agent to map response messages to the corresponding request messages, and the output parameters. In this example, it was determined, based on passive DNS data, that the domain *a-jsm.infobusinessus.org* was mapped to the IP address 69.195.129.72 at time 12/23/2017 12:18:07 PM because that IP address was assigned to the domain at the time in *mappingStartTime* (01/15/2017 11:11 GMT" and it is still assigned (*mappingEndTime* is set to "present").

The response message also includes a human-readable description of what the found evidence means (*evidenceDescription*), and the value denoted in the *evidenceCredibility*, which is a value on a linear probability scale ranging from L0 (Lack of Support) to L11 (Certain). In this case, the Collection Manager is certain the returned evidence is credible. The response is matched with the JSON response template associated with the corresponding collection rule, and the ontology actions associated with the same rule are used to generate the ontology fragments corresponding to the data elements in the response message and store them in the knowledge base as evidence as shown in Fig.8.

This evidence can now be used by automatic analysis agents to decide on the likelihood of an actual APT1 attack present on the network, or to request additional evidence, if needed.

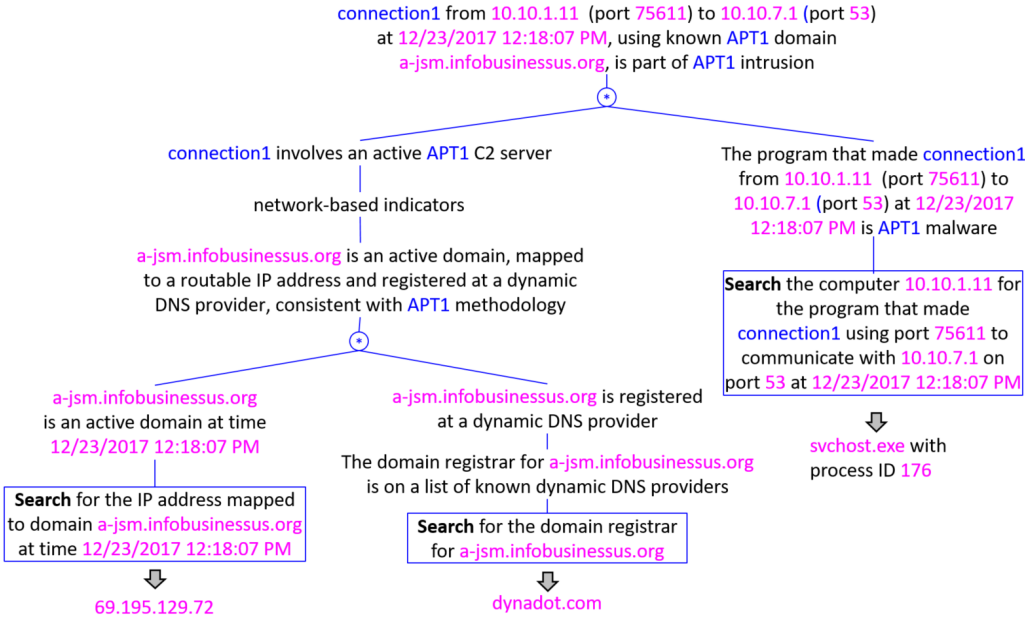

Figure 7: Hypothesis-driven search for evidence.

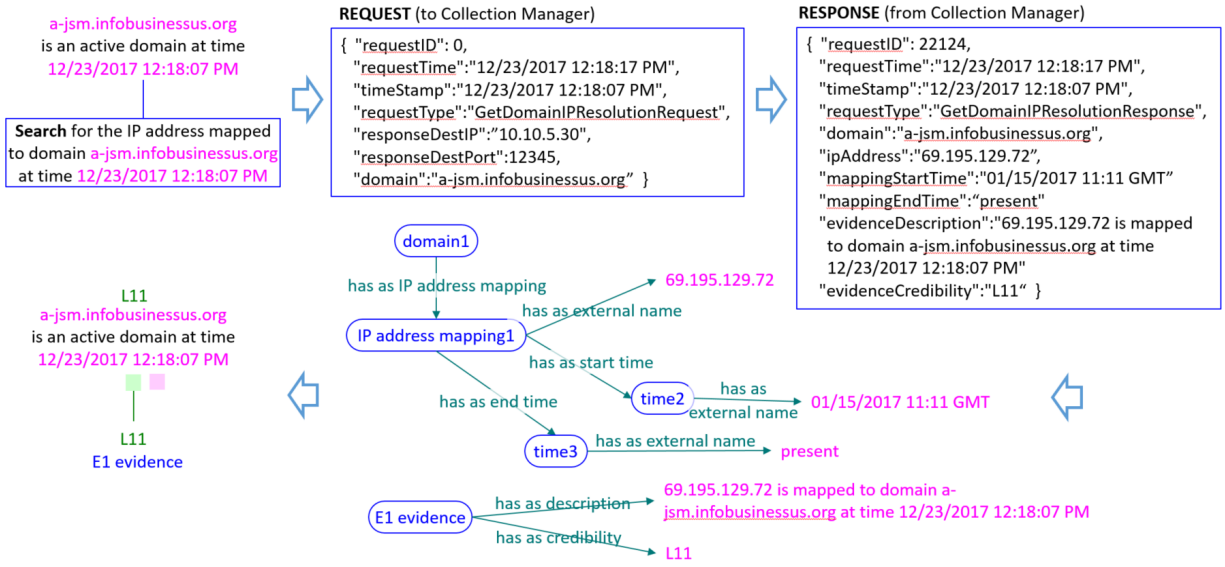

Figure 8: Automatic evidence collection and use.

Collection Manager Architecture

The Collection Manager is the main integration point between the cognitive agents and CSOC infrastructure. The analysis agents know what information is needed to expand their analyses, but the search requests are in abstract form. The primary function of the Collection Manager is translating high-level (abstract) search instructions into specific API calls to host and network agents, determining which such agent to send search requests to, and wrapping calls to specific search agents with a RESTful API. Results returned from search agents are then converted into evidence and added to the knowledge bases of the analysis agents.

Fig.9 is an overview of the Collection Manger process. When the analysis agents analyze competing hypotheses, many of the searches generated by the hypothesis-driven search process (such as those in Fig.7) are sent to the Collection Manager and added to the request queue. Requests are then dispatched for processing and a receipt message is sent back to the caller. The receipt includes the *requestID* so the response can be matched to the search. To minimize redundant searching and increase performance, the Collection Manager performs caching of search results. Each search request is hashed, and the hash value is used as a key to the cache table. A duplicate search will have a matching hash value and its result can be used instead of re-executing the search. When a search is conducted, results of each search are added to the cache with the appropriate key and a time to live (TTL) value. Once the TTL is expired, the search results are considered invalid and are purged from the cache.

Abstract searches requested by analysis agents require evidence from multiple types of data sources available to the CSOC. In order for the analysis agents to integrate with real networks, the Collection Manager uses a plugin architecture for search agent wrappers, allowing it to easily translate abstract search requests into requests for information from real search agents.

Depending on the amount of time required to complete a search, requests to a search agent can be either synchronous or asynchronous. From the perspective of analysis agents, all requests are asynchronous, but internally, the Collection Manager supports both call models.

In the synchronous call model, there is a single thread for responsible for dequeuing abstract search requests from the Request Queue, formatting a concrete search for a specific search agent and dispatching the request to the appropriate search agent. It then waits on the TCP connection for a synchronous response. When it is received, the thread is responsible for parsing the response to extract digital artifacts, formatting it as evidence, and sending it to the caller.

In the asynchronous call model, the call happens in two threads. In one thread, the abstract search request is received from the Request Queue. The request is parsed, and a concrete search request is prepared and sent to the intended search target. A second thread is responsible for polling the search target on an interval for the response. When it is available, the response data is parsed, artifacts are extracted, and a response is prepared and sent to the Response Queue.

The Collection Manager's flexible architecture and support of multiple call models allows it to easily integrate with a wide variety of security infrastructure, making the process of porting CAAPT to a new CSOC straightforward.

Status and Future Research

A first prototype of the presented system was completed in January 2018. A final prototype is currently under development, with a focus on improving agent teaching, experimen-

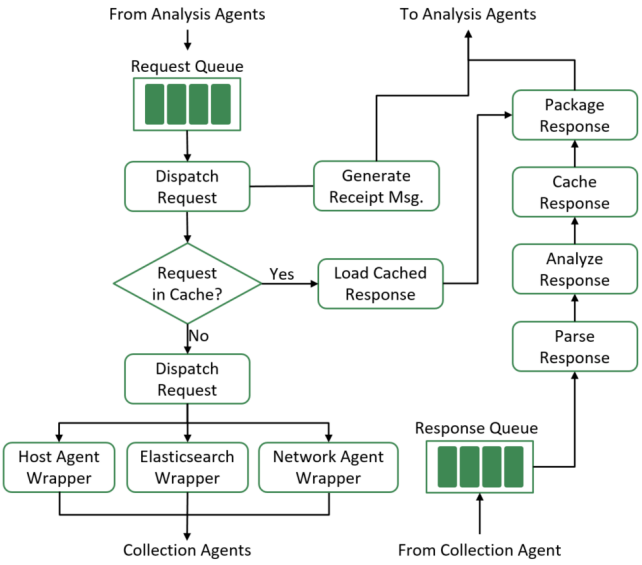

Figure 9: Collection Manager Overview.

tation within a simulated CSOC, and experimental integration into a real CSOC. The CAAPT system will be evaluated using a testing methodology designed to demonstrate its ability to detect and adapt to APT attacks. Additionally, the system's performance will be compared against data collected from a large CSOC to compare it to manual analysis processes in real-world situations. We expect the final prototype will significantly increase the probability of detecting intrusion activity while drastically reducing the workload of the operators.

Acknowledgements

This research was sponsored by the Air Force Research Laboratory (AFRL) under contract number FA8750-17-C-0002, and by George Mason University. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

References

Beats. 2017. Beats, https://www.elastic.co/products/beats

Boicu, M., Tecuci, G., Marcu, D., Bowman, M., Shyr, P., Ciucu, F., and Levcovici, C. 2000. Disciple-COA: From Agent Programming to Agent Teaching, *Proc. 27th Int. Conf. on Machine Learning (ICML)*, Stanford, California: Morgan Kaufman.

Chuvakin, A., 2018. The Best Starting Technology for Detection? https://blogs.gartner.com/anton-chuvakin/2018/03/06/the-best-starting-technology-for-detection/

Elasticsearch, 2015. Elasticsearch: RESTful, Distributed Search & Analytics, https://www.elastic.co/products/elasticsearch

EnCase, 2017. EnCase Endpoint Investigator - Remote Digital Investigation Solution, https://www.guidancesoftware.com/encase-endpoint-investigator

FireEye. 2015. APT30 and the Mechanics of a Long-Running Cyber Espionage Operation, https://www.fireeye.com/blog/threat-research/2015/04/apt_30_and_the_mecha.html

GRR, 2013. Grr: GRR Rapid Response: Remote Live Forensics for Incident Response, https://github.com/google/grr

Mandiant, 2013. APT1 - Exposing One of China's Cyber Espionage Units, https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf

Meckl, S., Tecuci, G., Marcu, D., Boicu, M., and Bin Zaman, A., 2017. Collaborative Cognitive Assistants for Advanced Persistent Threat Detection, in *Proceedings of the 2017 AAAI Fall Symposium "Cognitive Assistance in Government and Public Sector Applications,"*171-178, AAAI Technical Report FS-17-02, Arlington, VA: AAAI Press, Palo Alto, CA.

MuleSoft, 2016. What Is REST API Design? https://www.mulesoft.com/resources/api/what-is-rest-api-design.

Splunk, 2015. Operational Intelligence, Log Management, Application Management, Enterprise Security and Compliance, http://www.splunk.com/

Paxson, V., 1999. Bro: A System for Detecting Network Intruders in Real-Time, *Computer Networks 31*, 23–24, https://doi.org/10.1016/S1389-1286(99)00112-7

Rekall, 2017. Rekall Memory Forensic Framework, http://www.rekall-forensic.com/

Tecuci, G. 1988. Disciple: A Theory, Methodology and System for Learning Expert Knowledge, *Thése de Docteur en Science*, University of Paris South.

Tecuci G., 1998. Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies, San Diego: Academic Press.

Tecuci G., Boicu M., Boicu C., Marcu D., Stanescu B., and Barbulescu M., 2005. The Disciple-RKF Learning and Reasoning Agent, *Computational Intelligence*, Vol.21, No.4, pp. 462-479.

Tecuci G., Boicu M., Marcu D., Boicu C., and Barbulescu M. 2008. Disciple-LTA: Learning, Tutoring and Analytic Assistance, *Journal of Intelligence Community Research and Development.*

Tecuci G., Marcu D., Boicu M., Schum D.A., 2016a. *Knowledge Engineering: Building Cognitive Assistants for Evidence-based Reasoning*, Cambridge University Press.

Tecuci, G., Schum, D. A., Marcu, D., and Boicu, M. 2016b. *Intelligence Analysis as Discovery of Evidence, Hypotheses, and Arguments: Connecting the Dots,* Cambridge University Press.

Tecuci, G., 1993. Plausible justification trees: A framework for deep and dynamic integration of learning strategies. *Machine Learning*, 11(2-3), pp.237-261.

Tecuci G., Kodratoff Y. (eds.) 1995. *Machine Learning and Knowledge Acquisition: Integrated Approaches,* Academic Press.

Tecuci G., Marcu D., Meckl S., Boicu M., 2018. Evidence-based Detection of Advanced Persistent Threats, in *Computing in Science and Engineering,* November.

Ponemon Institute, 2017. The Cost of Insecure Endpoints. https://datasecurity.dell.com/wp-content/uploads/2017/09/ponemon-cost-of-insecure-endpoints.pdf

Volatility, 2015. Volatility, *GitHub,* https://github.com/volatilityfoundation/volatility

Zimmerman, C., 2014. *Ten Strategies of a World-Class Cybersecurity Operations Center*. MITRE Press.