AIFIS: Artificial Intelligence (AI)-Based Forensic Investigative System

Rami Alnafrani
Cyber Security Engineering Department
George Mason University
Fairfax, VA, USA
ralnafra@gmu.edu

Duminda Wijesekera
Cyber Security Engineering Department
George Mason University
Fairfax, VA, USA
dwijesek@gmu.edu

2022 10th International Symposium on Digital Forensics and Security (ISDFS) | 978-1-6654-9796-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISDFS55398.2022.9800801

***Abstract*— The scope of forensic investigations has recently expanded. Since most Internet of Things (IoT) devices are plug and play and do not have much memory or storage to pre-process data, it is a challenge for forensic investigators to identify and obtain relevant evidence to reconstruct attacks. As a solution, we propose using artificial intelligence (AI)-inspired techniques to automate the forensic analysis process by emulating attacks in the process of identifying and collecting forensic evidence. We used a differentiable inductive logic programming (∂ILP) system to obtain attack emulation information from different sources, such as device- and subsystem-level vulnerabilities gathered by assessing device components in an enterprise network, and to predict potential attacks from previous attacks on similar configurations. Our experimental results showed that the proposed methodology could successfully generate rules that can assist forensic examiners in identifying evidence to emulate attacks without execution.**

Keywords*— *∂ILP, digital forensics, forensic investigation, artificial intelligence, IoT

I. Introduction

Due to the introduction of Mobile Edge Computing (MEC), many cheap wireless devices are getting connected to cloud-based enterprise services. These IoT device (such as surveillance cameras, lighting systems, and sensors) have limited computing power, used many data formats, and depended on services provided by the cloud-based services. Nevertheless, these devices can be exploited to gain easy access to the cloud-based services, install malware bots, cause DDoS attacks, and create unexpected service disruptions. The Mirai botnet is a case in point, where a frustrated college student created a simple malware that exploited unpatched IoT devices, such as IP cameras and home routers, to launch DDoS attacks against some of the service providers all over the world [1].

Given that our previous research addressed Cloud service forensic, investigating exploitation of MEC services and their forensics is the next logical step. Also, the 5G networking involves multiple MEC services. Later, hacktivist groups, such as Anonymous and New World Hackers, have changed the basic attack capabilities and caused large scale service outages, such as the Dyn cyber-attack launched on October 21, 2016. These hacking groups and others have collaborated in targeting many institutions using multiple strategies. Hence, to address cyber forensics in this area, one must understand the motive, tactics and techniques of the attackers, and be prepared to find attack related data and patterns. Therefore, we propose to use MITRE's ATT&CK framework to understand potential attacker capabilities, use data analytics technology to gather known attack patterns, and be prepared to accept ones that have not been seen before on mobile edge services. As cyber forensics is concerned with the identification, collection, examination, and analysis of data obtained during a cyber-attack investigation [2], both these techniques will enhance existing work on MECs.

II. Background

A. Artificial Intelligence and Cyber Forensics

According to Russell and Norvig [3], intelligence is concerned with rational action, and ideally an intelligence agent. The autonomous entity directs its activity toward achieving goals and takes the best possible action in a situation. These agents use scientific methods, such as searching, reasoning, planning, acting, quantifying uncertainty, making decisions, and learning from available knowledge to solve specific problems. Because of the greater use of scientific methods, AI has advanced more rapidly in the last decade, and has found common ground with other disciplines.

AI can be applied to different stages of the forensic investigation process, which include identifying digital evidence, preserving digital evidence to ensure its integrity and continuity, analyzing digital evidence, and presenting the evidence [4]. In other words, AI has been applied to anomaly detection in the following ways: (1) knowledge-based systems that capture a legal expert's understanding of the law can be built to signal unusual cyber behavior; (2) neural networks can be trained to model different users' behaviors so that the unusual user patterns can be signaled or logged, and (3) data mining and machine learning techniques can be used to discover behavior patterns and flag exceptions. The key technique of using AI in digital forensics is formally representing the knowledge so that the reasoning can be applied, and one significant challenge is the clarity of explaining the AI algorithm in the reasoning process.

B. Digital Forensic Analysis Process

The Digital Forensic Research Workshop (DFRWS) released the Investigation Process for Digital Forensic Science that outlined the traditional forensic investigation process shown in Fig. 1, as consisting of six steps: (1) digital evidence identification, (2) preservation, (3) collection, (4) examination, (5) analysis, and (6) presentation [2].

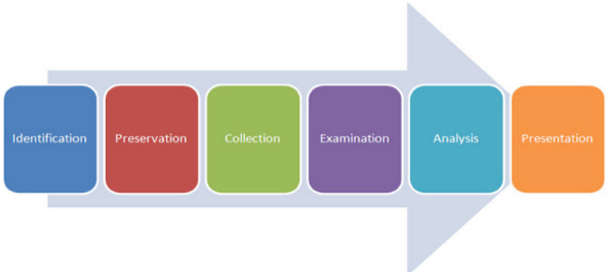

Fig. 1. Digital Forensic Analysis Process

The first digital evidence identification step involves detecting an incident or event. Missing essential evidence at

978-1-6654-9796-1/22/$31.00 ©2022 IEEE

this step affects the rest of the investigation or increase the cost. Identified evidence is preserved to establish the proper chain of custody. The evidence collection step (step 3) uses approved techniques to gather relevant data from the attacked enterprise in order to evaluate them during the examination phase to reduce volumes of the collected data to relevant ones. Finally, the investigators examine the context and content during the analysis phase (step 5) to determine the evidence's relevance and present the most probable attack that fits the evidence in the presentation phase (step 6) by describing the attacks reconstructed by using relevant evidence.

C. *Cyber-Attack Lifecycle Framework of MITRE*

The cyber-attack lifecycle is consisted of seven different phases as shown in Fig. 2. These are: Recon, Weaponize, Deliver, Exploit, Control, Execute, and Maintain. The Reconnaissance phase can be best described as the adversary that can develop a target. The Weaponize phase is when the attack is ready to be executed on the victim's machine or network. The Deliver phase can be defined as what method is used to deliver the vulnerability to the target. The Exploit phase is when the attacker gains access to the target's machine. The Control phase is mechanisms that are used to deal with the victims. The Execute phase is when the attacker achieves their goal by using different techniques to successfully complete the plan. Lastly, the Maintain phase is when the attacker tries to be in the victim's environments for long-term access [5].

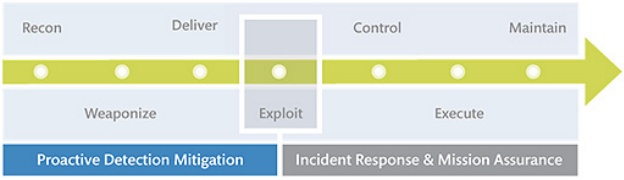

Fig. 2. Cyber-Attack Lifecycle Framework of MITRE

D. *Differentiable Inductive Logic Programming (∂ILP)*

Inductive logic programming (ILP) is a collection of techniques for constructing logic programs from examples. If we have a set of positive examples, and a set of negative examples, an ILP system constructs a logic program that entails all the positive examples, but does not entail any of the negative examples. It is very data efficient since it can learn a program from a small number of examples. Also, it is a human readable program, and it can generalize well outside the training data. However, the main disadvantage of ILP systems is their inability to handle noisy, mislabeled, or ambiguous data. A Neural Program Induction (NPI), on the other hand, learns a procedure for mapping inputs to outputs. The NPI generates the output from the input directly by using a latent representation of the program. The properties of NPI are not very data efficient because they need a massive number of examples before they will learn the right kind of program. They are also uninterpretable; the output is not human readable. They can sometimes generalize well outside of their training data although this is not always true. They are, nevertheless, highly strong in terms of being robust to mislabeled data as well as ambiguous data.

Since none of these two approaches mentioned above are completely data efficient, interpretive, all generalized, robust to mislabeled data, and ambiguous data, there was an urgent need for a new approach that can account for all these properties. Therefore, the idea of ∂ILP has emerged, which stands for differentiable inductive logic programming. It attempts to combine the advantages of the ILP with the advantages of NPI. ∂ILP is a system that learns logic programs, specifically data log programs, from examples. It can efficiently handle the programs which contain robust, noisy, mislabeled, and ambiguous data by giving some input examples and learning a general procedure for transforming inputs into human-readable outputs [6].

III. Previous Work

The work of applying AI to forensic investigation has mainly been focused on two phases: (1) using AI to analyze pre-processed evidence and automate the correlation of the evidence to reconstruct the attack scenario for presentation [7], and (2) using AI to identify evidence from a large amount of data collected after the attack [8-11].

In the logic-based framework, all forensic information that includes evidence, expert knowledge, computer configuration, federal regulations on evidence, security policies, attack techniques (interaction rules), and access control is written to the predicates and reasoned by rules to construct the attack scenario, as shown in Fig. 3. This framework has been useful in automating the evidence correlation and providing an intuitive graphical representation of how the attack happened [7]. However, this framework requires manually collecting and preprocessing evidence, which is the reason why we propose using AI to collect and identify forensic evidence.

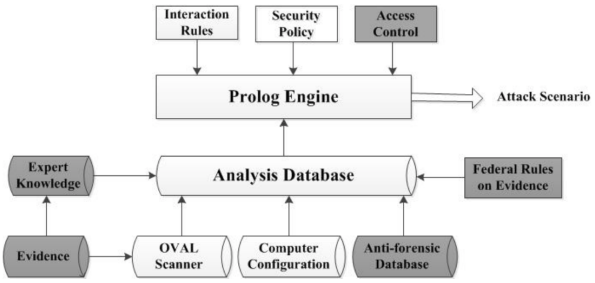

Fig. 3. Logic-Based Framework for Forensic Analysis

Reasoning has been used to correlate evidence to form crime scenarios. In the area of digital forensics, Wang and Daniels [12] described a fuzzy-rule base to correlate attack steps substantiated by aggregated security event alerts. This schema aggregates security event alerts by checking if they have the same source-destination pair, belonging to the same attack class, and fall within a self-extending time window. A self-extending time window is elongated to include all alerts within a predefined time difference to the original event. This work did not provide a good way to resolve the problem when the evidence is missing, nor did it use any standards to determine the acceptability of evidence. In order to solve these limitations, they proposed using an anti-forensics database to implement the expert knowledge, so that it can help generate hypotheses about the missing or destroyed evidence to substantiate an expert's default assumptions [12]. In order to determine the potential acceptability of digital evidence, they also used MITRE's OVAL database and corresponding federal rules on digital evidence. However, both works are only partially automated, which requires forensic investigators to manually collect and identify the evidence.

Researchers have described ways to use AI to help with network forensic analysis. Merkele [11] has investigated using AI to automate the analysis of network-based evidence and

concluded that AI could help reduce the complexity problem of analyzing raw traffic data and the quantity problem of processing the large amount of questionable data. Turner [8] suggested using one of the intelligence techniques, a selective image approach, to deal with the digital evidence bag, a universal container for any source that allows the provenance to be recorded and continuity to be maintained throughout the life of the investigation. Mukkamala, Janoski, and Sung [13] used AI techniques including artificial neural network and support vector machines for offline intrusion detection to identify useful network events, and record minimum representative attributes for each event so that the least amount of information with the highest probable evidence could be stored. Meanwhile, some other researchers have started using AI to automate the attacker planning and simulation in order to find the weakest attack link and enhance the network security [14]. However, to the best of our knowledge, no work has combined automated knowledge inherent in attack planning steps to use them for enriching the forensic analysis effort.

IoT is crucial in virtually all aspects of our lives, from managing our homes to optimizing metropolises through connected transportation and smart city initiatives. However, the complexity introduced by plug and play like IoT devices in fog computing, coupled with the absence of unified security standards for these devices, constrains digital investigation significantly [15]. Forensic investigators struggle to collect digital evidence when these devices are used, creating the need for the scientific community to address the challenges, albeit slowly. Additionally, it is now apparent that the needs of the IoT forensic process are different from those in conventional digital forensic procedures. Reference [16] supports this argument by highlighting the unique potential sources of forensic evidence in IoT environments. The vital assets include the cloud, the web, perimeter devices, networks, and hardware end nodes. Contrarily, Zawoad and Hasan [17] analyze the existing challenges in IoT environments to propose a model to support the underlying forensic investigations. Similarly, Kebande and Ray [15] propose a digital forensic investigation framework for IoT with a primary objective of expanding investigation capabilities with a high degree of certainty. Despite being conducted on a high-level basis, most of these studies are introductory.

IV. Framework Architecture

Our goal of having automated adversary emulation is to provide a framework that can guide forensic investigators on what evidence should be logged for identification. To achieve this goal, we need to emulate the attacks and know the attack paths without executing the attack steps so that the collected path steps can be used to create a surface map and provide a blueprint to look for evidence. We expect this framework to identify evidence efficiently. This process can be done by conducting three main phases, as shown in Fig. 4.

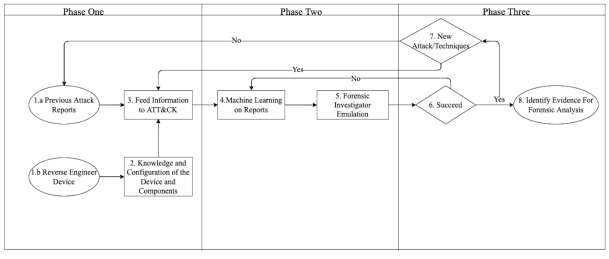

Fig. 4. AI Based Forensic Investigative System

A. Framework Phases

1) Phase one: Gather Forensic Intelligence

In the first step, we reverse-engineered devices that are connected to an enterprise network and combined their configurations to search for potential vulnerabilities in the IoT devices that connect to MECs. We also needed to gather potential attack information from different sources, such as previous attack reports; which are reports generated by well-known cybersecurity companies, potential attacks constructed by using the vulnerabilities of the devices and their configuration in enterprise networks, and potential attacks predicted by using machine learning algorithms based on previous attack knowledge, which will be described in Phase Two.

Once the potential attacks are identified, the other steps of the cyber-attack lifecycle, which include "weaponize, deliver, exploit, control, and execute" will be applied, as shown in Fig. 2 above, to perform attack emulation on an enterprise network to log evidence.

MITRE ATT&CK is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations [2,14,18], and it uses the cyber-attack lifecycle to emulate cyberattacks. Our framework proposes leveraging ATT&CK to perform adversary emulation. Unlike other threat models that were built by analyzing available threat/vulnerability reports, ATT&CK describes behaviors commonly employed by real adversaries as follows: (1) there are public descriptions of attack techniques; how they are leveraged, and why network defenders should pay attention, (2) the platform and data sources can be used to determine what the network defenders should monitor or how to specifically mitigate the attack risks, and (3) all attacking techniques are real-world examples of malware or from threats used by a red team [14]. In this emulation system, attack planning can be described as Definition 1 below:

Definition 1: $\pi i = \langle Pi, A, I, G \rangle$, where the solution to πi is a sequence of attack actions that might change during the actual attacking process (this is practical because the attacker might adjust the attack plan according to the real system configuration); Pi is the set of propositions that the adversary knows exist at time i, though the adversary might not know the true value of the proposition, A is a set of actions enabling a Pa to be true after this action; $I \subset P$ are the starting states of a set of propositions, and G are goal propositions. In this definition, the truth of P is considered static unless the adversary changes it.

By defining the adversary emulation, we propose using logic-based reasoning to reason the network configuration Pi and action I to determine the action outcome. Overall, the first component of the system gathers the intelligence that drives the automated adversary emulation tool in emulating potential attacks.

2) Phase Two: Rule Mining Using ∂ILP

While finding vulnerabilities by using the methodology described in Phase One can help identify the most attacks, it does not cover all potential attacks. To emulate potential zero-day attacks, we propose using machine learning algorithms to predict them [19]. In doing so, we propose using rule mining with ∂ILP to not only mine for predicate instances, but also for rules that are required during the forensic analysis and automated adversary emulation phase. The details are described below.

a) Logic-Based Reasoning to Construct an AI-Based Forensic Investigation System

The syntax of MITRE ATT&CK is shown in Fig. 5, where actions are written as "parameters," propositions are written as "preconditions" representing start states, and "post-conditions" represent post-states after the actions have been executed.

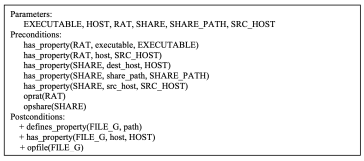

Fig. 5. The Sample Syntax of Definition 1

Rule-based reasoning is used to automate the attack emulation and to perform the corresponding forensic analysis. A rule-based reasoning system is mainly composed of predicates and rules that reason to find a correlation between predicates. In our design, we encoded all facts to predicates, including "pre-conditions" and "post-conditions," as shown in Fig. 5, and encoded actions (we also call them attack techniques in our design) to reasoning rules. An example rule in our design is listed in Fig. 6, which is a horn clause in the form of L1:-L2,..,Ln (meaning L1 is true only when L2, .. Ln are true). In this example rule, post-state L1 is the predicate before ":-"(execCode(Attacker, Host, Priv)), which means the attacker has obtained (privilege Priv on the host), and all pre-states, such as L2, …, Ln, are encoded as predicates after ":-" (we do not explain the details, since the predicates are straightforward). These pre-states include the vulnerability and facts that can be used to exploit the vulnerability, which include the network configuration, security protocol, and attacker privilege. Though the rule simulates the attack action, the predicate name before the ":-" denotes what action has been taken. An attack path is usually composed of several rules that are correlated by matching an attack step's post-state with its pre-state. The reasons why we have proposed using predicates and rules in the form of Fig. 6 are as follows: (1) it is easy to use predicates to encode the data stored by using ontologies and to use rules to encode the attacking technique represented by relations in the ontologies to generate the attack path, which can also be easily loaded into a Prolog environment and executed (we propose to use XSB [20] as the logic engine); and (2) previous work has constructed rules to reconstruct the evidence graph by using evidence that is written to the predicates [7]. This makes it easy to join the two together.

```
execCode(Host, Priv) :-
        vulExists(Host, VulID, Program).
        vulProperty(VulID, range, consequences).
        networkService(Host, Program, Protocol, Priv).
```

Fig. 6. An Example Interaction Rule Composed of Predicates

b) Using Deep Learning to Mine for Rules

There are multiple reasons for using machine learning techniques. The first is that they can mine log files for unexpected entries. However, these entries are to be used in our AI-based rule engines. For example, Fig. 5 and 6 show that these rules gather multiple attributes abstracted in predicates. These predicates are used to ascertain whether an attacker could have obtained relevant information from outside sources or used some known techniques but applied them in the current context. We posit that the application would yield some relevant evidence in the log files. Therefore, we propose using a system that mines both facts and rules and also collects probabilities of the applicable rules for the current context.

A good candidate for a system that satisfies all of these requirements is deep neural networks, which can mine for logical rules. First, we briefly describe how this may be done, and then we describe the creation of a prototype. Because the rules that will be mined will be probabilistic, the framework will transform our previous logic-based forensic analysis framework and the ATT&CK framework into probabilistic rules. We chose to use the differentiable inductive logic programming-based framework proposed by Evans and Grefenstette [6] in their article "Learning Explanatory Rules from Noisy Data" on Google Mind. There are multiple reasons for this choice. The first is that to be used as evidence, forensic analysis results need to show why the learned facts and rules matter in the reasoning process. Hence, having a system that mines rules with probabilities and rule chaining is important in explaining claims. Second, the system can withstand a reasonable amount of noise in the primary facts. The last is the fact that this framework will fix our logical (Prolog-like) rules, but they will be mined from ATT&CK databases and application-specific logs.

c) Differentiable Inductive Logic Programming (∂ILP) and Rule Mining

Logic programming uses Horn clauses like A(x1,…xn):- B1(x1,…,xn), …, Bm(x1,…,xn) to specify rules, where A(x1,…,xn) is referred to as the head of each of Bi(x1,…xn), which are referred to as the body predicates with variables (x1,…,xn). An instance of a predicate without variables is said to be a ground instance. By repeatedly applying rules like A(x1,…xn):- B1(x1,…,xn), …, Bm(x1,…,xn), one can infer (derive) all ground instances that are valid in a model by backward chaining. Ideally, given a rule-based approach, one can compute all truth values that can be derived by using backward chaining under the closed-world assumption (i.e., that predicate instances not derivable by this method are false). However, sometimes, the data are noisy, and the rules valid for a given set of facts (i.e., ground instances) may not be known a priori. Hence, there have been attempts to create rules, such as A(x1,…xn):- B1(x1,…,xn), …, Bm(x1,…,xn) from a given set of instances, which becomes the subject of inductive logic programming. This one can derive a rule like A(x1,…xn):- B1(x1,…,xn), …, Bm(x1,…,xn) from the instances available in a data store like a large log file. This leads to two different issues: the first is that suppose for 80% of the attribute n-tuples the rule A(x1,…xn):- B1(x1,…,xn), …, Bm(x1,…,xn) is valid, but for the other 20%, the rule does not hold. As a solution, probabilistic rule mining would generate a rule A(x1,…xn):- B1(x1,…,xn), …, Bm(x1,…,xn) with 80% probability. The second is that if the predicates A, B1,…Bn were not specified, but 80% of the instances of n-ary instances, A(x1,…xn) and B1(x1,…,xn), …, Bm(x1,…,xn) were true, how can the rule A(x1,…xn):- B1(x1,…,xn), …, Bm(x1,…,xn) be derived? The latter is an issue of inductive logic programming. The way this issue is addressed is to specify a rule template (i.e., rule schema) like Predicate(x1,…xn):- Predicate1(x1,…,xn), …, Predicatem (x1,…,xn). Inductive logic programming would find both the instances of predicates Predicate, Predicate1, …, Predicatem, such as A and B1, …, Bm and the instances of variables

(x1,…xn) that match the rule with the probability of such matches. All 1 + m collections of predicate symbols allow for the syntax, and n-ary words over the constants would be reported by the mining algorithm. Thus, this now becomes a more complex (i.e., second-order) data analytics problem.

3) Phase Three: Rules Generating and Validating

After using machine learning algorithms (∂ILP) to mine for rules, the output rules will be examined and validated to make sure these rules are a good candidate, a process that can help forensic examiners identify evidence. If the output results are not correct or meaningful enough, Phase Two with the machine learning algorithms is to be repeated. The output rules can be used to generate attack graphs, which can help examiners log evidence for identification.

B. Experiment Setup

In the experiment, we used the ∂ILP code[1]. The code was written using Python in TensorFlow, a machine learning system that operates on a large scale and in heterogeneous environments. TensorFlow uses data-flow graphs to represent the computation, the shared state, and the operations that mutate that state. It maps the nodes of a data-flow graph across many machines in a cluster and within a machine across multiple computational devices.

We made some modifications to the code itself to be combatable with the experimental design. We ran our experiment using Mason Cluster, which is a primary web server for George Mason University. This was mainly because we needed a large amount of memory. The input[2] of our experiment consisted of three different files written in text, as illustrated in Fig. 9. The first file was for background knowledge, the second file was for positive examples, and the third file was for negative examples. These files can be altered based on the experiment. Then, it was ready to be run in the cluster. After that, the output of the experiment was made available as a text file, as shown in Fig. 9.

V. EVALUATION

We conducted an experiment using our framework to examine its validity. As described above, the goal of the first phase is to gather the intelligence that drives the automated adversary emulation tool to emulate potential attacks. Therefore, we reverse-engineered devices connected to an enterprise network and combined their configurations to search for potential vulnerabilities on the IoT devices. We started by combining device components and configurations to find vulnerabilities.

For example, the left-hand side of Fig. 7 shows a surveillance system through which attackers can access surveillance cameras connected to a router and authenticate users using a webserver, allowing them to visit the video server at the back end. Suppose the attacker's target is to enter the building without alarming surveillance cameras. In this case, the attacker looks for vulnerabilities in the network and tries to create possible attack paths. To emulate potential attacks, we reverse-engineer all connected devices in the network and combined their functionalities and system configurations to list potential attack paths. In our example, wireless video cameras from AWS DeepLens were connected to an MEC. The camera's front and back, shown on the right-hand side of Fig. 7, allow developers to develop an object recognition system based on deep learning. By searching the Internet, we found important hardware and software information about AWS DeepLens. For example, from its developers' guide [21], we learned that its hardware is composed of the following components: (1) CPU: Intel Atom process, (2) memory: 8 GB RAM, (3) OS: Ubuntu OS-16.04 LTS, (4) 16 GB memory built-in expandable storage, and (5) Intel Gen9 Graphics Engine. Its software is mainly composed of two packages: AWS Greengrass core SDK and the device-optimized MXNet package. Using the information on AWSDeepLens' hardware and software, a simple search can yield (a) vulnerability CVE-2018-3639 [22] in the CPU hardware component and (b) vulnerability CVE-2018-1281 [23] in the MXNet software component that can be used to attack the cameras. In addition, because DeepLens uses a deep learning technique to detect abnormal behaviors, a capable attacker can easily fool the camera by providing false training images. Furthermore, any attacker can use the router as a stepping-stone to the cameras.

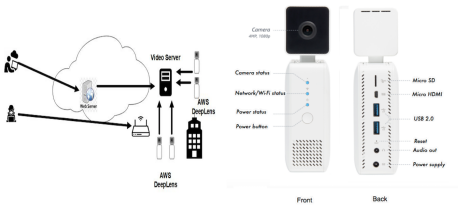

Fig. 7. A simple Surveillance System Created from AWS Deep Lens Cameras

Accordingly, the system was set up to capture and identify relevant data as evidence. Take an example configuration on the AWS DeepLens in Fig. 7: if the pre-state P1 = connect to server on 127.0.0.1, the action A = attackers' listening on 0.0.0.0, and post-state G = sniffing the users' data, the attacker would succeed if he/she takes action to listen on 0.0.0.0 since the vulnerability CVE-2018-1281 [23] would enable him/her to listen on 0.0.0.0.

After completing Phase One and collecting all the important information regarding this IoT device and network environment, we moved to Phase Two, which handles constructing the logic rules. Fig. 8 illustrates the constructed rules for the AWS DeepLens camera.

```
/* configuration information of AWSDeep */
networkServiceInfo(AWSDeep, Intel, tcp, user).
vulExists(AWSDeep, 'CVE-2018-3639', Intel).
vulProperty('CVE-2018-3639', remoteExploit,SSP).

/* configuration information of AWSDeep */
networkServiceInfo(AWSDeep, MXNet, tcp, user).
vulExists(AWSDeep, 'CVE-2018-1281', MXNet ).
vulProperty('CVE-2018-1281', remoteExploit,privEscalation).
```

Fig. 8. Constructed Rule for The AWS Deep Lens Camera

After that, we used ∂ILP to mine these rules to generate new ones. ∂ILP has some constraints. For example, the predicates should be arity of two. To mine for rules using ∂ILP, we need to transfer our constructed rules to be ∂ILP-friendly predicates since our initial constructed rules have multiple parameters and the ∂ILP are restricted to predicates with arity of two. Therefore, we transformed predicates to be

[1] The source code is available at: https://github.com/ai-systems/DILP-Core
[2] The experiment input is available at: https://github.com/ralnaf/AIFIS-Artificial-Intelligence-AI--Based-Forensic-Investigative-System

compatible with the ∂ILP restrictions, transferring the input predicates to be ∂ILP-friendly predicates. Following that, we ran our experiment, as illustrated in Fig. 9, which shows the input and output for the ∂ILP system.

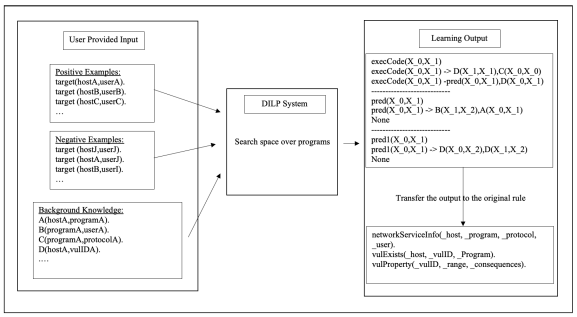

Fig. 9. The ∂ILP Structure for The Experiment

The last phase is responsible for generating rules. The purpose of this experiment was to learn the predicate of executing code, which represents the target predicates (the "user-provided input" section in Fig. 9). In the background knowledge file, we included all the facts about this example that represent the configurations, such as the rules in Fig. 8, which illustrate the network configuration, vulnerabilities in predicates, and others. One of the solutions that the ∂ILP found is shown in the "learning output" section in Fig. 9. This output showed a rule that can be used by the investigator for visual representations, such as attack graphs. This will help them identify where the evidence can be logged based on the new given rule. Overall, this experiment showed us that we can use a machine learning model, such as ∂ILP, to generate rules that can help forensic examiners identify evidence.

VI. Conclusion and Future Work

This paper presented a framework that can aid forensic investigators in determining what evidence should be logged for identification without executing an attack. To do this, we constructed our framework as a roadmap for forensic examiners. We utilized the ∂ILP to generate new rules, which can also be used later as a visualization technique in generating the attack path in the form of a graph. The framework proved its efficiency in generating new rules. As illustrated in Fig. 9, the generated rules demonstrated an example of an attack scenario.

However, there are several limitations to this research that can be addressed in future work. ∂ILP has some structural constraints, such as the number of arity in the predicates. As stated above, ∂ILP requires predicates to have arity of two or fewer. However, some rules have more than two arities in the predicate, especially complicated rules, and breaking them down into two arities is difficult. If we can solve this problem, it will be easier to use ∂ILP and gain better new rules as a result, leading to improved attack scenarios.

In our experiment, we limited the number of atoms in each predicate to two, and we chose the most significant variables to generate the most accurate rules that could later be displayed as an attack graph. However, we believe this is insufficient, and the result would be improved if the user could include more background knowledge and more positive and negative examples. In addition, ∂ILP is very expensive model because it consumes a lot of memory and takes a relatively long time to complete, especially for complex experiments. By overcoming these limitations in the future, forensic investigators will be able to use this advanced technique in a variety of fields since it is advantageous to generate new rules automatically.

References

[1] CloudFlare. (n.d.). What is the Mirai Botnet?. https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/

[2] Palmer, G. (2001), 'A Road Map for Digital Forensic Research', Technical Report DTR-T001-01, DFRWS, Report From the First Digital Forensic Research Workshop (DFRWS).

[3] Russell, S., & Norvig, P. (2016). Artificial intelligence: a modern approach, Pearson Education Limited.

[4] Surat, P. (2019, March 20). AI in Forensic Science. News Medical. https://www.news-medical.net/life-sciences/AI-in-Forensic-Science.aspx

[5] Industry Perspective on Cyber Resiliency. (n.d.). Cyber Attack Lifecycle. MITRE. http://www2.mitre.org/public/industry-perspective/lifecycle.html

[6] Evans, R., & Grefenstette, E. (2018). Learning explanatory rules from noisy data. Journal of Artificial Intelligence Research, 61, 1-64.

[7] Liu, C., Singhal, A., & Wijesekera, D. (2015, January). A logic-based network forensic model for evidence analysis. In IFIP International Conference on Digital Forensics (pp. 129-145). Springer, Cham.

[8] Turner, P. (2005). Unification of digital evidence from disparate sources (digital evidence bags). Digital Investigation, 2(3), 223-228.

[9] Turner, P. (2006). Selective and intelligent imaging using digital evidence bags. digital investigation, 3, 59-64.

[10] Turner, P. (2007). Applying a forensic approach to incident response, network investigation and system administration using Digital Evidence Bags. Digital Investigation, 4(1), 30-35.

[11] Merkle, L. D. (2008, July). Automated network forensics. In Proceedings of the 10th annual conference companion on Genetic and evolutionary computation (pp. 1929-1932).

[12] Wang, W., & Daniels, T. E. (2008). A graph based approach toward network forensics analysis. ACM Transactions on Information and System Security (TISSEC), 12(1), 1-33.

[13] Mukkamala, S., Janoski, G., & Sung, A. (2002, May). Intrusion detection: support vector machines and neural networks. In proceedings of the IEEE International Joint Conference on Neural Networks (ANNIE), St. Louis, MO (pp. 1702-1707).

[14] Miller, D., Alford, R., Applebaum, A., Foster, H., Little, C., & Strom, B. (2018). Automated adversary emulation: A case for planning and acting with unknowns. MITRE CORP MCLEAN VA MCLEAN.

[15] Kebande, V. R., & Ray, I. (2016, August). A generic digital forensic investigation framework for internet of things (iot). In 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud) (pp. 356-362). IEEE.

[16] Oriwoh, E., Jazani, D., Epiphaniou, G., & Sant, P. (2013, October). Internet of things forensics: Challenges and approaches. In 9th IEEE International Conference on Collaborative computing: networking, Applications and Worksharing (pp. 608-615). IEEE.

[17] Zawoad, S., & Hasan, R. (2015, June). Faiot: Towards building a forensics aware eco system for the internet of things. In 2015 IEEE International Conference on Services Computing (pp. 279-284). IEEE.

[18] MITRE. (n.d.). ATT&CK. https://attack.mitre.org/

[19] Alpaydin, E. (2014). Introduction to machine learning. MIT press.

[20] Rao, P., Sagonas, K., Swift, T., Warren, D. S., & Freire, J. (1997, July). XSB: A system for efficiently computing well-founded semantics. In International Conference on Logic Programming and Nonmonotonic Reasoning (pp. 430-440). Springer, Berlin, Heidelberg.

[21] AWS DeepLens. (n.d.). AWS DeepLens Developer Resources. Amazon. https://aws.amazon.com/deeplens/resources/

[22] NIST. (2018, May 22). CVE-2018-3639. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2018-3639

[23] NIST. (2018, June 8). CVE-2018-1281. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2018-1281