Early Detection of Cybersecurity Threats Using Collaborative Cognition

Sandeep Narayanan, Ashwinkumar Ganesan, Karuna Joshi, Tim Oates, Anupam Joshi and Tim Finin
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County, Baltimore, MD 21250, USA
{sand7, gashwin1, kjoshi1, oates, joshi, finin}@umbc.edu

***Abstract*—The early detection of cybersecurity events such as attacks is challenging given the constantly evolving threat landscape. Even with advanced monitoring, sophisticated attackers can spend more than 100 days in a system before being detected. This paper describes a novel, collaborative framework that assists a security analyst by exploiting the power of semantically rich knowledge representation and reasoning integrated with different machine learning techniques. Our Cognitive Cybersecurity System ingests information from various textual sources and stores them in a common knowledge graph using terms from an extended version of the Unified Cybersecurity Ontology. The system then reasons over the knowledge graph that combines a variety of collaborative agents representing host and network-based sensors to derive improved actionable intelligence for security administrators, decreasing their cognitive load and increasing their confidence in the result. We describe a proof of concept framework for our approach and demonstrate its capabilities by testing it against a custom-built ransomware similar to *WannaCry*.**

I. Introduction

A wide and varied range of security tools and systems are available to detect and mitigate cybersecurity attacks, including intrusion detection systems (IDS), intrusion detection and prevention systems (IDPS), firewalls, advanced security appliances (ASA), next-gen intrusion prevention systems (NGIPS), cloud security tools, and data center security tools. However, cybersecurity threats and the associated costs to defend against them are surging. Sophisticated attackers can still spend more than 100 days [8] in a victim's system without being detected. 23,000 new malware samples are produced daily [33] and a company's average cost for a data breach is about $3.4 million according to a Microsoft study [20]. Several factors ranging from information flooding to slow response-time, render existing techniques ineffective and unable to reduce the damage caused by these cyber-attacks.

Modern security information and event management (SIEM) systems emerged when early security monitoring systems like IDSs and IDPSs began to flood security analysts with alerts. LogRhythm, Splunk, IBM QRadar, and AlienVault are a few of the commercially available SIEM systems [11]. A typical SIEM collects security-log events from a large array of machines in an enterprise, aggregates this data centrally, and analyzes it to provide security analysts with alerts. However, despite ingesting large volumes of host/network sensor data, their reports are hard to understand, noisy, and typically lack actionable details [39]. 81% of users reported being bothered by noise in existing systems in a recent survey on SIEM efficiency [40]. What is missing in such systems is a collaborative effort, not just aggregating data from the host and network sensors, but also their integration and the ability to reason over threat intelligence and sensed data gathered from collaborative sources.

In this paper, we describe a cognitive assistant for the early detection of cybersecurity attacks that is based on collaboration between disparate components. It ingests information about newly published vulnerabilities from multiple threat intelligence sources and represents it in a machine-inferable knowledge graph. The current state of the enterprise/network being monitored is also represented in the same knowledge graph by integrating data from the collaborating traditional sensors, like host IDSs, firewalls, and network IDSs. Unlike many traditional systems that present this information to an analyst to correlate and detect, our system fuses threat intelligence with observed data to detect attacks early, ideally before the exploit has started. Such a cognitive analysis not only reduces the false positives but also reduces the cognitive load on the analyst.

Cyber threat intelligence comes from a variety of textual sources. A key challenge with sources like blogs and security bulletins is their inherent incompleteness. Often, they are written for specific audiences and do not explain or define what each term means. For example, an excerpt from the Microsoft security bulletin is *"The most severe of the vulnerabilities could allow remote code execution if an attacker sends specially crafted messages to a Microsoft Server Message Block 1.0 (SMBv1) server."* [22]. Since this text is intended for security experts, the rest of the article does not define or describe *remote code execution* or *SMB server*.

To fill this gap, we use the Unified Cybersecurity Ontology [36] (UCO)[1] to represent cybersecurity domain knowledge. It provides a common semantic schema for information from disparate sources, allowing their data to be integrated. Concepts and standards from different intelligent sources like STIX [1], CVE [21], CCE [24], CVSS [9], CAPEC [23], CYBOX [25], and STUCCO [12] can be represented directly using UCO.

We have developed a proof of concept system that ingests information from textual sources, combines it with the knowl-

[1]https://github.com/Ebiquity/Unified-Cybersecurity-Ontology

edge about a system's state as observed by collaborating hosts and network sensors, and reasons over them to detect known (and potentially unknown) attacks. We developed multiple agents, including a process monitoring agent, a file monitoring agent and a Snort agent, that run on respective machines and provide data to the Cognitive CyberSecurity (CCS) module. This module reasons over the data and stored knowledge graph to detect various cybersecurity events. The detected events are then reported to the security analyst using a dashboard interface described in section V-D. We also developed a custom ransomware program, similar to *Wannacry*, to test the effectiveness of our prototype system. Its design and working are described in section VI-A. We build upon our earlier work in this domain [26].

The rest of this paper is organized as follows. Section II identifies key challenges in cybersecurity attack detection followed by a brief discussion of related work in Section III. Our cognitive approach to detect cybersecurity events is described in Section IV. Implementation details of our prototype system and a concrete use case scenario to demonstrate our system's effectiveness are in Sections V and VI, before we discuss our future directions in Section VII.

II. Background

Despite the existence of several tools in the security space, attack detection is still a challenging task. Often, attackers adapt themselves to newer security systems and find new ways past them. This section describes some challenges in detecting cybersecurity attacks.

A critical issue which affects the spread and associated costs of a cyber-attack is the time gap between an exploit becoming public and the systems being patched in response. This is evident with the infamous *Wannacry* ransomware. The core vulnerability used by *Wannacry* (Windows SMB Remote Code Execution Vulnerability) was first published by Microsoft Security Bulletin [22] and Cisco NGFW in March 2017. Later in April 2017, *Shadow Brokers* (a hacker group) released a set of tools including *Eternal Blue*[2] and *Double Pulsar* which used this vulnerability to gain access to victim machines. It was only by mid-May that the actual *Wannacry* ransomware started to spread[3] internally using these tools. A large-scale spread of *Wannacry* that affected over two hundred thousand machines could have been mitigated if it had been quickly identified and affected systems had been patched.

Variations of the same cyber-attack is another challenge faced by existing attack detection systems. Many enterprise tools still use signatures and policies specific to attacks for detection. However, smart attackers evade such systems by slightly modifying existing attacks. Sometimes, hackers even use combinations of tools from other attacks to evade them. An example is the Petya ransomware[4] attack, which was discovered in 2016 and spreads via email attachments and infected computers running Windows. It overwrites the Master Boot Record (MBR), installs a custom boot loader, and forces a system to reboot. The custom boot-loader then encrypts the Master-File-Table (MFT) records and renders the complete file system unreadable. The attack did not result in large-scale infection of machines. However, another attack surfaced in 2017 that shares significant code with Petya. In the new attack, named NotPetya[5], attackers use *Eternal Blue* to spread rather than using email attachments. Often, the malware itself is encrypted and similar code is hard to detect. By modifying how they spread, systems used to detect potential behavioral signatures can also be bypassed.

Yet another challenge in attack detection is a class of attacks called Advanced Persistent Threats (APTs). These tend to be sophisticated and persistent over a longer time period [18][34]. The attackers gain illegal access to an organization's network and may go undetected for a significant time with knowledge of the complete scope of attack remaining unknown. Unlike other common threats, such as viruses and trojans, APTs are implemented in multiple stages [34]. The stages broadly include a reconnaissance (or surveillance) of the target network or hosts, gaining illegal access, payload delivery, and execution of malicious programs [3]. Although these steps remain the same, the specific vulnerabilities used to perform them might change from one APT to another. Hence, new approaches for detecting threats (or APTs) should have the ability to adapt to the evolving threats and thereby help detect the attacks early on.

Our prototype system, detailed in Section IV ingests knowledge from different threat intelligence sources and represents them in such a way that it can be directly used for attack detection. Such fast adaptation capabilities help our system cater to changing threat landscapes. It also helps to reduce the time gap problem described earlier. Moreover, the presence of the knowledge graph and reasoning based on them helps to identify variations in attacks.

III. Related Work

A. Security & Event Management

As the complexity of threats and APTs grow, several companies have released commercial platforms for security information and event management (SIEM) that integrate information from different sources. A typical SIEM has a number of features such as managing logs from disparate sources, correlation analysis of various events, and mechanisms to alert system administrators [35]. IBM's QRadar, for example, can manage logs, detect anomalies, assess vulnerabilities, and perform forensic analysis of known incidents [15]. Its threat intelligence comes from IBM's X-Force [27]. Cisco's Talos [5] is another threat intelligence system. Many SIEMs[6], such as LogRhythm, Splunk, AlienVault, Micro Focus, McAfee, LogPoint, Dell Technologies (RSA), Elastic, Rapid 7 and

[2]https://en.wikipedia.org/wiki/EternalBlue

[3]https://en.wikipedia.org/wiki/WannaCry_ransomware_attack

[4]https://blog.checkpoint.com/2016/04/11/decrypting-the-petya-ransomware/

[5]https://www.csoonline.com/article/3233210/ransomware/petya-ransomware-and-notpetya-malware-what-you-need-to-know-now.html

[6]https://www.gartner.com/reviews/market/security-information-event-management/compare/logrhythm-vs-logpoint-vs-splunk

Comodo, exist in the market with capabilities including real-time monitoring, threat intelligence, behavior profiling, data and user monitoring, application monitoring, log management and analytics.

B. Ontology based Systems

Obrst et al. [29] detail a process to design an ontology for the cybersecurity domain. The study is based on the diamond model that defines malicious activity [16]. Ontologies are constructed in a three-tier architecture consisting of a domain-specific ontology at the lowest layer, a mid-level ontology that clusters and defines multiple domains together and an upper-level ontology that is defined to be as universal as possible. Multiple ontologies designed later-on have used the above mentioned process.

Oltramari et al. [31] created CRATELO as a three layered ontology to characterize different network security threats. The layers include an ontology for secure operations (OSCO) that combines different domain ontologies, a security-related middle ontology (SECCO) that extends security concepts, and the DOLCE ontology [19] at the higher level. In Oltramari et al. [30], a simplified version of the DOLCE ontology (DOLCE-SPRAY) is used to show how a SQL injection attack can be detected.

Ben-Asher et al.[2] designed a hybrid ontology-based model combining a network packet-centric ontology (representing network-traffic) with an adaptive cognitive agent. It learns how humans make decisions while defending against malicious attacks. The agent is based on instance-based learning theory using reinforcement learning to improve decision making through experience. Gregio et al. [13] discusses a comprehensive ontology to define malware behavior.

Each of these systems and ontologies looks at a narrow subset of information, such as network traffic or host system information, while SIEM products do not use the vast capabilities and benefits of an ontological approach and systems to reason using them. In this regard, Cognitive CyberSecurity (CCS) takes a larger and more comprehensive view of security threats by integrating information from multiple existing ontologies as well as network and host-based sensors (including system information). It creates a single representative view of the data for system administrators and then provides a framework to reason across these various sources of data.

This paper significantly improves our previous work [37], [38], [26] in this domain, where semantic rules were used to detect cybersecurity attacks. CCS uses the Unified Cybersecurity Ontology that is a STIX-compliant schema to represent, integrate and enhance knowledge about cyber threat intelligence. Current extensions to it help linking standard cyber kill chain phases to various host and network behaviors that are detected by traditional sensors like Snort and monitoring agents. Unlike our previous work, these extensions allow our framework to assimilate incomplete text from sources so that cybersecurity events can be detected in a cognitive manner.

IV. Cognitive Approach to Cybersecurity

This section describes our approach to detect cybersecurity attacks. It is inspired by the cognitive process used by humans to assimilate diverse knowledge. Oxford dictionary defines cognition [7] as *"the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses"*. Our cognitive strategy involves acquiring knowledge and data from various intelligence sources and combining them into an existing knowledge graph that is already populated with cyber threat intelligence data about attack patterns, previous attacks, tools used for attacks, indicators, etc. This is then used to reason over the data from multiple traditional and non-traditional sensors to detect and predict cybersecurity events.

A novel feature of our framework is its ability to assimilate information from dynamic textual sources and combine it with malware behavioral information, detecting known and unknown attacks. The main challenge with the textual sources is that they are meant for human consumption and the information can be incomplete. Moreover, the text is tailored to a specific audience who already have some knowledge about the topic. For instance, if the target audience of an article is a security analyst, the line *"Wannacry is a new ransomware."* carries more semantic meaning than the text itself. Based on their background knowledge, a security analyst can expand the previous description and infer the following actions that *Wannacry* may perform:

- *Wannacry* tries to encrypt sensitive files;
- A downloaded program may have initiated the encryption;
- Either downloaded keys or randomly generated keys are used for encryption; and
- *Wannacry* modifies many sensitive files.

However, a machine cannot infer this knowledge from the text alone. Our cognitive approach addresses this issue by integrating the experiences or security threat concepts (attacks patterns, the actions performed and associated information like source and target of attack) in a knowledge graph, and combining it with new and potentially incomplete textual knowledge using standard reasoning techniques.

To address the challenge of structurally storing and processing such knowledge about the cybersecurity domain, we use the intrusion kill chain, a general pattern observed in most cybersecurity attacks. Hutchins et al. [14] described an intrusion kill chain with the following seven steps.

- **Reconnaissance:** Gathering information about the target and various existing attacks (e.g., port scanning, collecting public information on hardware/software used, etc.)
- **Weaponization:** Combining a specific trojan (software to provide remote access to a victim machine) with an exploit (software to get first unauthorized access to the victim machine, often exploiting vulnerabilities). Trojans and exploits are chosen taking the knowledge from the reconnaissance stage into consideration.

- **Delivery:** Deliver the weaponized payload to the victim machine. (e.g., email attachments, removable media, HTML pages, etc.)
- **Exploitation:** Execution of the weaponized payload on the victim machine.
- **Installation:** Once the exploitation is successful, the attacker gains easier access to victim machine by installing the trojan attached.
- **Command and Control (C2):** The trojan installed on the victim machine can connect to a Command and Control machine and get ready to receive various commands to be executed on the victim machine. Often APTs use such a strategy.
- **Actions on Objectives:** The final step is to carry out different malicious actions on the victim machine. For example, a ransomware starts searching and encrypting sensitive files while data ex-filtration attacks send sensitive information to the attackers.

Many attacks conform to these seven steps. Hence, we represent the steps in a knowledge graph and link them to related information like potential tools and techniques used in each step, indicators from traditional sensors which detects them and so on. For example, we associate the tool *nmap* with the reconnaissance step and when its presence is detected by traditional network detectors like Snort, we infer a potential reconnaissance step.

A well-populated knowledge graph links many concepts and standard deductive reasoning techniques can be used for inference. Such reasoning over the knowledge-graph and network data can find other steps in the cyber kill chain, if they are present, similar to a human analyst. It should be pointed out that not all attacks apply all seven steps during their lifetime. For example, some attacks are self-contained such that there is no requirement of a command and control setup. Our system's confidence that an attack is happening increases as more indicators are inferred.

There are many other advantages of representing cybersecurity attack information around a cyber kill chain. First, it helps easily assimilate information from textual sources into the knowledge graph. For example, the same exploit *Eternal Blue* is detected in the Weaponization stage for major attacks like those of *Wannacry*, *NotPetya* and *Retefe*. Let us assume that the knowledge graph already has information about *Eternal Blue*, perhaps because it was added as a part of a previous attack. Now with the new information that *NotPetya* uses *Eternal Blue* for exploitation, several things can be inferred, such as the indicators that give evidence for *NotPetya*'s activities even if they are not explicitly specified in the graph.

Another advantage is that it helps in detecting variations of existing attacks. To evade attacks, attackers often employ different tools that can perform similar activities. For example, if there is a signature that specifies *nmap* is used in an attack, adversaries may try to evade detection by using another scanner, like *Angry IP Scanner*[7] *Solar Winds*[8]. Our technique will still detect a reconnaissance with the help of other indicators – the graph links *nmap* to these tools as their purpose is similar – that helps to reduce evasive tactics.

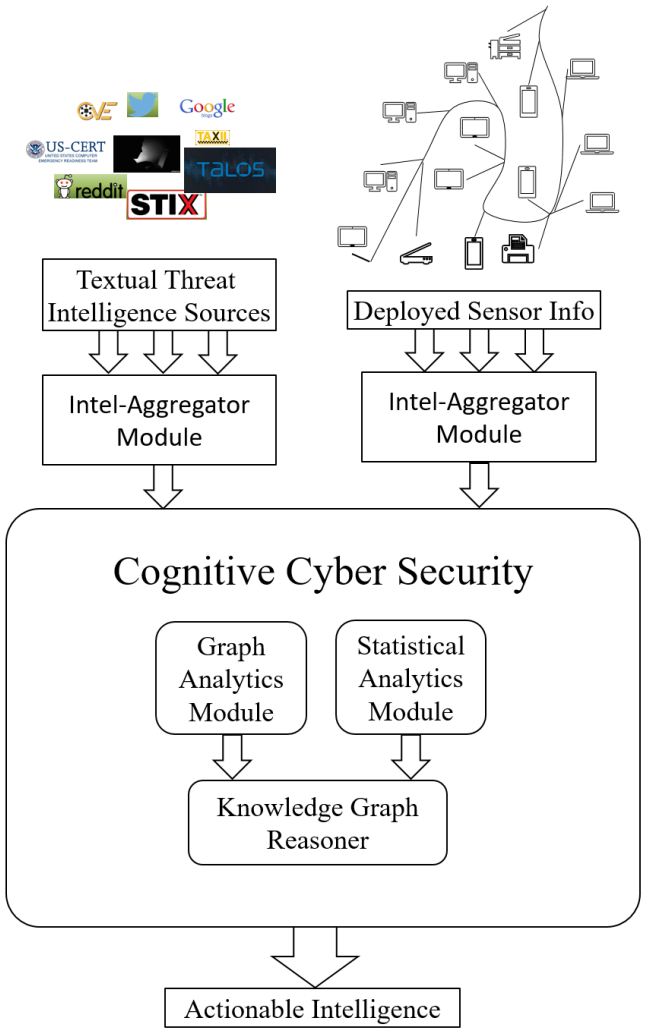

Fig. 1. Cognitive CyberSecurity Architecture contains modules that process different kinds of data, storing it in a structured representation and reasoning over it. The broad patterns or rules to detect attacks are defined by security experts.

Moreover, some new attacks are permutations of tools/techniques used in older ones. There are many situations where similar tools are used, or vulnerabilities are exploited in different attacks. For instance, *Petya* and *NotPetya* share similar Action-on-Objectives (encryption of MFT). The former uses phishing to spread while the latter uses *Eternal Blue*. Since we combine information about different attacks and fuse it with textual information, we can also detect such new attacks.

A detailed example of our inference approach is presented here. Let us assume that a blog reported a new ransomware

[7]https://angryip.org/
[8]https://www.solarwinds.com/

that uses *nmap* for reconnaissance and *Eternal Blue* for exploitation. Our knowledge graph is already populated with common information that includes "*Eternal Blue* uses malformed SMB packets for exploitation", "a generic ransomware modifies sensitive files", "ransomware increases the processor utilization" and so on. Let's also consider that our sensors detected sensitive file modifications, malformed SMB packets, and a nmap port scan. This data independently cannot detect the presence of a ransomware attack with high confidence because they may occur for other reasons such as files being modified by a user or incorrect SMB packets transmitted due to a bad network. However, when we process the information from a source like a blog that *a new ransomware uses Eternal Blue for exploitation*, it provides the missing piece of a jigsaw puzzle that indicates the presence of an attack with better confidence.

Fig. 2. STIX representation of *Wannacry* Ransomware

A. Attack Model

To constrain the system, we make some assumptions about the attacker. First, the attacker does not have complete inside knowledge of the system being attacked. This implies that the person performs some probing or reconnaissance. The second assumption is that not all attacks are completely new. Attackers reuse published (in security blogs, dark market, etc.) vulnerabilities in software/systems to perform different malicious activities like Denial-of-Service (DoS) attacks, data ex-filtration or unauthorized access. Finally, we assume that our framework has enough traditional sensors to detect basic behaviors in networks (NIDS) and hosts (HIDS).

We categorize attackers into three categories that differ in their knowledge and sophistication: script kiddies, intermediate and advanced state actors. Often, script kiddies use well-known existing techniques and tools and try to execute simple permutations of known approaches to perform intrusions. On the other hand, intermediate attackers modify known attacks or tools significantly and try to evade direct detection, but attack behaviors remain generally the same. Adversaries that are state actors or experts mine new vulnerabilities and design zero-day attacks. Our system tries to defend effectively against the first two categories. It is difficult to defend against the third category of attackers until information about these attacks is added to the knowledge graph.

V. System Architecture

In this paper, we describe a cybersecurity cognitive assistant to detect cybersecurity events by amalgamating information from traditional sensors, dynamic online textual sources and knowledge graphs. The system architecture of our cybersecurity cognitive assistant is shown in Figure 1. There are three major input sources to our framework: dynamic information from textual sources, traditional sensors, and human experts. The Intel-Aggregate module captures information from blogs, websites and even social media, and converts them to semantic web RDF representation. The data is then delivered to the CCS (Cognitive CyberSecurity) module which is the brain of our framework where actionable intelligence is generated to assist the security analyst. The various components are described in the following sections.

A. CCS Framework Inputs

The first input is from textual sources. This input can either be structured information in formats like STIX, TAXII, etc. (from threat intelligence sources like US-CERT and Talos) or plain text from sources like blogs, twitter, Reddit posts, and dark-web posts. Part of a sample threat intelligence in STIX format shared by US-CERT on *wannacry* is presented in Figure 2. We use an off-the-shelf Named-Entity Recognizer (NER) trained on cybersecurity text from Joshi et al. [17] for extracting entities from plain text. The next input is from traditional network sensors (Snort, Bro, etc.) and host sensors (Host intrusion detection systems, file monitoring modules, process monitoring modules, firewalls, etc.). We use the logs from these sensors as input to our system. Finally, human experts can define specific rules to detect complex behaviors or complex attacks. Input from human experts is vital because an analyst's intuitions are often used for identifying potential intrusions. Capturing such intuitions makes our framework better. Moreover, analysts can specify standard policies being used in the organization. For example, an analyst can specify if a white-listing policy (i.e. only IP addresses from a specific list are accepted by default) is enforced or not, should IP addresses with geo-location corresponding to specific locations be considered spurious, etc. All these inputs are then sent to the Intel-Aggregate module for further processing.

B. Cognitive CyberSecurity Module

The Cognitive CyberSecurity (CCS) module aggregates and infers cybersecurity events using different inputs to the system and is considered as the brain of our framework. The outputs of this module are actionable intelligence for the security analyst. The core of this module is knowledge representation. We use an extension of UCO (Unified CyberSecurity Ontology) using a W3C standard OWL format to represent knowledge in the domain. We extend UCO such that it can reason over the inputs from various network sensors like Snort, IDS, etc. and

information from the cyber-kill chain. We also use SWRL (Semantic Web Rule Language) to specify rules between entities. For instance, SWRL rules are used to specify that an attack would be detected if different stages in the kill chain are identified for a specific IP address. The information in the knowledge graph is general such that experts can easily add new knowledge to it. They can directly use different known techniques as indicators because our knowledge graph already has knowledge on how to detect them (either directly from sensors or using complex analysis of these sensors). For example, an expert can mention port scan as an indicator and the reasoner will automatically infer that Snort can detect it and looks for Snort alerts.

The statistical analytics and graph analytics sub-modules check for anomalous events in the data stream by building association rules between events detected by a sensor and then clustering them to analyze trajectory patterns of events generated in the stream. Also, a hidden markov model (HMM) is used to learn the pattern (from existing annotated data) and isolate patterns in the stream that are similar. Any standard technique can be utilized to generate indicators as long as they are in the form of standard OWL triplets that can be fed to the CCS knowledge graph. An RDF/OWL reasoner (like JENA [4]) is also part of the CCS module that will reason over the knowledge graph generating actionable intelligence. A concrete proof of concept implementation for this model is described in Section V-D.

C. Intel-Aggregate Module

The Intel-Aggregate (IA) Module is responsible for the conversion of various traditional and non-traditional network sensor inputs to the standard semantic web OWL format. Various inputs to our system are mentioned in Section V-A. However, they will produce outputs in different formats and will be incompatible with our framework's knowledge graph (represented using UCO). To be consistent with entities and classes defined in UCO, the data need to be transformed. This module takes in all inputs to the cognitive framework, maps them to UCO classes and generates their corresponding well-formed OWL statements. The IA module is part of all the sensors which are attached to the framework.

D. Proof of Concept Implementation

Our proof of concept cybersecurity cognitive assistant has an architecture similar to real-world systems like Symantec's *Data Center Security* and *Crowd Strike*. Its master node is the CCS module, which detects various cybersecurity events and coordinates with and manages a configurable set of cognitive agents which run on host systems collecting various statistics, as shown in Figure 3. The cognitive agents that are run in different systems report their respective states to the CCS module which integrates the information and draws inferences about possible cybersecurity events.

1) Cognitive Agents: A full agent is a combination of the Intel-Aggregator (IA) module and a traditional sensor. The Intel-Aggregator module is developed in such a way that it can be customized to work with multiple traditional sensors collecting and sending information to the CCS module as RDF data supported by the UCO schema for further processing. The experimental implementation included process monitoring and file monitoring agents and a Snort agent as described below.

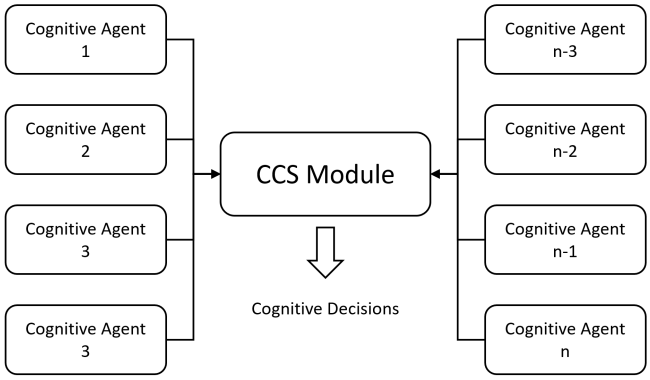

Fig. 3. In our CCS proof of concept architecture each agent is responsible for processing data from a specific sensor / aggregator.

- *Process Monitoring Agent:* This agent combines a custom process monitor and an IA module, and runs on all host machines in the network. It monitors different processes in the machine, their parent hierarchy while gathering statistics like memory and CPU usage. It also notes the files that are being accessed and/or modified by the process, paying special attention to restricted files. The agent converts all this information into RDF data using the IA module and reports them to the CCS module. We implement our process monitoring tool with the Python psutil [32] module. The monitor maintains its own process table (list of all processes in the system) so that it can monitor the state of each process, i.e., if it has been created or is running or it has exited.
- *File Monitoring Agent:* A custom file monitor is attached to an IA module that is similar to a process monitoring agent and also runs on all host machines aggregating various file-related statistics. To avoid monitoring all files and directories, we maintain a list of sensitive ones when detecting suspicious files. Suspicious files are new files created or modified by a new process, large files that are downloaded from the Internet or files copied from mass storage devices. Information sent to the CCS module includes the process that modified the file, size, how it was created, and other file meta-data. The file monitoring agent is implemented in Python using the Watchdog observer library that allows us to monitor all file operations on sensitive directories. Similar to the processing monitoring agent, it converts the file operation information to an RDF representation and reports them to the CCS module.
- *Snort Agent:* This agent is a combination of a Snort log processor and an IA module. It reads Snort's output log file and generates RDF triples consistent with the CCS

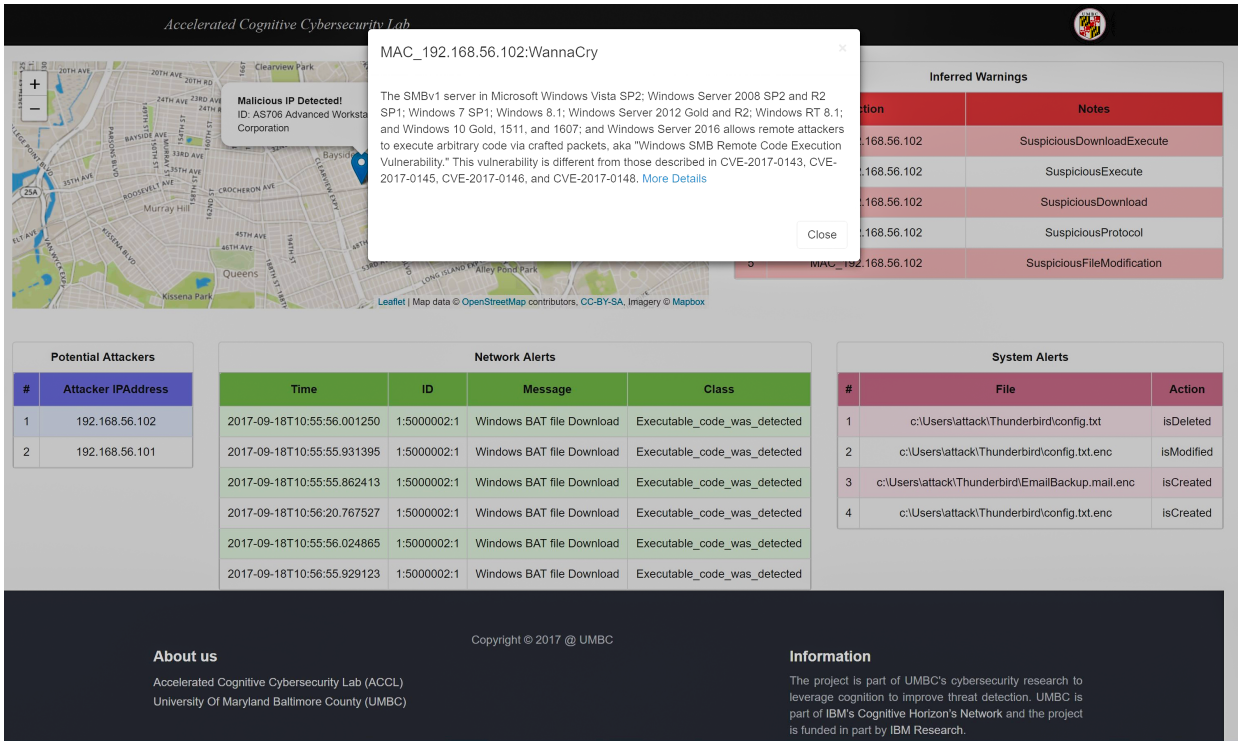

Fig. 4. The CCS Dashboard's sections provide information on sources and targets of network events, file operations monitored and sub-events that are part of the APT kill chain. An alert is generated when a likely complete APT is detected after reasoning over events.

module's knowledge graph.

2) CCS Module: The CCS module is the brain of our approach which uses cognitive analytics to detect attacks using information from agents. It uses an Apache Fuseki server [10] configured with a SWRL rule engine and Jena reasoner. This module gets inputs from all of the agents, performs deductive reasoning over them and feeds output to a dashboard Web interface. The dashboard dynamically displays the information such as source and target IPs, detected activities from sensors and complex events that are observed. When a full-scale attack is detected, the dashboard raises attack alerts as shown in Figure 4.

VI. Use Case Scenario

We tested the effectiveness of our system with a concrete use case of a ransomware attack. We describe the custom ransomware attack designed for the use case scenario, the network topology used in the test, the series of steps taken by the cognitive assistant while detecting the attack, and our evaluation of our cognitive assistant.

A. Custom Ransomware Design

Our custom ransomware targets Windows 7 machines which have the CVE-2017-0143 vulnerability [6], a buffer overflow related to SMB protocol. We use the exploit from Metasploit for this CVE to get access to the victim machine. Once the custom ransomware gains access, it downloads the malware script from the attacker's machine to the victim machine. The downloaded malware script then performs the following steps.

1) Download an executable to encrypt files;
2) Download a public key from the attacker machine;
3) Discover the sensitive files and folders in the victim's machine using a compiled list of potential locations such as the Default Thunderbird email client storage location, default Outlook location, documents folder and default downloads folder, and avoiding system files, since encrypting them may hamper booting;
4) Split the selected files into chunks and generate a random key for each chunk;
5) Encrypt files from each chunk using AES (chosen because RSA implementations are not normally used to encrypt large files) and the corresponding random key;
6) Securely delete raw files corresponding to the encrypted data files;
7) Create a file with all of the encrypted file locations and corresponding random keys used for their encryption;
8) Encrypt this newly generated file using RSA and the downloaded attacker public key; and
9) Delete the raw text file with chunk info securely;

B. Proof of Concept Network Architecture

To deploy our system and infect it using the custom ransomware described in section VI-A, we created a network with three different machines, as shown in Figure 5.

1) Attack Machine: The attack machine is an Ubuntu 16 loaded with custom scripts and a webserver. The attack script is responsible for scanning the network for vulnerable

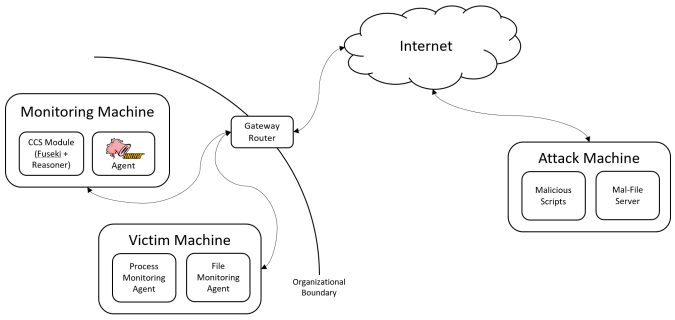

Fig. 5. Proof of Concept Network Architecture

machines and identifying their IP addresses. Once the IP list is compiled, it begins the attack by sending malformed SMB packets and get access to the victim machine. The next step is to download the ransomware script from the attack machine to the victim machine and start it executing. The machine will also run a webserver, which hosts the ransomware script, encryption software, etc. along with a mechanism to generate, send and save public-private key pairs for each of the requested IP addresses.

2) Victim Machine: Our evaluation uses an exploit for CVE-2017-0143, as described in section VI-A, which targets Windows 7 machines. Hence, we choose a fresh installation of Windows 7 SP1 as the victim machine. The only additional software we installed on it were the file monitoring and process monitoring agents described in Section V-D1. We also added "valuable" files into folders like *Documents* and *Pictures*.

3) CCS Master Machine: The core detection techniques are installed on the CCS master machine, which runs the two major components. The first is a Fuseki server loaded with Jena, a standard OWL DL reasoner, the modified UCO Ontology and related SWRL rules. The second is the CCS module, which extracts new information about host machine activities and possible new attacks, runs the analysis, and dynamically updates the CCS dashboard. In addition, we run Snort and a Snort agent in this machine, though Snort can also be run on another machine since the snort agent will take care of sending its information to the CCS Master module.

C. Proof of Concept Timeline

We implemented the associated modules of CCS and created a network described in Section VI-B. Our next goal is to check if we can detect the ransomware attack or not. Our knowledge graph is updated with common knowledge, including the cyber-kill chain and knowledge mentioned in section IV about cyber-attacks. We demonstrate that even such simple information could be used for detecting newer attacks using this experimental system. Figure 6 shows the timeline of the attack performed and the actions from our CCS module. Each step in it is detailed below.

- **Step 1:** Attacker performs a port scan on the victim machine using Nmap;
- **Step 2:** Snort agent detects port scan and reports to the CCS module;
- **Step 3:** Attacker uses the attack script to exploit the victim machine (using "Eternal Blue")
- **Step 4:** Snort agent detects malformed SMB packets in the network;
- **Step 5:** On successful exploitation, the attacker injects malware into the victim machine;
- **Step 6:** The first attack script starts running the malware from the victim machine
- **Step 7:** As described in section VI-A, the malware now initiates downloads of encryption software, keys, etc.;
- **Step 8:** Encryption software and keys get downloaded into the victim machine. The next task from the malware is the detection of sensitive files and their encryption using the downloaded tool;
- **Step 9:** Snort detects downloads from unknown / potentially bad IP addresses;
- **Step 10:** The file monitoring agent detects new files downloaded from the Internet;
- **Step 11:** While performing encryption, the malware modifies many sensitive files and the file monitoring agent reports it to the CCS module; and
- **Step 12:** When encryption is performed on larger files, the processor usage showed larger values and the process monitoring agent reports it to the CCS module.

In this test, the CCS Knowledge graph has the information about a new ransomware attack from textual sources. The new information from textual sources are *"Wannacry is a ransomware"* and *"Wannacry uses malformed SMB packets to exploit"*. In the attack timeline, at Step 2, Snort reports a port scan which will be inferred by the CCS module as a potential reconnaissance step. At step 4, when Snort detects some mal-formed packets in the network, it is not conclusive to tell that it is an attack as it might just be some error packets.

Subsequent steps, Steps 9 through 12, detect downloads from unknown sources, sensitive file modification and increased processor usage. From the knowledge graph, we know the typical characteristics and indicators of a ransomware attack's "Action-on-objective" include multiple instances of sensitive file modification accompanied by high processor usage caused by encryption.

However, these can also occur because of normal usage. For example, the user may have manually modified the files, downloaded and installed new applications that make Internet connections, and then run them. The presence of these indicators taken alone cannot be used to reliably detect a ransomware. However, the CCS system already knows about a new ransomware attack using malformed SMB packets from textual sources. This information when combined with data from various sensors, the CCS system infers that an attack similar to *Wannacry* attack is happening in the system and it is displayed on the dashboard as shown in Figure 4.

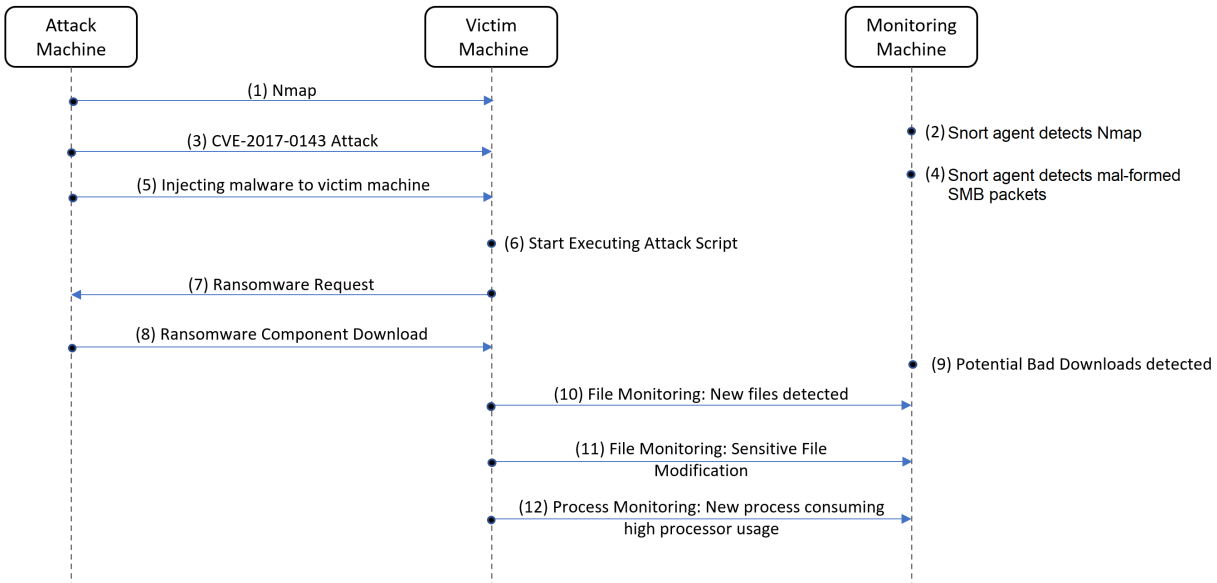

Fig. 6. Timeline of the proof of concept ransomware attack showing each step being executed with its source and target.

VII. Conclusion

In this paper, we described the design and implementation of a collaborative cognitive assistant to detect cybersecurity events and attacks. Our technique assimilates and interprets often incomplete textual information from a variety of sources such as security bulletins, CVE's and blogs, and represents it as a knowledge graph using terms from the Unified Cybersecurity Ontology. It represents the data from traditional host and network sensors, as well as any analysis generated by machine learning techniques, in the same knowledge graph. It reasons over this knowledge to detect complex cybersecurity-relevant events and to predict attacks that may be occurring.

We also developed a proof of concept CCS system which features a cognitive dashboard where cybersecurity events are reported to the security analysts. The capability of our system is demonstrated by testing it against a custom built ransomware, that uses the SMB vulnerability to infect victims similar to the infamous *Wannacry* ransomware. Our technique reduces the cognitive load on the analyst to interpret complex events occurring in large enterprises by fusing information from multiple sources and reasoning over it much like a human analyst.

In ongoing work, we are addressing the scalability of our system by adding more sensors and concepts that define the behavior of various processes running on typical networks. We are revising our UCO schema to improve its ability to represent and reason about temporally-qualified data and information, manage and use data provenance, and support STIX version 2.0. We are also implementing a system to add new cyber threat intelligence data to the knowledge graph on a continuous manner from feeds from TAXII servers [28].

Acknowledgment

This research was conducted in the UMBC Accelerated Cognitive Computing Lab (ACCL), which is supported in part by a gift from IBM. We thank the other members of the ACCL Lab for their input in developing this system.

References

[1] Sean Barnum. Standardizing cyber threat intelligence information with the structured threat information expression (stix). *MITRE Corporation*, 11:1–22, 2012.

[2] Noam Ben-Asher, Alessandro Oltramari, Robert F Erbacher, and Cleotilde Gonzalez. Ontology-based adaptive systems of cyber defense. In *STIDS*, pages 34–41, 2015.

[3] Parth Bhatt, Edgar Toshiro Yano, and Per Gustavsson. Towards a framework to detect multi-stage advanced persistent threats attacks. In *Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on*, pages 390–395. IEEE, 2014.

[4] Jeremy J Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *13th International World Wide Web conference, Alternate track papers & posters*, pages 74–83. ACM, 2004.

[5] CISCO. CISCO Talos. https://www.cisco.com/c/en/us/products/security/talos.html, 2017. [Online].

[6] Common Vulnerabilities and Exposures. CVE-2017-0143 Detail. https://nvd.nist.gov/vuln/detail/CVE-2017-0143, 2017. [Online; accessed 2-March-2018].

[7] Oxford Dictionaries. Definition of cognition in English:. https://en.oxforddictionaries.com/definition/cognition.

[8] FireEye. Common vulnerability scoring system. https://www.fireeye.com/content/dam/collateral/en/mtrends-2018.pdf.

[9] First. Common vulnerability scoring system. https://www.first.org/cvss/specification-document.

[10] Apache Software Foundation. Apache Jena Fuseki. https://jena.apache.org/documentation/fuseki2/, 2018.

[11] Gartner. Reviews for Security Information and Event Management (SIEM) Software. https://www.gartner.com/reviews/market/security-information-event-management. [Online; accessed 3-March-2018].

[12] John R. Goodall. STUCCO: A cyber intelligence platform. https://www.ornl.gov/division/projects/stucco, 2017. [Online].

[13] André Grégio, Rodrigo Bonacin, Olga Nabuco, Vitor Monte Afonso, Paulo Lício De Geus, and Mario Jino. Ontology for malware behavior: A core model proposal. In *WETICE Conference (WETICE), 2014 IEEE 23rd International*, pages 453–458. IEEE, 2014.

[14] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[15] IBM. IBM QRadar Security Intelligence Platform. http://www-03.ibm.com/software/products/en/qradar-siem/, 2017. [Online].

[16] J Ingle. Organizing intelligence to respond to network intrusions and attacks. In *Briefing for the DoD Information Assurance Symposium*, 2010.

[17] Arnav Joshi, Ravendar Lal, Tim Finin, and Anupam Joshi. Extracting cybersecurity related linked data from text. In *7th Int. Conf. on Semantic Computing*, pages 252–259. IEEE, 2013.

[18] Frankie Li and A Atlasis. A detailed analysis of an advanced persistent threat malware. *SANS Technology Institute*, 2011.

[19] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, Alessandro Oltramari, and Luc Schneider. The wonderweb library of foundational ontologies. Technical report, Institute of cognitive sciences and technologies, National Research Council, Italy, 2002.

[20] John Mason. Cyber Security Statistics. https://thebestvpn.com/cyber-security-statistics-2018/, 2018. [Online; accessed 2-March-2018].

[21] Peter Mell and Tim Grance. Use of the common vulnerabilities and exposures (cve) vulnerability naming scheme. Technical report, National Institute of Standards and Technology, Computer Security Div., 2002.

[22] Microsoft. Microsoft Security Bulletin MS17-010 - Critical-Security Update for Microsoft Windows SMB Server (4013389). https://docs.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010, 2017. [Online; accessed 2-March-2018].

[23] Mitre. Common attack pattern enumeration and classification. https://capec.mitre.org/.

[24] Mitre. Common configuration enumeration. https://cce.mitre.org/about/index.html.

[25] Mitre. Cyber observable expression. https://cyboxproject.github.io/.

[26] Sumit More, Mary Matthews, Anupam Joshi, and Tim Finin. A knowledge-based approach to intrusion detection modeling. In *Security and Privacy Workshops*, pages 75–81. IEEE, 2012.

[27] Mark Nicolett and Kelly Kavanagh. Magic quadrant for security information and event management. Technical report, Gartner, Dec. 2017.

[28] OASIS. Introduction to TAXII. https://oasis-open.github.io/cti-documentation/taxii/intro.html, 2018.

[29] Leo Obrst, Penny Chase, and Richard Markeloff. Developing an ontology of the cyber security domain. In *STIDS*, pages 49–56, 2012.

[30] A. Oltramari, L. Cranor, R. Walls, and P. McDaniel. Building an ontology of cyber security. In *Conf. on Semantic Technology for Intelligence, Defense and Security*, pages 54–61, 2014.

[31] Alessandro Oltramari, Lorrie Faith Cranor, Robert J Walls, and Patrick McDaniel. Computational ontology of network operations. In *Military Communications Conference, MILCOM 2015-2015 IEEE*, pages 318–323. IEEE, 2015.

[32] Giampaolo Rodola. Psutil packacge: a cross-platform library for retrieving information on running processes and system utilization. https://psutil.readthedocs.io/en/latest/, 2018.

[33] Panda Security. 27% of all recorded malware appeared in 2015. https://www.pandasecurity.com/mediacenter/press-releases/all-recorded-malware-appeared-in-2015/. [Online; accessed 2-March-2018].

[34] A. Sood and R. Enbody. Targeted cyberattacks: a superset of advanced persistent threats. *IEEE Security & Privacy*, 11(1):54–61, 2013.

[35] David Swift. A practical application of sim/sem/siem automating threat identification. *Paper, SANS Infosec Reading Room, The SANS*, 2006.

[36] Zareen Syed, Ankur Padia, Tim Finin, M Lisa Mathews, and Anupam Joshi. Uco: A unified cybersecurity ontology. In *AAAI Workshop: Artificial Intelligence for Cyber Security*, 2016.

[37] Jeffrey Undercoffer, Anupam Joshi, Tim Finin, and John Pinkston. Using DAML+OIL to classify intrusive behaviours. *The Knowledge Engineering Review*, 18(3):221241, 2003.

[38] Jeffrey Undercoffer, John Pinkston, Anupam Joshi, and Tim Finin. A target-centric ontology for intrusion detection. In *IJCAI Workshop on Ontologies and Distributed Systems*, pages 47–58. Morgan Kaufmann, August 2004.

[39] Alex Vovk. How to Overcome SIEM Limitations. https://blog.netwrix.com/2016/03/21/how-to-overcome-siem-limitations/, 2016. [Online; accessed 2-March-2018].

[40] Alex Vovk. Infographics: Common Drawbacks of SIEM Solutions. https://blog.netwrix.com/2016/03/15/infographics-common-\-drawbacks-of-siem-solutions/, 2016. [Online; accessed 2-March-2018].