Tracking Advanced Persistent Threats in Critical Infrastructures through Opinion Dynamics

Juan E. Rubio[1], Rodrigo Roman[1], Cristina Alcaraz[1], and Yan Zhang[2]

[1]Department of Computer Science, University of Malaga,
Campus de Teatinos s/n, 29071, Malaga, Spain
{rubio,alcaraz,jlm}@lcc.uma.es
[2]Department of Informatics, University of Oslo, Oslo, Norway
yanzhang@ieee.org

Abstract

Advanced persistent threats pose a serious issue for modern industrial environments, due to their targeted and complex attack vectors that are difficult to detect. This is especially severe in critical infrastructures that are accelerating the integration of IT technologies. It is then essential to further develop effective monitoring and response systems that ensure the continuity of business to face the arising set of cyber-security threats. In this paper, we study the practical applicability of a novel technique based on opinion dynamics, that permits to trace the attack throughout all its stages along the network by correlating different anomalies measured over time, thereby taking the persistence of threats and the criticality of resources into consideration. The resulting information is of essential importance to monitor the overall health of the control system and correspondingly deploy accurate response procedures. Advanced Persistent Threat Detection Traceability Opinion Dynamics.

1 Introduction

Traditional SCADA (Supervisory Control and Data Acquisition) systems that manage the main production cycle of most of the industries have been working in an isolated fashion during years. In turn, the current scenario shows an evolution towards a model in which the organization externalizes some services by interconnecting their resources to Internet networks. The counterpart of this modernization is the appearance of new cyber-security threats and an increase of vulnerabilities in the industrial sector, as some reports show [1].

Many of these attack vectors are leveraged in APTs (Advanced Persistent threats). This is a type of sophisticated attack perpetrated against a particular

organization, where the perpetrator has significant experience and resources to penetrate the victim network without being noticed for a prolonged period of time [2]. Mechanisms such as firewalls, Intrusion Detection Systems (IDS), antivirus, etc. represent a first solution to the wide range of cyber-security threats faced by an industrial control system in presence of an APT. However, there is still a latent need to find advanced mechanisms that are capable of firstly detecting and then tracing one of this threats from a holistic perspective, during its entire life-cycle.

In this context, graph theory can be leveraged to apply distributed algorithms. Such algorithms can correlate various anomalies measured over the network that are potentially consequence of these attacks, while being able to locate the most affected areas within the topology. More specifically, we take the proposed scheme in [3] as a basis for our extended solution. This previous work proposed the use of opinion dynamics as a multi-agent collaborative algorithm, focusing only on the detection of topological changes over a graph-defined network. In this article, we show the feasibility of using the core of this approach to actually include realistic sources of anomaly and successfully trace the movement of an APT within a defined network architecture, which helps to deploy tailored response techniques. In order to achieve this, we review the literature of the most reported cases of APTs with the aim to realistically represent their stages and the sort of anomalies detected in each step of their kill chain. Finally, the effectiveness of the solution is theoretically demonstrated and shown in a test-case. We can summarize our contributions as:

- Modeling of an APT and its attack actions considering the persistence and criticality of resources.
- Adaptation and implementation of a distributed algorithm to detect realistic anomalies affecting the network nodes.
- Creation of indicators to inform about the threat evolution and the network health status.

The remainder of this paper is organized as follows: Section 2 outlines the proposed architecture and introduces the concept of opinion dynamics. In Section 3 the literature is reviewed to extract information about the APT modus operandi. Based on this extracted model, an algorithm that can detect and trace the presence of APT is simulated in Section 4. Then, the approach is experimentally analyzed using Matlab in Section 5. Finally, the conclusions drawn are presented in Section 6.

2 Preliminaries

In this section, we lay the theoretical base that permits, on the one hand, the formal representation of actual APT attacks over a defined network, and the execution of the detection technique, on the other.

2.1 Proposed network architecture

As discussed in the Introduction, most industrial ecosystems are nowadays adopting cutting-edge technologies onto their production chain and monitoring systems. The counterpart of the modernization of industrial technologies (which we will refer to as 'operational technologies' or OT) and its integration of IT ('information technology') in this context comes with the appearance of new cyber-security threats. Some of them are inherited from the IT paradigm and some other arise from the growing integration between IT and OT. We are talking about attack vectors such as denial of service, presence of malware in the control teams, exploitation of vulnerabilities in communication protocols, phishing and social engineering, etc. that will be further described in Section 3.1. For this reason, since there are several reported APTs that attempt to compromise resources belonging to both the IT and OT parts of the industrial network, it makes sense that the whole industrial topology can be split into these different sections: IT and OT, which will be interconnected by firewalls.

The formalization of the proposed network architecture is explained in the following. Let $G(V, E)$ be a graph that represents the entire network topology, that contains devices and communication links that transmit information and control commands between them. This network is composed by the IT and OT sections, which are respectively represented with subgraphs $G(V_{IT}, E_{IT})$ and $G(V_{OT}, E_{OT})$. These sections are joined by a set of firewalls placed in between (V_{FW} henceforth), so that $V = V_{IT} \cup V_{OT} \cup V_{FW}$. In order to understand how these network sections are merged, we firstly must introduce a graph theory concept related structural controllability [4] and power dominance [5]. The aim is to select the set of those nodes within the network that have the maximum dominance, which are called the *driver nodes* (denoted by N_D). As introduced in [5] and extended in [6], let us assume the following two observation rules over a given network $G(V, E)$:

OR1 *A driver node n_d in $\mathbf{D_N}$ observes itself and all its neighbors*: this is, the rest of nodes that share a communication link with n_d. This conforms the Dominating Set (DS) of G, and implies that every node not in $\mathbf{D_N}$ is adjacent to at least one member of $\mathbf{D_N}$.

OR2 *If a driver node n_d in $\mathbf{D_N}$ of degree $d \geq 2$ is adjacent to $d-1$ observed driver nodes, then the remaining un-observed vertex becomes observed as well.* This also implies that **OR1** $\subseteq$ **OR2** given that the subset of nodes that comply with **OR1** becomes part of the set of nodes that complies with **OR2**, conforming the Power Dominating Set (PDS). It means that every edge in E is adjacent to at least one node of $\mathbf{D_N}$.

An example of the election of these driver nodes is depicted in Figure 1. More specifically, the PDS will be used in the OT section of the industrial topology to represent the set of devices that are connected to the firewalls that also connect to the IT nodes, thereby merging both sections. The reason for such election is that in an operational environment multiple kinds of devices coexist.

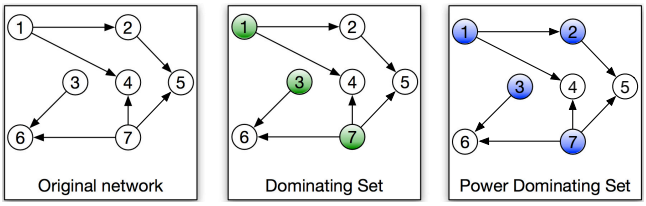


Figure 1: Observation rules for the election of the most dominating nodes

However, apart from sensors and actuators, PLCs and HMIs, only SCADA systems and high-level servers are actually connected to external networks (i.e., the IT section or Internet). Therefore, these are the nodes that hierarchically have more connectivity (so they will be linked to the firewall nodes), which is equivalent to the controllability concept introduced before. As for the IT section, since most of the devices range from ERP to customer-end systems (and whose computational capabilities are not as restricted as OT devices), we assume all nodes are connected to the firewalls and thereby can access the operational area.

However, concerning the network topology of the IT and OT section, we must note that each of these subnetworks is built with a different network distribution. On the one hand, $G(V_{OT}, E_{OT})$ follows a specific network construction centered on power-law distributions of type $y \propto x^{-\alpha}$, which is extensively used to model the topological hierarchy of a electric power grid and their monitoring systems [7]. These networks commonly contain substations, which are nodes with high degree (i.e., the number of edges incident on the node) connected to nodes with lower degree, like sensors and actuators. In turn, the IT section (given by $G(V_{IT}, E_{IT})$) is modeled according to a small-world network distribution, that represents the conventional topology of TCP/IP networks [8].

Once we have established the architecture for the network, we are in position to not only simulate attacks over the topology, but also deploying the detection system based on opinion dynamics, which is the main contribution of our work.

2.2 Opinion dynamics

In this section, we present the fundamentals behind the distributed detection technique from a theoretical point of view. In order to better understand what this solution measures and how it provides a valuable insight for further monitoring and response procedures, we must attend to how an APT behaves. As introduced in the first section, one of these threats comprises several stages over which the attacker manages to compromise certain devices over the victim network until he/she reaches an interest point. It is then when the intruder usually chooses to either disrupt the productive process or exfiltrate information to the attacker headquarters, as described further in Section 3.1.

This chain of individual attack actions commonly takes quite a long time to perpetrate the network resources; over this evolution, it would be of paramount interest to extract two main pieces of information:

1. The portion of the network that is subject of attack at any time, being possible to distinguish what set of devices are experiencing the same degree of anomaly, which can be produced by an attack. This is essential for applying effective response techniques and potentially isolate the attack, while the rest of the areas can keep functioning as in normal conditions, hence ensuring the continuity of the production by this way.

2. The traceability of events occurred to the network, with respect to the evolution of the intrusion throughout the network since the very first moment it broke into it. In this sense, when it comes to APTs, we must also take the persistence of attacks into special consideration at all times, since an advanced threat can go unnoticed during months and suddenly perform a new attack. In terms of the detection technique, this implies that it is also necessary to keep track of old subtle anomalies noticed in the network, to serve as feedback to the technique and correlate their relevance with current detected anomalies, that altogether may be part of a more ambitious threat. As it is technically described in Section 4, this weight given to anomalies experienced on the network in the past devalues over time depending on the criticality of the victim devices and the type of anomaly detected.

These objectives are accomplished by the means of a distributed cooperative algorithm called based on Opinion Dynamics [9], since it models the actual opinion formation among the individuals of a society: each of these individuals (denoted as agents in the following) does not simply share or disregard the opinion of the rest of agents, but he/she takes them into account to a certain extent in order to form his own opinion. From this moment on, what the opinion dynamics process does is to take an average over the opinions that can be repeated over and over again. This eventually leads to formed consensus of opinions belonging to different agents closer to each other. Correspondingly, it is equivalent to obtaining a fragmentation of the different opinions within the society, which can be applied to intrusion detection by representing the opinion according to the level of anomaly that each agent (representing a device of the network) experiences.

In the following, we formalize the intrinsics of this multi-agent algorithm, which constitutes a light modification of the approach proposed in [9] and an extension of the work presented in [3]. Let A be the set of agents of the system such that $A = \{a_1, a_2, ..., a_n\}$. Here, $x_i(t)$ represents the individual opinion of each a_i at time t (ranging from zero to one), where t refers to the iteration of the algorithm. On the other hand, the weight given to the opinion of any other agent j is denoted by w_{ij}, where $\sum_{k=1}^{n} w_{ik} = 1$ (therefore, agent i also takes its own opinion into account). Finally, the formation of the opinion for agent i in the next iteration $t + 1$ is described as follows:

$$x_i(t+1) = w_{i1}x_1(t) + w_{i2}x_2(t) + ... + w_{in}x_n(t)$$

Consequently, every agent adjusts its opinion in period $t + 1$ by taking a weighted average of the opinions of the rest of agents. When t tends to infinity, consensus of opinions are formed (so finally there are just a few opinions shared by clusters of agents), which can also be represented visually. Conversely, what we want to accomplish in our particular scenario is to use these opinions as a way to represent a detected anomaly by a given agent that is installed within the network, so that similar values (provoked by the same threats) converge the most critically affected areas from a high-level perspective (and the severity of such attacks) can be ultimately located.

One aspect that needs to be clarified is the assignment of weight among agents: for simplicity, for a given agent, we assume that the weight value assign to its neighbors is uniformly divided into those agents whose opinion is very close to its one (we establish a epsilon value of 0.2 of deviation between both values). This models the fact that agents close to each other with the same degree of anomaly are likely to be detecting the same threat in their surroundings.

In order to successfully apply this concept of a multi-agent algorithm to the context of anomaly detection in an industrial setting, there are various questions that need to be further addressed: i) who can play the role of agents within the industrial network, considering that there should be as many logical agents as nodes within the network ($|V|$ in our case); ii) how each anomaly can be represented as an opinion held by an agent, and how to retrieve such anomaly values; and iii) how the attacks affect the persistence and the anomaly detection, depending on their severity and the criticality of the victim nodes, which influences the persistence and the application of the opinion dynamics. These questions will be reviewed and answered in Section 3 through the analysis of real-word APTs and existing defense mechanisms and architectures.

3 Attack and defense models

3.1 Review of existing APTs, APT stages, and defenses

For the specification of the opinion dynamics algorithm, we need to provide an accurate representation of APT attacks in the context of our network model. Therefore, here we will first review the most important APT threats and groups that have specifically targeted industrial control systems. For the interested reader, a more detailed review of these APTs – including exploited vulnerabilities, software modules, etc – is available at [10].

Stuxnet (2009). Stuxnet was one of the APTs that popularized this concept and brought it to the limelight. Developed by a state agent, the main goal of this worm was to hinder the enrichment of uranium in the Iranian nuclear facility of Natanz [11]. It is believed that its primary infection vector, which was used to infiltrate the facility, was USB flash drives. Once the malware was installed in the 'patient zero' computer, it also used other mechanisms (network shares, infected project files) to spread through the internal network, searching

for the computers that directly controlled the uranium enriching centrifuges. Finally, the malware modified the code that controlled the centrifuges in order to silently destroy them.

DragonFly group (2013-2014, 2015-). Active since 2010, this particular APT actor has always focused on cyberespionage. On 2013, it started several campaigns against energy suppliers [12]. In its first wave of attacks, the main goal was to discover and map the existence of OPC (Open Platform Communications) SCADA servers located in the attacked network. For this purpose, after the initial infection, the malware queried the network in search of OPC servers using specific OPC DCOM (Distributed Component Object Model) calls. On the other hand, its second wave of attacks followed a more conservative approach: it retrieved information mostly by extracting documents and screenshots from the infected computers.

BlackEnergy (2015-2016). The BlackEnergy malware, created by an APT actor known as Sandworm, was used to attack the energy infrastructure of Ukraine in December 2015 [13]. After the initial infection, the first goal of the malware was to replicate to as much computers as possible through Windows Admin Shares (e.g. through PsExec and remote file execution). The second goal of the malware was to set up various connections to external command&control networks. Using these networks, malicious operators were able to activate various components (KillDisk, circuit breaker manipulator) that caused havoc in electricity distribution companies.

ExPetr (2017). ExPetr was a wiper disguised as ransomware, which targeted local administrations and various industrial companies in Russia and Ukraine [14]. It used two primary infection vectors: a modified version of the EternalBlue exploit used by WannaCry, and an trojanized version of the MEDoc tax accounting software. Once 'patient zero' was infected, this malware used both the EternalBlue exploit and the BlackEnergy propagation mechanisms to propagate over the local network. Inmediately afterwards, the fake ransomware component of the malware would be activated.

Another element that is essential for the formalization of the behaviour of APTs in our network model is the definition of the different attack stages (i.e. intrusion kill chains) that are performed by APTs. These attack stages – whose order can be changed depending on the specific APT – have been extensively studied and described by various academic and industrial researchers [15, 16, 17], and can be summarized in the following steps:

- ***Reconnaissance***. Adversaries gather information about the targeted industrial network, and create an attacking plan.
- ***Delivery***. After choosing a set of vulnerable computers ('patient zero') at the targeted industrial network, adversaries deliver the malware to those

computers, either directly (e.g. through email or vulnerable services) or indirectly (e.g. contaminating websites with malware).

- ***Compromise***. At this stage, the malware is executed in the target machine, and takes control of it. This stage involves several steps, such as *privilege escalation*, maintaining *persistence*, and executing *defense evasion techniques*.

- ***Command and Control***. Once the malware controls 'patient zero', it opens a communication channel with the remote attacker, which will be used to send commands, extract information, etc.

- ***Lateral Movement***. The concept of lateral movement encompasses the different steps that the malware takes in order to control other computers located in the targeted network. Lateral movement includes *internal reconnaissance*, *compromise* of additional systems, and *collection of sensitive information*.

- ***Execution***. The malware finally performs the attack against the targeted industrial network. Attacks range from *exfiltration* (extraction of sensitive data) to *destruction* of resources.

Finally, in order to define our defense model, and to provide an answer to the questions raised in the previous section, it is necessary to provide a brief overview on the actual state of the art of the existing defense mechanisms against the attack stages defined above. This information is extracted from more detailed reviews that are already available in the literature, such as [18]. Here, we will only highlight the most important aspects that will influence over the defense model of our network and the different detection probabilities:

- *Detection coverage.* As of 2018, there are multiple intrusion detection and prevention mechanisms, both commercial and academic, that are able to analyze the state of all elements and communication systems in industrial networks, including the field devices.

- *Central correlator systems.* There are several commercial platforms, such as [19], whose goal is to provide support for event correlation. These platforms can retrieve events and alerts from various domains (e.g. IT, OT networks) and from various sources (e.g. SIEM systems, vulnerability scanners) in a distributed way.

- *Beyond attack signatures.* There exist several solutions that are able to indicate the potential existence of anomalous situations, even if the attack signatures are unknown. Examples include not only diverse statistics (e.g. traffic volume, network connections, protocols used), but also machine learning mechanisms, specification-based systems, and industrial honeypots.

- *Network features.* in comparison to the IT infrastructure, OT networks exhibit a more consistent behaviour. This feature is actually used by certain detection mechanisms to more accurately pinpoint the existence of anomalies.

3.2 Representation of APT attacks and detection probabilities

After reviewing the behaviour of industrial APTs and the state of the intrusion detection mechanisms, we can define a realistic attack and defense model for our network architecture, thereby addressing the questions raised in Section 2.2. Our *attack model* is simple: we assume that, given a certain goal (exfiltration and/or destruction), adversaries are able to successfully perform an APT attack against the network architecture defined in Section 2.1 using any set of the attack stages defined in Section 3.1. As for the *defense model*, and given the actual state of the art in the area, we assume that all the elements of the network are covered by anomaly detection mechanisms, whose outputs can be retrieved by correlation systems similar to the ones described in [18].

By assuming the existence of a correlation system, it is possible to centralize the computation of the opinion dynamics algorithm in a more computationally powerful node (that gathers all the opinions and perform the correlation). As a consequence, the agents described in Section 2.2 can now be instantiated as logical agents, whose inputs will be retrieved from the different outputs of the anomaly detection mechanisms. From those inputs, every agent can now derive a certain opinion $x_i(t)$, or detection probability (i.e. the probability that an attack is taking place) for a given interval of time. These opinions are in turn influenced by the amount of alerts and their criticality. For example, a combination of anomalous statistics will slightly raise the opinion of an agent, and the existence of a confirmed attack (e.g. through the detection of an attack signature) will maximize that opinion. Compared to traditional detection mechanisms, the effectiveness of this approach resides in the ability to correlate anomalies throughout the network and hence trace the location of attacks, also considering their severity and persistence.

Taking into account the attacker model, we can now provide a formal representation of the intrusion kill chain of APT attacks. Let $attackStages$ be a *set of potential attack stages* that an APT can perform against the industrial control network $G(V, E)$ as defined in Section 2.1, such that $attackStages = \{attack\ stage_1, attack\ stage_2, ..., attack\ stage_n\}$. This set comprises the following elements:

- $\boldsymbol{initialIntrusion}_{(IT,OT,FW)}$. The initial access that affects a node n_0 (known as 'patient zero') of the IT network, OT network, and firewall, respectively.

- $\boldsymbol{compromise}$. The adversary takes control of a certain node n_i, obtaining higher privileges, maintaining persistence, and executing defense evasion

techniques. Moreover, this stage also includes the internal reconnaissance of the direct neighbourhood of n_i, $neighbours(n_i)$.

- ***targetedLateralMovement***$_{(IT,OT,FW)}$. From a certain node n_i, the adversary chooses a FW, IT, or OT node n_j from the set $neighbours(n_i)$, and executes a lateral movement towards that node. Note that, in this model, the concept of lateral movement only encompasses the delivery of malware towards the target node.

- ***controlLateralMovement***. From a certain node n_i, the adversary chooses the node n_j from the set $neighbours(n_i)$ with the highest betweeness (i.e. the node with significant influence over the network), and executes a lateral movement towards that node.

- ***randomLateralMovement***. From a certain node n_i, the adversary chooses a random node n_j from the set $neighbours(n_i)$, and executes a lateral movement towards that node.

- ***spreadLateralMovement***. From a certain node n_i, the adversary executes a lateral movement towards all nodes from the set $neighbours(n_i)$.

- ***exfiltration***. From a certain node n_i, the adversary establishes a connection to an external command&control network, and extracts information using that connection.

- ***destruction***. The adversary either destroys the node n_i, or manipulates the physical equipment (e.g. uranium enriching centrifuges) controlled by node n_i.

- ***idle***. In this phase, no operation is performed.

Once the set *attackStages* is defined, it is possible to represent APT attacks that target our particular network model $G(V, E)$. In particular, for every APT APT, there can be an ordered set $attackSet_{APT}$, comprised by one or more elements of the *attackStages* set, that represent the APT chain of attack actions. As an example, the attack set of Stuxnet can be represented as follows:

$$\begin{aligned} attackSet_{Stuxnet} = \{&initialIntrusion_{IT}, compromise, exfiltration, \\ &targetedLatMove_{FW}, compromise, targetedLatMove_{OT}, \\ &..., targetedLatMove_{OT}, idle, ..., destruction\} \end{aligned}$$

These particular instances are defined taking into consideration the overall goal of every APT. For example, in the case of the Stuxnet malware, its goal is to find a particular node $n_{OT'} \in V_{OT}$ that manages an uranium enriching centrifuge. Therefore, after infecting patient zero $n_{IT^0} \in V_{IT}$, it seeks the location of a firewall node $n_{FW} \in V_{FW}$ that connects the $G(V_{IT}, E_{IT})$ and $G(V_{OT}, E_{OT})$ regions. Afterwards, it moves inside the $G(V_{OT}, E_{OT})$ region until

it finds node $n_{OT'}$. Finally, after waiting for some time, the malware executes its payload, manipulating the centrifuge.

Regarding how the different attack stages influence over the application of the opinion dynamics and the calculation of the detection probabilities, we need to consider that certain attack stages will generate more security alerts. This, in turn, will increase the probability of detecting that particular attack stage. Therefore, we need to consider the existence of different classes of detection probabilities. Here, we define Θ as an *ordered set of detection probabilities of size d*, where $\Theta = \{\theta_1, ..., \theta_d\}$ and $\theta_i = [0, 1]$, such that $\forall \theta_i, \theta_i > \theta_{i+1}$.

$initialIntrusion(n_0)$	θ_3
$compromise(n_i \rightarrow neighbours(n_i))$	$\theta_2 \rightarrow \theta_5$
$*LateralMovement_{IT,FW}(n_i \rightarrow n_j)$	$\theta_5 \rightarrow \theta_4$
$*LateralMovement_{OT}(n_i \rightarrow n_j)$	$\theta_5 \rightarrow \theta_3$
$spreadLateralMovement(n_i \rightarrow neighbours(n_i))$	$\theta_5 \rightarrow \theta_4$
$exfiltration(n_i)$	θ_4
$destruction(n_i)$	θ_1

Table 1: Map of *attackStages* to Θ

Once Θ is defined, we can create a model that maps every element of the set *attackStages* to the elements of Θ. Such model, where $d = 5$ and $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$, is described in Table 1. We explain the rationale behind this mapping in Appendix B.

4 Detection of APTs

After formally representing the attack stages, plus their relation to the detection probabilities, we can now use the proposed detection probabilities as inputs to the opinion dynamics algorithm, and hence simulate its response in an industrial architecture when it faces a particular instance of APT.

Algorithm 1 describes the life cycle of an APT composed by a set of attack actions against a given network. Each of these attacks generates an anomaly that is detected by the corresponding agents (and possibly by their neighbors), increasing their opinion in a value defined by the previously introduced Θ. After this, as commented in earlier sections, we also introduce a attenuation value on the opinion that represents the effect of old attacks in order to reduce their influence when computing the current opinion. This "decay" value, applied in the UpdateOpinionsWithDecay function, depends on the attack stages suffered in the past by the agent and the criticality of its monitored device: the more devastating the alert generated is (during the detection phase), the longer its effect will take to disappear. Consequently, we define Φ as an ordered set of decay values, where $\Phi = \{\phi_1, ..., \phi_d\}$ and $\phi_i = [0, 1]$, such that $\forall \phi_i, \phi_i < \phi_{i+1}$. Therefore, for all $i \in d$, ϕ_i is inversely proportional to the θ_i value, and both are applied to the detected anomaly value after each stage. This procedure, explained in Algorithm 2, is a way to account for the persistence when computing the opinion dynamics. It is important to note that both the respective anomaly

and decay addition or reduction implies a normalization of the opinion value, from 0 to 1.

Algorithm 1 APT life cycle - anomaly calculation

output: δ *representing the delta value*
local: *Graph* $G(V, E)$ *representing the network, where* $V = V_{IT} \cup V_{OT} \cup V_{FW}$
input: $attackSet \leftarrow attackStage_{APT_x}$, *representing the APT chain of attack actions*

$x \leftarrow zeros(|V|)$ (*initial opinion vector*)
$\{performedAttacks \leftarrow \oslash\}$
$\{attack \leftarrow firstattackfromattackSet\}$
while $attackSet \neq \oslash$ **do**
  if $attack == initialIntrusion_(IT, OT, FW)$ **then**
    $attackedNode \leftarrow random\ v \in V_{(IT,OT,FW)}$
    $x(attackedNode) \leftarrow x(attackedNode) + \theta_3$
  else if $attack == compromise$ **then**
    $x(attackedNode) \leftarrow x(attackedNode) + \theta_2$
    for neighbour **in** neighbours(attackedNode) **do**
      $x(attackedNode) \leftarrow x(attackedNode) + \theta_5$
    end for
  else if $type(attack) == LateralMovement$ **then**
    $previousAttackedNode \leftarrow attackedNode$
    $attackedNode \leftarrow$ SelectNextNode$(G, attackedNode)$
    $x(previousAttackedNode) \leftarrow x(previousAttackedNode) + \theta_5$
    $x(attackedNode) \leftarrow x(attackedNode) + \theta_{3,4}$
  else if $attack == exfiltration$ **then**
    $x(attackedNode) \leftarrow x(attackedNode) + \theta_4$
  else if $attack == destruction$ **then**
    $x(attackedNode) \leftarrow x(attackedNode) + \theta_1$
  else if $attack == idle$ **then**
    $No\ attack\ performed$
  end if

  $x \leftarrow$ UpdateOpinionsWithDecay$(x, performedAttacks)$
  $performedAttacks \leftarrow performedAttacks \cup attack$
  $mergedOpinions \leftarrow$ ComputeOpinionDynamics(x)
  $\delta \leftarrow$ ComputeDelta$(mergedOpinions)$
  $attackSet \leftarrow attackSet \setminus attack$
end while

Once the x vector of opinions is updated with the new attack action (with θ) and attenuated due to old stages (through Φ), the opinion dynamics algorithm is executed to identify the affected areas of nodes and the level of severity of these attacks. However, although this gives insight of the location of threats (as it is visualized in the experimentation section), it would be also necessary to obtain an overall value of the network health from the opinion dynamics processing. Therefore, we have created the so-called delta indicator, which represents a global anomaly value and is computed in the ComputeDelta function. This value is calculated with the weighted average of opinions by the amount of agents that hold the same detected abnormality, as described in Algorithm 3. However, since this aggregated value is dependent on the number of agents to calculate the average, in practice we can compute it over different sections of the network (i.e., IT or OT), thereby increasing its granularity. Using these values, we can quickly know the overall anomaly degree of every portion of the network.

Note that all these algorithms and the approach itself are validated from a theoretical point of view in Appendix A.

Algorithm 2 Decay of anomaly values over time depending on the attack action

```
function UpdateOpinionsWithDecay(x, performedAttacks)
    for attack in performedAttacks do
        affectedNode ← getAffectedNode(attack)
        if attack == initialIntrusion_{IT,OT,FW} then
            x(affectedNode) ← x(affectedNode) − ϕ3
        else if attack == compromise then
            x(affectedNode) ← x(affectedNode) − ϕ2
            for neighbour in neighbours(affectedNode) do
                x(affectedNode) ← x(affectedNode) − ϕ5
            end for
        else if type(attack) == LateralMovement then
            origin ← getOriginOfMovement(attack)
            x(origin) ← x(origin) − ϕ5
            x(affectedNode) ← x(affectedNode) − ϕ3,4
        else if attack == exfiltration then
            x(affectedNode) ← x(affectedNode) − ϕ4
        else if attack == destruction then
            x(affectedNode) ← x(affectedNode) − ϕ1
        end if
    end for
    return x
end function
```

Algorithm 3 Computation of delta value

```
function ComputeDelta(mergedOpinions)
    opinionClusters ← uniqueValues(mergedOpinions)
    frequencyVector ← zeros(|opinionClusters|)
    for i:=1 to size(opinionClusters) step 1 do
        frequencyVector(i) ← CountOccurrencesOfOpinion(opinionClusters(i), mergedOpinions)
    end for
    δ ← 0
    for j:=1 to size(opinionClusters) step 1 do
        δ ← δ + frequencyVector(j) ∗ uniqueValues(j)
    end for
    δ ← δ/size(mergedOpinions)
    return δ
end function
```

i	1	2	3	4	5
θ_i	0.9	0.7	0.5	0.3	0.1
ϕ_i	0.01	0.025	0.05	0.075	0.1

Table 2: Detection probability and decay values used in the Stuxnet test case

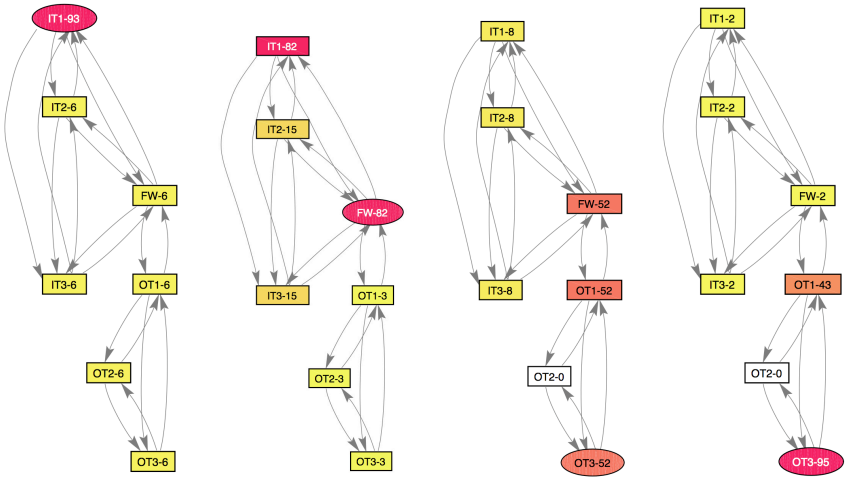
5 Experimental simulations and discussion

In the following, we present a test case for illustrating how we can apply the opinion dynamics-based technique while representing an APT against a given IT/OT industrial topology, as described in the paper. For this test case, we have implemented the network topology and algorithms 1, 2 and 3 in Matlab.

Let us assume that we have a topology composed by three OT nodes and three IT nodes connected by a firewall, as explained in Section 2.1. According to Section 3.2, Stuxnet comprises a set of nine different attack actions that will be perpetrated against the proposed network, where each node counts on an individual agent to monitor its anomalies. If we execute the opinion dynamics algorithm after each stage, we can analyze the different clusters of anomalies detected by sets of agents. Following the model presented in Section 3.2, we have assigned values for each θ and ϕ according to the ordered set of probabilities in Table 2, considering a realistic scenario. We have also introduced a deviation of 0.1 to values in θ to simulate a low level of noise or probability of detecting the corresponding anomaly after each attack stage. Figure 2 visually represents the resulting values in each agent after the four most representative stages, where (1) the attacker compromises the IT node and exfiltrates information, (2) compromises the firewall and then (3) moves to the last OT of the network and remains idle, right before the destruction of this node is performed (4). Four different idle operations are performed in this point, with a total of twelve attack actions. Numbers by the name of nodes represent the value of anomaly (opinions) that each agents holds.

As we can also see in Figure 2, the attacker traverses the whole network according to the Stuxnet behavior (where the current attacked node appears rounded), while the agents and its neighbors are able to detect the anomalies that consequently take place (the more red the node is, the greater the detected anomaly is). At the same time, we see how attenuation of anomalies also occurs, especially visible when the attacker leaves a node. In this example, the first IT node compromised is the number 1 while the final one is the OT number 3; the former is gradually attenuating its value as the attack evolves, according to the behavior explained in Section 4.

This ability to identify where the threat is active within the network is enabled by opinion dynamics. If we have a look at its value in form of a plot in some point, we obtain the graph in Figure 3. This corresponds to the execution of the algorithm (with 20 inner iterations) after the second stage depicted in Figure 2, where the FW is compromised after attacking the first IT nodes. As we can rapidly see in the resulting graph, there are two agents (the a_{FW} and



(a) Exfiltration (b) FW compromise (c) OT compromise (d) Destruction

Figure 2: Network topology used in the test case

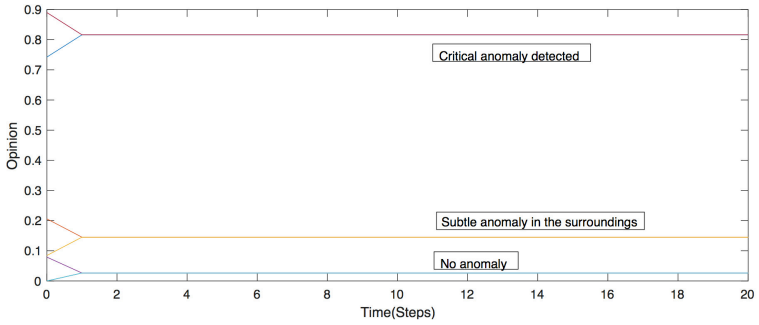


Figure 3: Opinion dynamics after the second stage

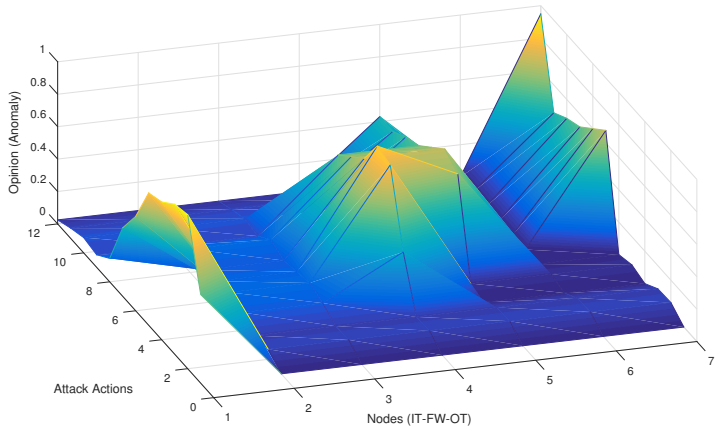


Figure 4: Evolution of the opinions over time to trace the APT stages

the a_{IT} node) that successfully detect the same level of critical abnormality in their area; this is also detected by some of their neighbors mildly, which is represented with the central consensus. Apart from these, the rest of nodes only detect a negligible value of anomaly.

By this means, we can statically identify where the threat is located and which severity it experiences. However, as commented in Section 2.2, it would be also necessary to trace all the events of the APT and highlight the most affected nodes it has traversed. In this sense, if we represent the succession of opinions agreed by agents over time for the Stuxnet attack described previously, we easily have such information, which is represented with Figure 4.

As we can see there, the opinion profile for all agents evolves over the set of APT attack actions, showing a more pronounced value in the IT section in earlier stages and the OT in latter phases of the Stuxnet APT, as the attack aims to ultimately compromise a PLC by firstly intruding the network through a IT node. A similar effect is seen when we study the change in the delta value, which can be calculated either in the whole network or on any of its subnetworks (i.e., IT or OT). Figure 5 shows the progression of this indicator in each case, which also shows us how IT delta decreases over time and its value in OT increases according to the chain of attacks. In general, the value acquires the highest value when the last OT node is compromised, since the network has suffered most of the attacks in the previous stages. Beyond that point, delta decreases (due to the idle operations) and then it finally increases with the destruction of the node.

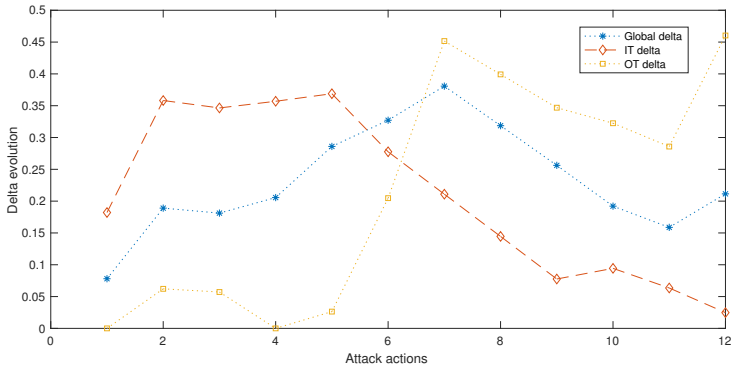


Figure 5: Evolution of delta opinions over the network for the Stuxnet attack

6 Conclusions

APTs nowadays represent a dramatic source of economic losses and reputation damage for the industry, which obligates researchers, managers and operators to make a great effort to analyze them to trace their behavior and anticipate their effect. It then becomes mandatory to explore new ways of detecting and tracing anomalies beyond traditional detection techniques. In this paper, we have described the feasible application of an already available theoretical approach based on a distributed collaborative algorithm (opinion dynamics). We review the literature to gather the set of attack vectors that these threats leverage with the aim of representing the anomalies and show the effectiveness of the algorithm in a realistic setting, which also considers the influence of persistence over time. As a result, we have valuable information about the status of the network at all times. This design constitutes the middle step towards a future implementation in a real testbed that is currently being under development, which also takes into account additional sources of detection and accurate indicators.

Acknowledgments

This work has been partially supported by the research project SADCIP (RTC-2016-4847-8), financed by the Ministerio de Economía y Competitividad, and DISS-IIoT, financed by the University of Malaga (UMA) trough the "I Plan Propio de Investigación y Transferencia" of UMA. Likewise, the work of the first author has been partially financed by the Spanish Ministry of Education under the FPU program (FPU15/03213). The authors also thank J. Rodriguez (NICS Lab.) for his valuable comments, support, ideas, and incredible help. You rock.

References

[1] L. Cazorla, C. Alcaraz, and J. Lopez. Cyber stealth attacks in critical information infrastructures. *IEEE Systems Journal*, 12(2):1778–1792, June 2018.

[2] Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon, Daesung Moon, and Jong Hyuk Park. A comprehensive study on apt attacks countermeasures for future networks communications: challenges solutions. *The Journal of Supercomputing*, pages 1–32, 2016.

[3] Juan E. Rubio, Cristina Alcaraz, and Javier Lopez. Preventing advanced persistent threats in complex control networks. In *European Symposium on Research in Computer Security*, volume 10493, pages 402–418, 2017.

[4] Ching-Tai Lin. Structural controllability. *IEEE Transactions on Automatic Control*, 19(3):201–208, 1974.

[5] Teresa W Haynes, Sandra M Hedetniemi, Stephen T Hedetniemi, and Michael A Henning. Domination in graphs applied to electric power networks. *SIAM Journal on Discrete Mathematics*, 15(4):519–529, 2002.

[6] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Parameterized power domination complexity. *Information Processing Letters*, 98(4):145–149, 2006.

[7] Giuliano Andrea Pagani and Marco Aiello. The power grid as a complex network: a survey. *Physica A: Statistical Mechanics and its Applications*, 392(11):2688–2700, 2013.

[8] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440, 1998.

[9] Rainer Hegselmann, Ulrich Krause, et al. Opinion dynamics and bounded confidence models, analysis, and simulation. *Journal of artificial societies and social simulation*, 5(3), 2002.

[10] Antoine Lemay, Joan Calvet, Franois Menet, and Jos M. Fernandez. Survey of publicly available reports on advanced persistent threat actors. *Computers & Security*, 72:26–59, 2018.

[11] Falliere, N., Murchu, L.O., Chien, E. W32.stuxnet dossier, version 1.4 (february 2011). https://www.symantec.com, last retrieved in April 2018, 2011.

[12] Symantec Security Response Attack Investigation Team. Dragonfly: Western energy sector targeted by sophisticated attack group. https://www.symantec.com, last retrieved in April 2018, 2017.

[13] SANS Industrial Control Systems. Analysis of the cyber attack on the ukrainian power grid. https://ics.sans.org, last retrieved in April 2018, 2016.

[14] Cherepanov, A. Telebots are back – supply-chain attacks against ukraine. https://www.welivesecurity.com, last retrieved in April 2018, 2017.

[15] MITRE Corporation. MITRE ATT&CK. https://attack.mitre.org, last retrieved in April 2018, 2018.

[16] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

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

[18] Juan Enrique Rubio, Cristina Alcaraz, Rodrigo Roman, and Javier Lopez. Analysis of Intrusion Detection Systems in Industrial Ecosystems. In *14th International Conference on Security and Cryptography*, pages 116–128, 2017.

[19] S2Grupo. Emas SOM – Monitoring System for Industrial Environments. https://s2grupo.es/es/emas-ics/, last retrieved in April 2018, 2018.

A Correctness proof: consensus-based detection and traceability

This section presents the correctness proof of the consensus-based detection and traceability problem for APTs. This problem is solved when the following conditions are met:

1. The attacker is able to find an IT/OT device in the system and attack it.
2. The detection system is able to trace the threat, thanks in part to the consensus (detection and traceability).
3. The system is able to properly finish in a finite time (termination).
4. The algorithm is capable of terminating and providing advanced detection at any moment (validity).

The first requirement is satisfied because we assume that the attacker is capable (i) declaring the chain of attacks in advance, such as scanning, lateral movement, exfiltration or destruction (see Section 3.2), and (ii) identifying kinds of devices (e.g. IT/OT nodes and firewalls) by their functionalities. The modus

operandi of the attacker is systematic except when the attacker needs to make a specific lateral movement, either through the selection of a new random neighbor node within the network or the selection of the neighbor with the highest betweeness. To comply with the predefined attack patterns, the attacker first needs to identify the first target node, which generally belongs to IT network − evidently, this characteristic depends on the type of attacker (insider or outsider) and their skills. If the attacker is an outsider, her goal is to find a $v_{IT_i} \in V_{IT}$ in order to penetrate by itself within the system, and to advance until reaching those nodes serving as firewalls such that $v_{FW_i} \in V_{FW}$. Once a v_{FW_i} is finally reached, the attacker tries to gain access in the operative network to compromise the most critical devices, i.e. $v_{OT_i} \in V_{OT}$. If the attacker is an outsider, the compromises relies, in this case, on the pre-established APT threat chain; i.e. on $attackSet$.

The second requirement is also found due to the software prevention agents, $a_i \in A$, integrated as part of v_{IT_i}, v_{FW_i} and v_{OT_i} of $G(V, E)$. These agents present capacities to detect anomalies and trace the intrusive presence by means of opinion dynamic parameters, the values of the which are attenuated according to time and aggressiveness of the threat (the decay factor). This attenuation, dependent on Φ_i, does not means to completely forget an incident in past. But rather, in remembering the most significant aftermaths of the previous attacks in order to show the advance of the threat in real time, and therefore its traceability.

Through induction we demonstrate the third requirement, corresponding to termination of the approach. To do this, we specify the initial and final conditions together with the base case. Namely:

Precondition: by assumptions, we assume that the attacker is an advanced expert with skills to reach the IT-OT communication channels belonging to $G(V, E)$. However, this capacity depends on the set $attackSet$ defined in Algorithm 1, which defines threat chain such that $attackSet \neq \oslash$.

Postcondition: (i) the attacker reaches the network $G(V, E)$ and compromises at least a node in V such that $attackSet = \oslash$ after the loop in Algorithm 1. And (ii) the system successful detects the threat such that $\delta > 0$ and marks the traceability according to the real consensus state of $G(V, E)$, registered in the array vector x.

Case 1: $attackSet \neq \oslash$, but $\mid attackSet \mid = 1$. In this case, the attacker needs to launch the unique attack defined in $attackSet$. As mentioned, if the attack does not imply a lateral movement, the success of the threat is concentrated on just one node in V, since the following iteration of the loop implies that $attackSet \leftarrow attackSet \backslash attack$, and therefore $attackSet = \oslash$. To the contrary, if the attack entails a lateral movement, then the attacker has to select a new neighbor node, either from a random or target point of view.

Any attack in V means an impact on the attacked node with a significant influence in its opinion dynamic (i.e. $x(attackednode)$. If, in addition, the

decay factor is activated, the system weakens, but does not delete, the aggressiveness of the threat to stress the current trace of threat over the time. This computation is possible through Φ_i in Algorithm 2. Once x is updated, the system computes the δ value taking into account the weighted average of the opinion dynamics of the entire system (see Algorithm 3).

Induction: if we assume that we are in step k ($k \geq 1$) of the loop where $attackSet \neq \oslash$, then **Case 1** is going to be considered each time. When $k = \mid attackSet \mid$, the system computes **Case 1** and ends the detection algorithm with $\delta > 0$ since $attackSet = \oslash$, showing the traceability of the threat through x and complying with the postcondition.

Finally, the latter requirement is also satisfied since the algorithm finalizes and detects the threat through opinion dynamic (either individual or collective) and shows the traceability of the threat over the time.

B The mapping of the $attackStages$ to Θ

We have presented in Section 3.2 a model that maps every element of the set $attackStages$ to the elements of $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$. For this mapping, we have taken into consideration the defense mechanisms analyzed in Section 3.1. In particular, the rationale behind this mapping is as follows:

- We assign θ_1 only to the *destruction* stage, because any major disruption in the functionality of a device (e.g. unavailable resources, device turned off) will trigger multiple high priority alerts. Note that, as explained in our defense model, we assume that all field devices are also covered by detection mechanisms, thus any attack (e.g. the Stuxnet final payload) against these sensitive devices can be easily detected.

- θ_2 is only assigned to the element at the left side of the *compromise* stage ($n_i \rightarrow neighbours(n_i)$). The reason is simple: the act of compromising and taking control of n_i will not only trigger various host alerts, but also multiple network alerts due to the various discovery queries targeting all $neighbours(n_i)$. The correlation of all these events will draw attention to the state of n_i.

- For θ_4, we consider the security alerts caused by combination of a single anomalous connection to a node plus the delivery of malware to that node. As such, this θ covers all the elements at the right side of the $lateralMovement$ stages. Note, however, that in some particular cases (like the $initialIntrusion$ stage and the $*LateralMovement_{OT}$ stages), additional anomalies will be detected: a potentially anomalous external connection, and a certain instability in the otherwise stable OT communication environment, respectively. Therefore, the θ assigned to the elements of those stages will be θ_3.

- Finally, θ_5 is assigned to those stages where the nodes produce or receive anomalous traffic (e.g. a connection that deviates from what is considered as normal traffic). Again, in situations where a connection with the outside world is made (e.g. $exfiltration$ stage), as the possibility of anomalous traffic will increase, the θ will be increase as well.