Multi-Agent Systems for Scalable Internet of Things Security

Phillip Kendrick
John Moores University
Department of Computer Science
Liverpool, United Kingdom
P.G.Kendrick@2012.ljmu.ac.uk

Martin Randles
John Moores University
Department of Computer Science
Liverpool, United Kingdom
M.J.Randles@ljmu.ac.uk

Abir Hussain
John Moores University
Department of Computer Science
Liverpool, United Kingdom
A.Hussain@ljmu.ac.uk

Natalia Criado
King's College
Department of Informatics
London, United Kingdom
Natalia.Criado@kcl.ac.uk

ABSTRACT

Providing effective and scalable real-time security to Internet of Things devices can be a challenging task given the limited computational capacity of the devices and the amount of network traffic that can be viewed at any given time. Multi-Agent Systems have proven to be a valuable tool within the areas of cyber security, distributed networks and legacy systems because of their scalable and flexible architecture. In this paper we present a novel implementation of a Completely Decentralised Multi-Agent System for use within, or to support, Internet of Things networks through the distributed processing of security events to offload the computational cost of data processing from Internet of Things devices. The concepts of conditions and effects are introduced to allow agents to describe digital evidence found in an abstract language instead of sharing individual pieces of data to mitigate concerns of data leakage in extended networks. Emphasis is placed upon the scalable architecture design allowing domain experts to independently create agents specific to a particular technology or application process which will automatically work with other existing agents without further configuration.

Keywords

Multi-Agent System; Internet of Things; Cyber Security; Scalable Systems

1. INTRODUCTION

A Multi-Agent System (MAS) [27] can be distinguished from traditional software by its distributed and autonomous deployment model. Traditional approaches to cyber security have typically processed the entire contents of a network through a single Intrusion Detection System (IDS) [10, 25, 23]. With the ever increasing amount of traffic flowing through networks, IDSs require expensive and high-performance hardware to manage the computationally expensive task of processing data in real time. With the limited capacity of most Internet of Things (IoT) [26, 16] devices, a more scalable approach is required to deliver the required level of security to devices without the capability to process or store large amounts of data. In this paper, a distributed MAS is proposed to offload the computational cost from the IoT devices to specialised agents located on different networks to perform an in-depth, intelligent and automatic analysis of the network traffic flowing to IoT devices. Computational gains are made by automating techniques commonly used in manual network forensics [2]; by utilising agents that can search for digital evidence intelligently by considering what is already known about an attack and searching for additional information in the areas where it is most likely to be found.

Network forensics is a valuable process most commonly performed manually by trained practitioners who will analyse the cause and spread of an attack after the fact. In this paper, the forensic process is automated and adapted for use within the IoT environment where domain-specific factors such as having a multitude of independently created devices, a variety of different protocols and devices spread across a large network must be considered. By giving agents the tools to perform network forensics autonomously, we will be able to collect and analyse data faster, avoiding problems such as data degradation and concerns about leaking private information. By giving agents the ability to follow one line of investigation over another, when it is supported by previously collected evidence, agents will avoid performing unnecessary and unimportant data collection thereby making the system more efficient than traditional brute force attempts to analyse the entire contents of a given network. These processes will be facilitated through the use of a decentralised communications protocol well suited for the task.

It is uncommon for traditional security systems to take advantage of the points discussed above. Instead, detection normally takes place on a constant stream of network traffic

Cambridge University of Cambridge, London UK

DOI: 00.000/000 0

using either anomaly or signature detection [28]. A more proactive solution that will actively seek to find data on the network even if it wasn't initially collected is needed to ensure a higher level of accuracy and scalability for large distributed networks that do not have a strong central IDS such as in the case of IoT networks.

This paper is organised as follows: Section 2 contains an analysis of current security technologies and issues to consider within the IoT environment. Section 3 contains an explanation of the proposed Multi-Agent architecture. Section 4 describes the implemented simulator built to work with the concepts described throughout this paper. Section 5 lists related research as well as a comparative discussion on the proposed and existing solutions. Finally, Section 6 suggests possible next steps towards developing more scalable Multi-Agent IoT security technologies.

2. SECURITY ANALYSIS OF THE INTERNET OF THINGS

Scalability.

Traditional security models for protecting IoT networks have often included a number of nodes that collect and then deliver data to an IDS for processing. This hierarchical model where all data is sent back to a central location is inefficient in operation and can result in performance bottlenecks when too much data is delivered to the IDS for processing. Furthermore, it is not scalable since the security of the network as a whole is dependant on the capacity of the IDS, introducing more IoT devices onto the network would result in decreased security if the IDS is not also scaled. This limited model which would typically be found in a single organisation implementing a network of IoT devices cannot scale up to even greater levels, for example in a cooperative environment with multiple organisations. Instead, a distributed multi-agent architecture is proposed to support IoT devices by offloading the computational cost from the IoT network to the network of security-focused agents as will be discussed in Section 3).

Data Sharing.

There has been a historic reluctance in sharing network data, especially security data, because of the risk that unintended information (e.g., personal or organisational data) might be leaked. While merging networks together has proven to be a useful measure to take (e.g., in the case of the supply chain network), security has typically been performed independently on each network with minimal data sharing taking place. In the case of multiple interacting IoT networks the benefits of sharing data are even greater where performing computational processing is expensive and sharing conclusions about security events could considerably reduce the overall amount of work that must be done. Devices of similar type and function are more likely to experience similar attacks and so would benefit from being able to communicate that they are under attack, even without disclosing any specific data. A wider example of this in cyber security is an industry wide attack [11] where similar infrastructures are targeted, for example, a malware attack against the banking industry. In examples like this, knowing that there is an increased risk of being targeted and knowing the taxonomy of the attack ahead of time can be useful in preparing for the attack, making data sharing [8] a critical aspect of cyber security.

Network Forensics.

Network forensics [2] is the process of collection and analysis of digital evidence. This process is most commonly performed manually by trained practitioners after a successful cyber breach has occurred in an attempt to understand the event more clearly. Forensic practitioners will typically begin with a more general analysis of the network and then narrow down their search based upon the already collected digital evidence. In this way, network forensics is an iterative process of search and discovery using what is already known to find more evidence. Currently, network forensics is not apart of the average intrusion detection toolkit and is considered to be a separate component to signature and anomaly detection. However, the process itself is efficient [4] in the way evidence is searched for in only those locations that the analyst would expect evidence to be found in, given what is already known, and so would be valuable to the efficiency of an IoT security system by reducing the overall amount of work done by only search in the network locations that are more likely to contain digital evidence.

Domain Expert.

Security achieved through the use of a signature detection [17] requires that a domain expert create the rules for detecting the known attacks. Corporate software will often use a variety of proprietary protocols for data transfer, which as a result are not well supported by open source IDSs such as Snort [23] and Bro [13], two popular open source systems. With the increasing interest of IoT devices, the number of protocols has also steadily increased but widespread support is still lacking. A scalable approach wherein the developers of these protocols can easily and independently define malicious activities is required as a component that can be easily included in the running network. Given the example of network layer security [6], which typically uses sources of threat intelligence [15] (e.g., lists of known malicious IP addresses or a whitelist of allow locations) to perform security, we believe it is desirable to have multiple lightweight agents to monitor these sources and take actions, but also make it easy for developers to create their own agents for protocol-specific monitors.

3. MULTI-AGENT ARCHITECTURE FOR INTERNET OF THINGS

In this section we present an overview of our agent model for the decentralised collection and analysis of cyber security data. Our system is composed of a number of agents ($G = \{g_1, ..., g_i\}$), each capable of performing one data collection and analysis task for a specific service of technology. This information can be formalised by a set of features F representing different attributes or characteristics of a given activity; e.g., the IP address of a given connection, Virtual Private Network (VPN) usage, etc. Each feature ($f \in F$) has a domain (D_f) containing all its possible values. The data collection task entails the agent interacting with a data source which may be a host device, server, log file or any other component from which information can be gathered. These could be located either locally on the network that the agent is protecting or remotely on an internet-connected

server. Each data collection task is designed to only collect one piece of information so that many lightweight agents collecting different piece of data can be created for each source. As the network expands with new technologies being added, additional agents can be added to interact with them. A data analysis task will classify the data that was collected to determine, based on the agent's local signature or anomaly detection database, whether collected data is malicious or innocuous. These two components of data collection and analysis form the basis of the agent's abilities to be able to sense the environment, collect relevant information and analyse it without the need for administrator oversight. Consider the following two definitions:

Definition 1. A data collection action is defined as a tuple C, e in which:

- C is the action conditions; i.e., a set of pairs (f, v) where feature $f \in F$ and value $v \in D_f$;
- $e \in F$ is the action effect; i.e., a feature whose value will be determined by the action.

Definition 2. Given a set of pairs (f, v) representing the available information about a suspicious activity, a data analysis action is defined as a function returning a value between $[0, 1]$ representing the probability of the suspicious activity being malicious.

Each of the agents will also have a set of constraints placed on them which must be satisfied before the agent can perform its data collection and analysis tasks. Each constraint will be a piece of data about the security environment, for example, an agent monitoring a VPN service may hold the constraint that it requires a remote IP address before it may perform its data collection task. The data for this constraint will come from data that another agent collects during its data collection task. For the purposes of this system, the constraints will be called the conditions and the data gathered during the data collection process the effect. Each agent should only perform one data collection task and should produce one effect, however, it may have several conditions depending upon the actual data collection task it is designed to perform.

Definition 3. Given a set I formed by pairs (f, v) representing the available information about a suspicious activity, and a data collection action C, e we define that action conditions are satisfied iff for all $(f, v) \in C$, $(f, v) \in I$.

The concept of an extended data collection task is introduced to describe the process of several agents performing their data collection and analysis tasks together for the purposes of investigating a potential security event. Since the conditions are derived from the effects, when a new effect is gathered during the data collection process, it may satisfy the conditions of other agents, resulting in additional agents being able to perform their data collection and analysis tasks. In addition to participating in extended data collection tasks, each agent will be responsible for the continual monitoring of a data source. If during this continual process some data is flagged as being potentially malicious, the extended data collection process will begin with an agent communicating its effect to any agents who have the effect set as a condition. Figure ?? illustrates this concept using four agents (A1-A4) with each performing a data collection task and communicating the effect discovered during that task to the next agent. The result of the data analysis process is a judgement about the collected or monitored data of either malicious or innocuous, within the system, this is called the local decision.

A communication model allows for the transfer of information between agents, the main use of which involves the sending of a report between agents which is a grouping of the agent's ID, effect and local decision about the maliciousness of the data. The report is generated and then sent to the next agent whose conditions have been satisfied by the effects already known, each agent generates their own report and aggregates it with the reports that it receives. The transfer of the aggregated set of reports facilitates the build up of information within the agent network and is called the extended data collection task. This can be viewed in Figure 1 where data is being collected during stages 1, 3, 5 and 7, the effect gathered from the data source is being analysed by each of the 4 agents and then the aggregated set of reports is being build up and transferred between each agent during stages 2, 4 and 6. Figure 2 shows a similar extended data collection process occurring within our multi-agent simulator, this process can be viewed as connections being made between the agent nodes.

Definition 4. A local report defined as a tuple $g, (f, v), p$ where:

- $g \in G$ is the agent's identity;
- (f, v) is a pair feature value corresponding to the output of the data collection action performed by agent g;
- $p \in [0, 1]$ is the agent's analysis of the suspicious activity; i.e., the probability of the suspicious activity being malicious.

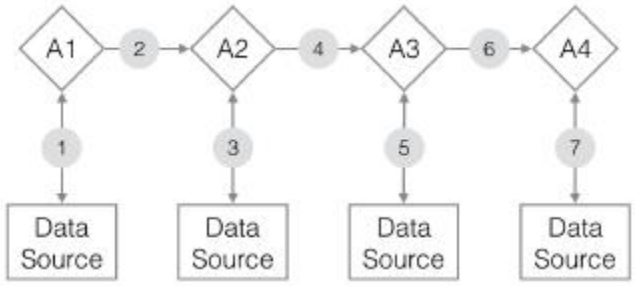

Figure 1: Flow diagram for the extended data collection task using agents (A1-A4) and data sources.

At some point during the extended data collection process there will be no more agents that can participate as

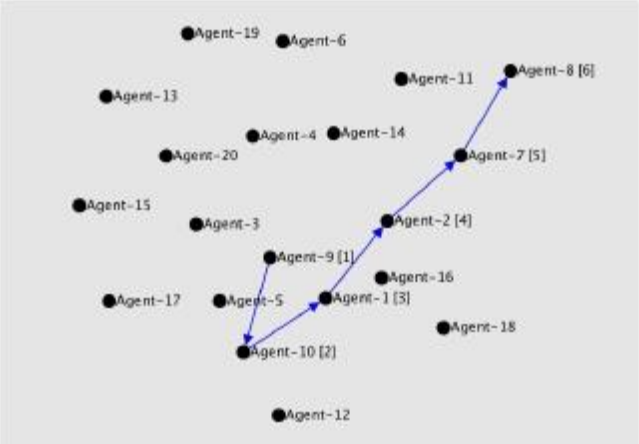

Figure 2: The extended data collection task being performed within the simulator.

their conditions have not been satisfied. At this point the aggregated set of reports will contain a number of IDs, effects and local decisions about the maliciousness of the effect data. Using this set of local decisions, the last agent to receive the set will calculate the final global decision for the participating agents. The final global decision is calculated through the use of a voting system [27] which will use the local decisions (collectively named the global report) to come to a final decision about the security event as a whole. If the global decision returned is "malicious" then the agents will be justified in taking action against the detected attacker knowing that other agents have also detected an attack. The communications protocol is also used to send the results of the global decision to each of the participating agents informing them after it has been made.

Definition 5. A global report R is defined as a set of local reports $\{ g_1, (f_1, v_1), p_1, ..., g_n, (f_n, v_n), p_n \}$ containing the information collected by different agents participating in a same extended data collection process.

Definition 6. Given a global report G representing the local decisions made by the agents participating in an extend data collection process, the global decision is a function returning a value between $[0, 1]$ representing the collective judgement about the maliciousness of the investigated activity.

This section has outlined the basic architecture for the propagation of information through the agent network. Each agent has a condition that must be satisfied before it may take part in the extended data collection and produces and effect based upon its findings. This architecture is desirable for a number of reasons:

Data Sharing.

Within this architecture the decision made by each agent is the important variable that must be shared. Within environments with multiple systems interacting with each other, it is as discussed previously, beneficial to share security data to gain a more holistic view of the network, however, businesses often have concerns about data being shared with potential competitors and so are reluctant to share security data. In the proposed architecture, the decisions can be sent from agent-to-agent without the actual information used to make that decision, making it a viable system for multiple systems (IoT networks belonging to different organisations) to interact and share conclusions about events without disclosing sensitive information. The approach of sharing higher level conclusions about events, for example, that a port scan occurred and it was found to be malicious, is a more secure approach than to share the actual network packet data which could contain sensitive information.

Network Forensics.

This architecture was designed to make use of the forensic process to be more efficient in detecting cyber attacks. With the limited capacity of IoT devices, probing every available devices for information is computationally expensive so the alternative approach of using what is already known to inform where to look next is taken. Agents may only take part in the extended data collection task if all of their conditions are first satisfied. If the information necessary for the agents data collection and analysis task is missing or not yet known, they will not attempt to take part in the investigation until enough evidence has been accumulated to suggest that they are likely to find more evidence.

Domain Expert.

This system makes it easy for the domain expert or device creator to create new agents in the environment without needing to be aware of other existing agents. Each individual agent is seen as an independent entity, with any information required for its analysis of the data to be included (or made aware of its location) upon its creation. The use of conditions and effects allows the agents to assimilate into the agent network by simply fitting into the extended data collection process when its conditions are satisfied.

3.1 Agent Discovery & Communication

To allow agents participating in an extended data collection to coordinate, this model includes an interaction protocol (depicted in Figure 3). The protocol is formed by five main phases: (i) request for participants; (ii) proposals from available participants; (iii) participant selection; (iv) inform summary; (v) inform result, described as follows.

Request for Participants.

Once an agent has performed its data collection and analysis tasks, the agent must add its local report to the global report and then send it onto the next agent for further information collection. The communication module is used to facilitate this.

The first step of the interaction protocol is to request help from other agents that can participate in the data collection process. In particular, the set of pairs feature value are extracted from the global report and then broadcast (by the initiating agent) to the other agents. Given a global report $\{ g_1, (f_1, v_1), p_1, ..., g_n, (f_n, v_n), p_n \}$ a request for participation is formalised as a set $\{(f_1, v_1), ..., (f_n, v_n)\}$ containing the available information about currently known broadcast out to all agents.

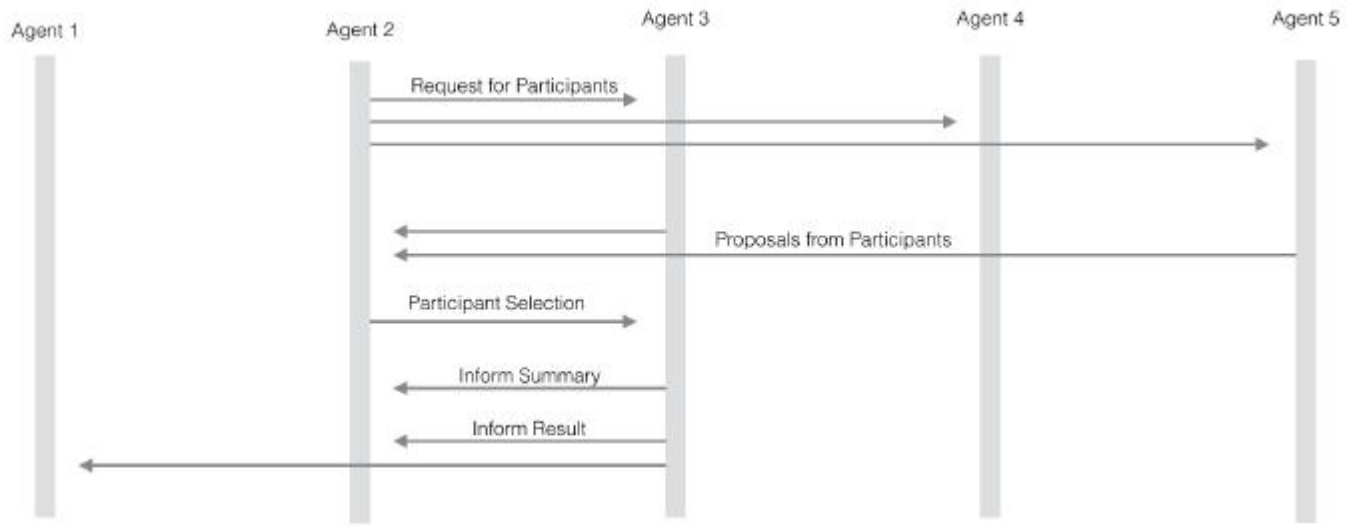

Figure 3: An example information flow between 5 agents using the decentralised communications protocol.

Proposals from Participants.

Any agents whose data collection action is satisfied by the information contained can respond by indicating their availability to participate in the extended data collection task. Figure 3 illustrates this with Agent 2 broadcasting to other agents on the network.

Participant Selection.

It is possible that several agents respond to the initial request indicating that they can work with the available data. The initiator must decide which agent will be selected to continue with the data collection process. In particular, the initiator will send the whole global report to this selected agent.

Unlike in other MAS solutions, the proposed model does not include a central repository of agents which can be queried to find the most suitable agent for a given task. This improves scalability but requires a system to allow agents to find each other. The agents will maintain a local database of agents that they have previously worked with. Deciding which agent should be selected as the preferred agent will affect the performance of the system as a whole. If the most optimal agent is selected for the task most of the time, the search process will improve as less time is spent performing data collection by unreliable agents. There are a number of ways in which the preferred agent can be identified based on what is important in a given situation. If accuracy is especially important for the current event the agents may select the agent that most often votes correctly, this will result in a more accurate search. If time is an important factor during some event the agent may choose the fastest performing agent to collect information quickly, this will produce a result faster than the previous but could potentially result in a less certain decision. While an analysis of factors such as these could be done to determine the optimal preferred agent selection algorithm, events within the security environment can often be unpredictable and allowing the agents to choose the preferred agent at run time could produce a more adaptable solution.

For example, in a Distributed Denial of Service (DDoS) attack, selecting the preferred agent based on the speed at which agents are capable of collecting information could not result in a timely collection if one part of the network is under attack. However, selecting agents based on their geographical location (e.g., using the agents that are not under attack), would improve the efficiency of the response.

Inform Summary.

The agent selected from previous stage (termed the child agent) will send back a summary to the previous agent (termed the parent agent) containing information about the decision it made during its own data collection task, this is done so the parent agent may evaluate the performance of the child agent by comparing its decision to the groups. This process can be viewed in Figure 3 where Agent 3 (the child) sends back a summary to Agent 2 (the parent). This summary will be logged by the parent agent for use in selecting the child agent in future extended data collection tasks. The parameters sent in the summary will include the agents local decision about the maliciousness of the event, as well as, performance variables such as the time taken to perform the collection task, the importance of the data collected and the computational cost of performing the collection.

Inform Result.

Once a final decision has been reached, the final decision will be sent to all of the participating agents, this can then be used by the agents to review its method for selecting the preferred agent (e.g., the preference can be increased for those agents with local decisions in-line with the final decision).

3.2 Remote Agent Communication

The interaction protocol described in this section has detailed how agents on the same network may communicate with each other to find other agents that can participate in the extended data collection task given the information already known. While this protocol works for a single network where messages can be broadcast across the entire network with little cost, networks that are disconnected from

each other across long distances would require an extended discovery protocol for making agent networks aware of each other. Many discovery protocols that perform a similar function exist already, the most commonly known being Domain Name System (DNS) [9] which uses a distributed network of information repositories that can be queried. Any number of similar solutions or alternatively a decentralised discovery protocol could be implemented to allow remote agent networks to link together for the purposes of participating in the extended data collection over the internet but is considered out of scope for this paper.

3.3 Bro IDS for Attacker Profiling

The two fundamental approaches to intrusion detection are to use either signatures of known malicious software or actions (e.g., a threshold for a port scan) or to detect behavioural anomalies (e.g., an unusual number of login attempts from a user). While signature detection is useful for detecting what is already known, the IoT environment is vast and varied with many different types of protocols and systems which makes manually creating signatures for timely detection a challenging task. A more scalable approach is to perform an analysis of higher-level features such as the identity of those interacting with the system, how those users interact with the system and to rely more on behavioural and anomaly analysis.

The Bro IDS [14] is a popular open source research tool for understanding networks. Bro distinguishes itself from other IDSs as its primary function is to understand and make sense of what is being monitored in a policy independent way rather than to determine what is and what is not malicious. Once what is being monitored is well understood and catalogued into a useful format, the behaviour of the users can be studied and malicious actions extracted using either policy scripts or anomaly analysis.

With several policy scripts enabled the ISCX IDS dataset [22] was analysed. The dataset was collected from an IDS on a network and includes both instances of infiltrating the network from inside and normal activity. Bro IDS produces an output file named notice.log containing any activity considered noteworthy by the policy scripts used, examples of noteworthy behaviours include attempted port scans, invalid SSL certificates and other non-standard network activity. To simulate the use of the Bro IDS as part of the proposed multi-agent architecture contents of the file was used to define agents for monitoring specific behaviours Bro can detect, for example, the feature that performs a lookup against SSL certificates is defined as its own agent within the simulator (described in more detail in the Section 4). The concepts of conditions and effects were then applied to the agents so that each of the Bro agents had their corresponding set of conditions and effects to work within the architecture. Adapting Bro IDS to the MAS was done to show how different technologies can be adapted using the concept of conditions and effects to work within the proposed architecture as well as a way to automatically define agents using existing technologies.

3.4 Agent Coalitions

Coalitions are used within the system to increase the speed at which agents can be consulted during the extended data collection task as well as to compensate for a large number of agents outvoting a smaller number when it is not desirable to do so. The most basic algorithm for use within this architecture is to consider the amount of votes cast by all of the participating agents for either malicious or innocuous and make a decision about the event as a whole based upon the total. This method of tallying up votes to decide the overall decision is useful in the detection of less advanced attacks where multiple agents are able to detect the attack from different vantage points on the network and classify it as malicious, but it becomes less useful in the case of advanced stealthy attacks because a large number of unreliable agents may outvote a smaller number of reliable agents. This is often the case in the example of network level threat intelligence lookups. If an IP address is well known for being malicious and multiple threat intelligence vendors have logged it as being so, then detection of the IP address given many votes for it being malicious accurately detects the user. However, if the IP address is unknown to many threat intelligence vendors and there are a large number of agents monitoring a variety of vendors, there will be an overwhelming number of innocuous votes which would outvote a smaller number of agents that have correctly identified the attack in some other area of the network.

Coalitions are used in this instance to reduce outvoting by a large number of agents. The coloured nodes in Figure 4 show a number of coalitions, with one agent randomly selected as the 'representative' of the group. Only the representative will respond to the extended data collection task when its conditions are satisfied, but will instead first communicate the global report containing all the accumulated effects to each of the members of the coalition to receive their individual decisions. This process can be viewed as a separate 1-stage extended data collection task performed locally within the coalition. The votes are collected by the representative and then a single decision for the group is decided based upon the chosen aggregation algorithm, the simplest of which is to take the highest number of votes in either direction of malicious or innocuous. This final group decision is then delivered by the representative as one vote.

Coalitions are locally formed when agents of the same type and function become aware of each other, for example, if several agents on a network all have the same conditions and possible effects, they are functionally the same and so will form a coalition. This most often occurs with agents responsible for monitoring threat intelligence sources, often multiple sources will exist for the same type of information (e.g., file hashes) and each will have a dedicated agent, but they all have the same conditions and same possible effects.

The algorithm used within the coalition to aggregate the decisions will be chosen based upon a number of factors, for example, the importance of the data source; if the threat intelligence source used by the agent is a reliable indicator of compromise, then only one vote for malicious may be required. However, for less reliable sources, a majority vote may be required. The selection of this algorithm may be automated to an extent based upon the consistency of group decisions, for example, if all agents tend to vote in a similar way, then the sources are likely more reliable than if the agents vote inconsistently.

4. SIMULATOR

A simulator was developed to explore the viability of the use of conditions and effects in the IoT environment. Figure 4 shows the simulator with the nodes (representing agents)

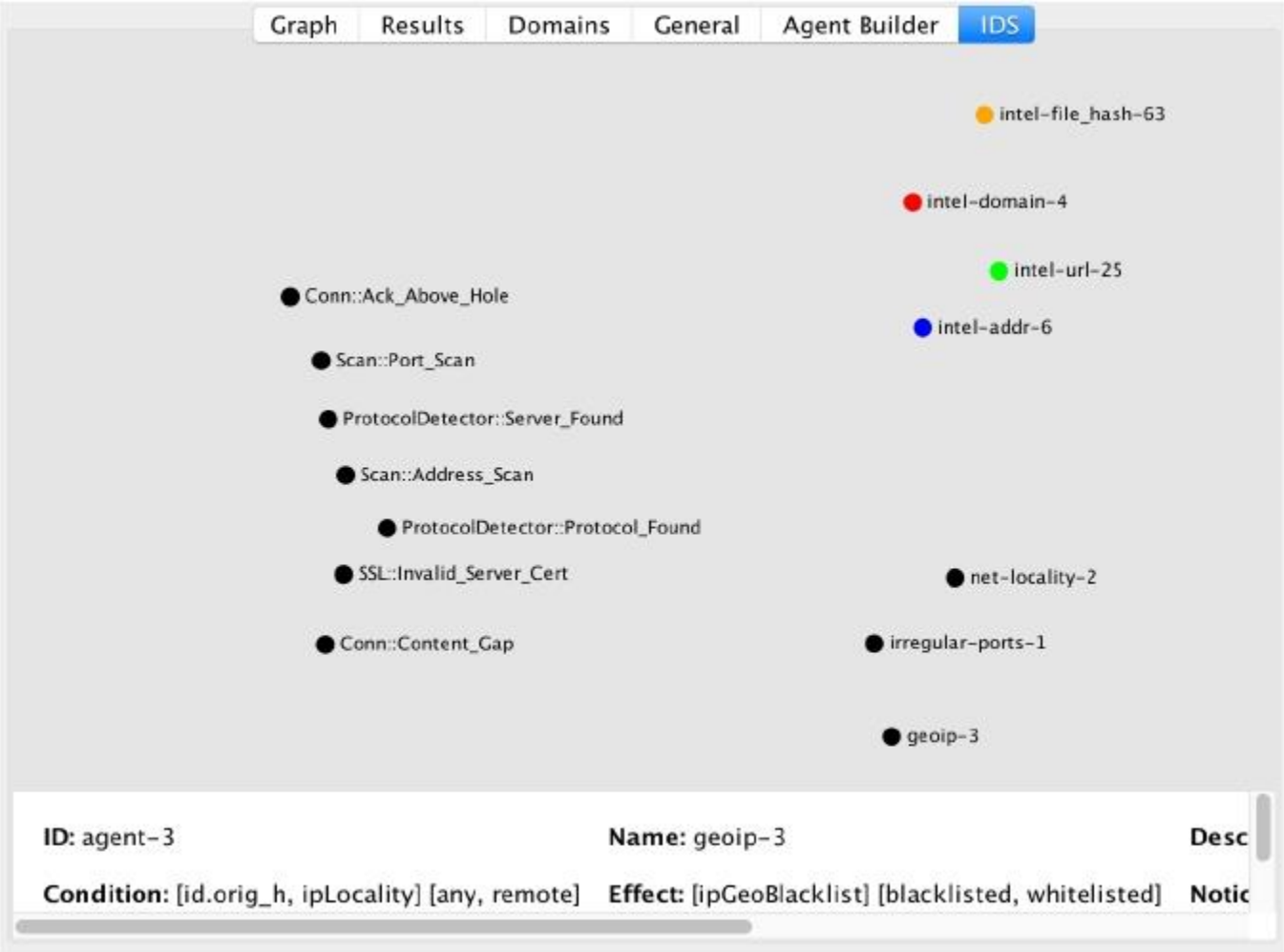

Figure 4: The multi-agent simulator consisting of three groups of agents (Bro IDS, Threat Intelligence, Manual) ordered.

spatially organised into three distinct groups showing the three types of agents used. The first type of agent (left group) is initialised from the notice.log file created by Bro IDS and represents the capabilities that it can perform. The second type of agent (bottom right group) show agents manually created using the agent-builder feature within the simulator allowing the creation of new agents for specific jobs. Finally, the third group (top right group) area created to represent sources of threat intelligence [15] found online. Agent nodes with a colour represent a coalition [7] of similar agents that all share the same condition type and effect type (further details in Section 3.4). An example of this is the intel-addr-6 agent which is capable of performing a reputation lookup against IP addresses for a particular source online. Within the simulator, there are many agents of this type which can perform a reputation lookup for an IP address and so are all grouped together under the intel-addr-6 agent (discussed further in Section 3.4). The white box at the bottom of Figure 4 is an interactive view activated by selecting agents to view information about them including their ID, a description of what actions they perform, the condition and effect information among other details.

4.1 Conditions & Effects Implementation

To increase compatibility across the different agent groups the conditions and effects were described in key:value pairs representing the condition or effect type and the condition or effect value. Wherever possible the Bro IDS syntax for describing types was used, for example, in Figure 4 agent-3 has been selected and its first condition can be seen as id.orig h representing the source IP address and the value is set to any meaning it requires any IP address to satisfy this condition. Agent 3 also has a second condition of ipLocality with a value of remote ensuring that it only performs its geographical lookup function on remote IP addresses, not internal IP addresses, which would be a waste of resources given the limited capacity of IoT devices. This agent in particular returns the an effect of ipGeoBlacklist with a value of either blacklisted or whitelisted. Once the agent has had both of its conditions satisfied and has participated in the extended data collection task it will broadcast its effect. If another agent has the condition of ipGeoBlacklist:blacklisted, it will then be able to perform its data collection task perpetuating the collection and analysis of relevant information.

4.2 Deployment Models

The architecture described so far has been framed as a distributed system that performs data collection and analysis over a network. The system can be deployed in two ways, the first being within the actual IoT devices if the capacity to support these agents is great enough. Having one or more agents operating on each device could be a viable distribution method if the devices have the capacity to support the processing required by each, which is far less taxing than having an entire IDS performing brute-force analysis from a single device. However, the agents may also be deployed in a supporting role by introducing more agents into the network. The agents described this far have all had a function to perform, e.g., an agent which can perform a geographical lookup against an IP address. However, this is a taxing function which may not be able to be supported on IoT devices. By including a number of lightweight agents that perform no real processing, but instead, just aggregate data for use within the extended data collection task can be used as a way to retrieve the information from IoT devices and send it to remote agents located on more suitable hardware for analysis. An example of this can be seen in Figure 4, the manual agent named "net-locality-2" performs no heavy processing such as performing a lookup but instead determines if a given IP address is locally or remotely located. This is a much less computationally taxing function to retrieve data from the IoT devices.

5. RELATED RESEARCH & DISCUSSION

Shakarian et al. [18] described a cyber attribution system [19, 20] that takes into consideration different data sources and uses MAS to reason about the origin of an attack through the use of agent reasoning. The system uses information gathered about the attack as well as information gathered from a wide range of military sources to reason in-depth about the attribution of an attack. A highlighted danger of relying on external sources of information is the trustworthiness of the source, which must be taken into consideration. Agents were used that could both reason about facts and make presumptions by factoring in trustworthiness for each individual source. Facts would, by default, be trusted more, while presumptions would be relied on less based on the trustworthiness of the source of information that the presumption was gathered from. This use of external information provided an effective way to gain extra contextual information for detected attacks but was heavily reliant on previously collected and catalogued information from military sources. When adhering to the design architecture of a CDMAS, agents must not rely on centrally collected and formatted data but instead be more adaptable to whatever information is available during time at which the agent searches for it. Furthermore, within the security environment where events can begin and finish on a very small time-scale, in order to enable useful real-time detection, the agents must be able to interact with the sources of information when needed rather than waiting on a, typically slower, human operator to provide the information. The importance of externally collected information, typically used during network layer detection is recognised and was included within the proposed architecture as one three types of agents (See Figure 4), however the system is extended beyond the use of previously collected and catalogued information through the implementation of agents that process information gathered during the live event. This approach is beneficial to security as the process of cataloguing data is typically performed sometime after events have occurred and so is not available in real time.

Jahanbin et al. [5] proposed a MAS framework for forensic information gathering which uses three types of agents for data collection, data analysis and alert generation. The authors note how the MAS paradigm is well suited to the task of forensic data collection as agents can be dispatched to areas of the network to perform collection and analysis of evidence such as log files. This system is structurally similar to Haack et al. [3] with layered agents passing information up the agent pipeline to a central agent for decision making. This central agent structure is similar to an IDS as it collects information and then makes a judgement based on that information, however, if some information is missing, the system would continue processing new information rather than actively searching missing data. It is desirable for agents to be able to actively seek out missing sources of information where some is expected in the security environment where attackers are prone to covering their tracks by removing information. Furthermore, in keeping with the principles of a MAS, agents should be adaptable when the environment changes. To improve upon this architecture and to allow agents to adapt to missing information the agents in our proposed model will search for information based upon what is already known (i.e., collected previously by other agents) and will continue to search for information in locations where previous evidence suggests more will be found. In the case of explicitly missing information, resulting in conditions that cannot be fulfilled, the data collection process is not halted but instead will continue to search for information where it can be found. Given the example shown in Figure 2 where a the extended data collection task begins with Agent-9 and includes several agents until concluding the search for information with Agent-8, if an intermediate agent could not collect information, an alternate series of agents could have been used to gather more information to compensate for the information gap.

Shanmugasundaram et al. [21] developed a distributed forensics system using a hierarchical approach with multiple configurable sensors that were distributed on a network. The system uses a variety of sensors and servers to collect and aggregate the information and attempt to derive the nature of the security event from the collected data. The system identifies the attack type based on which pieces of evidence are and are not found during the search. In the complex and changing environment of cyber security, this approach is desirable as the lack of information does not necessarily conclude no attack is taking place. The underlying system of conditions and effects is used to allow domain experts to independently create agents for specific applications and technologies without having to consult with each other. If information is required for the analysis of some application specific data, it can be abstractly defined within the agents condition, for example, a condition that the SSL certificate must be invalid, if this is then detected on the network the application specific agent may then participate in the extended data collection task without knowledge of how the SSL-monitoring agent works or which domain expert created it. Through the effects agents share conclusions about data they have collected and processed without exposing the

internal mechanism or particular data, this makes the system more scalable as domain experts can define agents using these abstract conclusions and have their agents fit into the existing agent ecosystem without the need to reconfigure existing agents.

Babar et al. [1] described eight concerns in regards to the security of IoT devices: user identification, tamper resistance, secure software communication, secure digital content, secure network access, availability, identification management and secure storage. While each of these concepts are particularly important to the security of IoT devices they lack a scalable and automatic mechanism for understanding attacks at a more abstract level. Currently, the focus of IoT security centres on the application of security technologies to the newly formed IoT domain. There is little focus on considering responsive architectures to detect and understand attacks against the wider network, for example, to attribute a number of smaller incidents across the network as part of a single targeted attack. The proposed agent architecture aims to dress the concerns of information leakage highlighted by Babar et al. by abstracting the low level data into abstract facts about the, for example, an agent that processes SSL certificates seen on the network will necessarily process sensitive information about the user but will only produce an abstract effect of either valid or invalid certificate. In an IoT environment which may contain multiple technologies by multiple vendors, implementing a security system that only shares the abstract conclusions rather than the personal data will reduce the risk of information being leaked from a compromised agent. Since only the conclusions are included in the global report transmitted between agents, a compromised agent will not have access to the personal information used to come to each of the individual decisions contained within it.

Oriwoh et al. [12] describes a forensics response model tailored for IoT environments taking into consideration the increased scope and complexity faced by forensic analysts. Performing manual forensics by trained practitioners in an IoT environment where the devices can be physically separated by great distances and the number of devices can be many can be a time consuming task. During the manual forensic process evidence is first gathered, then analysed for possible implication and if found to be suspicious is used as a starting point to find more related information. Through the use of conditions and effects the process of gathering information and using it as evidence to inform future searches has been modelled using a MAS to automate the forensic process in IoT environments. This model is beneficial over the traditional IDS approach of analysing security events where data is analysed according to known signatures or anomaly analysis without the forensic feedback loop to help inform future analysis of relevant data.

Suo et al. [24] discusses the security concerns at each of the four conceptual lays for IoT devices (perceptual, network, support and application). At the perceptual or hardware layer the integrity of the data stored is a concern and should be encrypted. At the network layer attacks on the systems availability are a concern (e.g., Denial of Service attacks) or the manipulation of protocols (e.g., Man-in-the-Middle attacks). At the support layer encryption and the use of anti-virus was recommended. Finally at the application layer data privacy and concerns about disclosing information were considered. Each of the described countermeasures at the specific layers are preventative and do not attempt to understand the attackers or consider the system as a whole. A more completely approach to security is needed wherein the behaviour of the attacker is studied to understand how they are attempting to penetrate the network and to collect relevant data from any affected nodes. The proposed agent architecture has attempted to fulfil this requirement by facilitating data collection across multiple network layers. Figure 4 shows the simulator with three classes of agents operating on different layers, for example, the Conn:Content Gap agent operates on the network level considering missing packets during a communication and the geoip-3 agent works at the application layer by matching IP addresses to physical locations. The benefit of this implementation is that an intrusion can be detected at any layer of the network stack and the resulting extended data collection task can cross the layer boundary to collect information, for example, an extended data collection task beginning with the detection of a port scan may progress up the network stack to collect information about the possible location of the attacker.

6. CONCLUSION & FUTURE WORK

In this paper we have discussed the use of a multi-agent security architecture for use within IoT networks. The distributed nature of IoT makes performing security more challenging as the devices do not have the benefit of a centralised IDS to monitor all connections. The distributed multi-agent architecture that aims to provide security by using the forensic process of gathering and analysing information to inform future actions will reduce the overall amount of communication and processing that must be done during the course of a security event by only considering the most relevant sources of data. The architecture is more scalable than traditional approaches to security as domain experts can create agents for a specific function and have them fit into the current agent ecosystem through the use of the conditions and effects mechanism.

We will continue to develop the architecture with a focus on developing distributed algorithms for use within it. Particular focus will be placed upon the scalability concerns of having domain experts create the knowledge of what is and what is not malicious. We believe that use of anomaly and behaviour analysis in place of manually created rules will be a better alternative for the IoT environment due to the increased chance of working with proprietary protocols which will often result in a knowledge gap and lack of support from traditional systems. The development of algorithms to work in environments such as this and the expansion of the simulator will be the primary focus of future work.

7. REFERENCES

[1] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad. Proposed security model and threat taxonomy for the Internet of Things (IoT). Communications in Computer and Information Science, 89 CCIS:420–429, 2010.

[2] S. L. Garfinkel. Digital forensics research: The next 10 years. Digital Investigation, 7:S64–S73, 2010.

[3] J. N. Haack, G. a. Fink, W. M. Maiden, a. D. McKinnon, S. J. Templeton, and E. W. Fulp. Ant-based cyber security. Proceedings - 2011 8th

International Conference on Information Technology: New Generations, ITNG 2011, pages 918–926, 2010.

[4] B. W. P. Hoelz, C. G. Ralha, and R. Geeverghese. Artificial intelligence applied to computer forensics. Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09, page 883, 2009.

[5] A. Jahanbin, A. Ghafarian, S. A. H. Seno, and S. Nikookar. A Computer Forensics Approach Based on Autonomous Intelligent Multi-Agent System. International Journal of Database Theory and Application, 6(5):1–12, oct 2013.

[6] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu. Security of the Internet of Things: perspectives and challenges. Wireless Networks, 20(8):2481–2501, 2014.

[7] S. Kraus. Negotiation and cooperation in multi-agent environments. Artificial Intelligence, 94(1-2):79–97, 1997.

[8] G. Lodi, L. Aniello, G. D. Luna, and R. Baldoni. An event-based platform for collaborative threats detection and monitoring. Information Systems, 39(1):175–195, jan 2014.

[9] P. Mockapetris. Domain Names - Concepts and Facilities. 1987.

[10] B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. IEEE Network, 8(3):26–41, 1994.

[11] A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke. SCADA security in the light of cyber-warfare. Computers and Security, 31(4):418–436, 2012.

[12] E. Oriwoh, D. Jazani, G. Epiphaniou, and P. Sant. Internet of Things Forensics : Challenges and Approaches. International Conference on Collaborative Computing: Networking, Applications and Worksharing, (9th), 2013.

[13] V. Paxson. Bro: a system for detecting network intruders in real-time. Computer Networks, 31(23-24):2435–2463, 1999.

[14] V. Paxson. Bro: a system for detecting network intruders in real-time. Computer Networks, 31(23-24):2435–2463, 1999.

[15] L. Randall. Critical Stack Intel.

[16] R. Roman, J. Zhou, and J. Lopez. On the features and challenges of security and privacy in distributed internet of things. Computer Networks, 57(10):2266–2279, 2013.

[17] K. Scarfone and P. Mell. Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology. Nist Special Publication, 800-94:127, 2007.

[18] P. Shakarian, G. I. Simari, G. Moores, and S. Parsons. Cyber Attribution : An Argumentation-Based Approach. Cyber Warefare, Springer International Publishing, pages 151–171, 2015.

[19] P. Shakarian, G. I. Simari, G. Moores, S. Parsons, M. A. Falappa, E. Engineering, C. Science, U. S. M. Academy, and W. Point. An Argumentation-Based Framework to Address the Attribution Problem in Cyber-Warfare. CoRR, abs/1404.6, 2014.

[20] M. A. F. Shakarian, Paulo, Gerardo I. Simari. Belief revision in structured argumentation. Foundations of Information and Knowledge Systems, pages 324–343, 2014.

[21] K. Shanmugasundaram, N. Memon, A. Savant, and H. Bronnimann. ForNet : A Distributed Forensics Network. Computer Network Security. Springer, pages 1–16, 2003.

[22] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers and Security, 31(3):357–374, 2012.

[23] Snort. Snort - Network Intrusion Detection & Prevention System, 2016.

[24] H. Suo, J. Wan, C. Zou, and J. Liu. Security in the internet of things: A review. Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, ICCSEE 2012, 3:648–651, 2012.

[25] T. Verwoerd and R. Hunt. Intrusion detection techniques and approaches. Computer Communications, 25(15):1356–1365, 2002.

[26] R. H. Weber. Internet of things aĂ¸Š Need for a new legal environment? Computer Law & Security Review, 25(6):522–527, 2009.

[27] M. Woolridge. An Introduction To MultiAgent Systems. Wiley, 2 edition, 2011.

[28] R. Zuech, T. M. Khoshgoftaar, and R. Wald. Intrusion detection and Big Heterogeneous Data: a Survey. Journal of Big Data, 2(1):1–41, 2015.