Attack Containment Framework for Large-Scale Critical Infrastructures *

Hoang Nguyen
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: hnguyen5@uiuc.edu

Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: klara@cs.uiuc.edu

***Abstract*— We present an attack containment framework against value-changing attacks in large-scale critical infrastructures, based on early warning and cooperative response approaches. We define an information structure, called *attack container*, which captures the trust behavior of a group of nodes and assists to contain the damage of the attack. The attack container is then used for distributed early warning and cooperative response in our framework. The simulation results show that our containment framework can detect, mitigate and contain large-scale attacks quickly.**

I. Introduction

Nowadays, critical infrastructure (CI) such as Power Grid is supported by large-scale computer information systems whose losses could lead to the reduction or even disruption of the critical infrastructure services. These computer information systems, however, are vulnerable to cyber-attacks as they move from isolated systems with propriety protocols to systems with commercial off-the-shelf components [1]. Therefore, protecting CIs against cyber-attacks becomes a very important problem [2].

In this paper, we consider *large-scale* critical infrastructures that are monitored and controlled by *multi-tier* and *hierarchical structure* computing networked control systems[1]. We are interested in this domain to investigate two issues: 1) the effect of large-scale value-changing worm-like attacks in the control systems for critical infrastructure and 2) fast containment of these attacks. We will investigate these attacks in CI sensing devices that sense the level of power distributed through power lines, or the level of gas flowing through the pipelines. The threat assumption will be that the sensing devices, leaves of the CI network hierarchy, will get "infected" due to erroneous vendor maintenance and upgrades of the devices [2].

The type of worm-like attacks we consider are slightly different from Internet worm attacks. We consider in a subtree group of sensors that a vendor erroneously upgrades one device, and then initiates an automated distribution firmware/software upgrade protocol, acting similar to a worm, to upgrade other devices in the subtree[3]. The vendor moves on to another subtree of devices to upgrade them in the same erroneous way[4]. We refer to this type of attacks as *value-changing attacks*. Furthermore, this attack could cause serious consequences. For example, in a station of the power grid, maliciously reported voltage values from a digital relay could cause a trip command to other power devices for protection, which may stop the operation of the this station. Furthermore, such local effect at one station could cause a cascading effect to other stations due to the voltage stability issue. Therefore, it is very important to detect spread of erroneous upgrades and contain this type of value-changing worm-like attacks.

Since we consider worm-like attacks, it is important to emphasize the work addressing Internet worms. The Code Red [8] and Slammer worm [9] outbreak have demanded a special attention from the research community on network worms. Moore et al. [10] and Staniford et al. [11] have shown that the worm containment must be *automatic* to have any chance of success because worms spread too fast for humans to respond. Much work on worm analysis and modeling [11][12][13][14][15] has shown that fast scanning worms have an exponential rate of infection after the slow-start phase. Therefore, it is very important to detect the worms at their *early stage* and to response *quickly*. Zou et al. proposed an algorithm for early warning of worms based on Kalman filter [16]. Moore et al. presented the concept of a centralized "network telescope", in analogy to light telescope, by using a small fraction of IP space to observe security incidents on the global Internet [17]. A distributed telescope was introduced in [18] [19] where smaller telescopes observing different regions of the network address space are combined into a single, large network telescope. Researchers have also suggested to use cooperative mechanisms where nodes exchange alerts among themselves [20] [21] [22] [23] [24] [25]. Nojiri et al. proposed the notion of "friends" where nodes could detect their infection and warn their friends [20]. Senthilkumar et al. further looked at the "friends" protocol in a hierarchical structure rather

*This material is based upon work supported by the National Science Foundation under Grant CNS-0524695. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[1]This topology can be found in many critical infrastructure such as SCADA networks and Power Grid[3][4][5].

[2]Intelligent Electronic Devices (i.e. sensing devices) in a power substation have firmware and other software-based services. A bug in updated firmware or an intended malicious bug in software services could cause the sensing devices behave maliciously

[3]EnerVista, a product of General Electric, allows to program and update the settings of all devices directly in a substation (i.e. subtree) via Ethernet [6].

[4]This threat of the moving vendor has been discussed in [7].

than just in a flat structure [21]. These results essentially showed that a cooperative response mechanism could give faster containment at the cost of false alarms.

Previous work on worm detection and containment has shown that distributed monitoring and detection is more accurate with lower false alarms while cooperative mechanisms could help in faster response time. None of the current solutions, however, achieve low false alarms and fast response to attacks. Hence, in this paper, we investigate the convergence of *distributed early warning* with *hierarchical cooperative response mechanisms* to achieve fast attack containment with low false alarms. Through distributed early warning, each node cooperatively exchanges their observations to carry out the detection for early warning. Through hierarchical cooperative response mechanisms, nodes organize themselves in a multi-layer hierarchical structure where both *vertical* and *horizontal* communication are exploited. The benefits of this multi-layer hierarchical response are in the "summarization" of the attack characteristics and suppression of false alarms while still achieving a fast alert propagation and containment throughout the system.

Our novel *Attack Containment framework* (ACF) is based on the novel "*attack container*" (AC) information structure. The AC structure is defined by a sensor group, and keeps track of the trust behavior of nodes in the group. The ACF issues distributed early warning and mitigates the attacks. The ACF also includes distributed monitoring and detection which allow the system to quickly detect *abnormal* and *critical* events. Each parent node in the network plays a monitoring role, uses a non-parametric Cumulative Sum (CUSUM) algorithm for quick detection of abrupt sensor measurement changes and classifies measurements according to two metrics: 1) *abnormality* to represent a fraction of sensors with abnormal behaviors and 2) *severity* to represent how severe the attack is in its sub-tree. We show that the metric pair (*abnormality, severity*), embedded inside the *attack container information structure* (ACIS), helps in characterizing attacks and defining two important regions: abnormal region where the system collects evidence about attacks and enters early warning, and critical region where the system is confident about the presence of attacks and reacts quickly to contain them.

Our contributions in this paper are three-fold: 1) concept of *attack container* that enables convergence of early warning and cooperative response algorithm via efficient data structures, and expressive metrics such as abnormality and severity, 2) attack containment framework including attack container and its important associated services and protocols which enable the fast containment of value-changing worm-like attacks, and 3) integration of early warning and hierarchical cooperative response mechanism in ACF and validation of the promising ability against other cooperative mechanisms.

The rest of the paper is organized as follows. First, we present our system models and assumptions in Section II. Then we present the concept of attack container and the attack containment framework in Section III. Section IV shows the architecture and the implementation of ACF services and protocols. In Section V, we show our simulation setup and results. Finally, we conclude the paper in Section VI.

II. System Models & Assumptions

A. Network Model

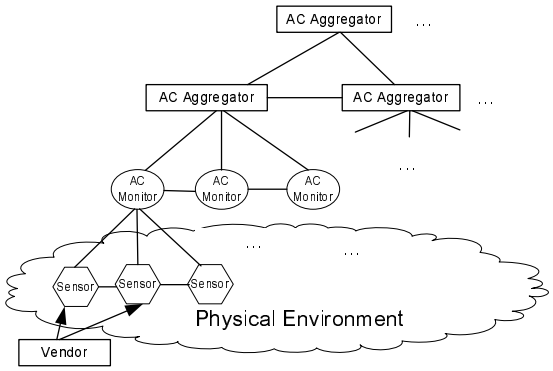


Fig. 1. Network Model

We model the underlying control system for the critical infrastructure as a *multi-tier and hierarchical* network with peers communicating at the same level, shown in Figure 1. Nodes are connected with their parents, siblings and children[5]. Leaf nodes in the tree are sensing nodes (called *sensors*). Sensors are digital devices attached to physical measurement devices that capture the measurement value in digital form and send this value to the higher level intermediate digital nodes for monitoring and control[6]. Nodes, that are monitoring multiple sensors, are called *AC monitors*. The function of these nodes is monitoring sensors, detecting anomalies of sensors and reporting their values to *AC aggregators*. We assume monitors have the ability to "immunize" the healthy sensors by issuing a command that cause sensors to deny any future upgrades. It implies that immunized sensors will be immune to erroneous upgrades. The last type of nodes are called *AC aggregators*. Aggregators receive reports from their children and neighbors, "summarize" and alert their neighbors and parents if necessary.

B. Trust Assumptions & Threats

Trust Assumptions: We assume *all AC nodes (monitors, aggregators) except sensors are trusted*. Specifically, the operating systems are trusted and the software running on these nodes is tampering-resistant. This can be achieved with the support of hardware such as eXcute Only Memory architecture (XOM) [26] or Trusted Platform Module (TPM) [27]. These techniques basically can prevent modifications on executed software. However, they cannot prevent Denial of Service attacks. We also assume that all nodes except sensors will trust each other. Since intermediate nodes are already trusted, this assumption can be achieved by using secure communication mechanism (authentication, encryption).

[5]In this model, when we talk about "neighbors" of a node A, we mean nodes that are in the same hierarchical level and connected with A.

[6]Here we consider sensors that are different from motes. Power-consumption is not an issue.

However, we assume *weak security assumptions on sensor nodes* (e.g. no trusted software). These nodes can be compromised and be used for some specific purposes. They can be infected via erroneous external updates and expose random behaviors in reporting values.

Threats: Although the critical infrastructure networks are usually isolated from open shared networks, they get connected to external machines. (e.g. vendors do software updates on sensing devices). These external updates introduce threats to inject erroneous upgrades causing undesirable value changes into the system.

Once a sensor is infected or compromised[7], it is fairly easy to infect neighbor nodes because they can directly talk to each other due to a given automated upgrade distribution protocol. This behavior can cause a dramatic spread of erroneous upgrades, exhibiting worm-like behaviors in the network.

Attack/Failure Model: Once a sensor is infected or compromised through an erroneous, it will engage in abnormal behaviors on readings. The first type of abnormal behavior will be the *value-changing attacks* where the sensors maliciously deliver modified readings. Specifically, these attacks will shift the mean of reading values arbitrarily. We call the attack the *mean-changing attack*. The consequence of reporting malicious values may possibly lead to false alarms of the critical infrastructure systems, inappropriate decisions of operators or other catastrophic failures of the critical infrastructure systems.

The second type of abnormal behavior will be changed reading patterns to cause DoS(Denial of Service) or WoS (Withdrawal of Service) attacks where the readings can be flooded or delayed arbitrarily. Although our framework enables the detection of both types of attacks, *we only consider the first type in this paper due to allowed paper length.*

Infection model:

Due to the automated upgrade distribution protocol in a subtree, we assume that the infection model in the subtree is a K-multicast model. In this model, we assume that once the vendor/attacker logins to a subtree, he then keeps choosing another K non-infected sensors to update until all sensors in the subtree are upgraded. If K is equal to the number of sensors in the subtree, it becomes the model being used in [6].

Besides the infection caused by the automated upgrade distribution happening within each subtree, we have another infection caused by the mobile vendor/attacker among subtrees. We assume the vendor logs in into each subtree, triggers the automated upgrade distribution protocol, logs out and moves to another subtree. We denote the time for the vendor/attacker to finish his job in a subtree before moving to another subtree as T_{vendor}.

C. Sensor Data Model

Data reported by sensors is the physical data such as temperature, voltage or water level. Since we are interested in both normal and abnormal behaviors of sensors, we model the data of a sensor i as the random process $\{X_t^i\}$

$$X_t^i = \mu_0^i + N^i I(t < k^i) + (h^i + M^i) I(t \geq k^i)$$

where $N^i = \{N_t^i\}_{t=0}^{\infty}, M^i = \{M_t^i\}_{t=0}^{\infty}$ are sensor measurement noise factors with $E[N^i] = E[M^i] = 0$, $I(\cdot)$ is the indication function, k^i is the time the sensor has abnormal behavior and h_i is the mean deviation of the sensor measurement. We further assume that $0 < h^i < h_{max}^i$, where h_{max}^i is the upper bound of the mean increased under abnormal condition.

Under this model, the sensor has normal behavior when $t < k^i$ and $E[X_i] = \mu_0^i$. It becomes abnormal when $t > k^i$ and $E[X_i] = \mu_0^i + h^i = \mu_1^i$ where μ_0^i, μ_1^i are usually referred to as the mean of X^i before and after the change happens.

III. Attack Containment framework (ACF)

The goals of our protection system are monitoring, detecting, isolating infected sensors and immunizing healthy sensors as soon as possible, under communication and false alarm rate constraints.

As mentioned in Section II-B, we only consider value-changing attacks. In the subsequent sections, first we show our approach to achieve the goals. Second, we give the concept of the attack container. Finally, we show the details of our attack containment framework (ACF) including protocols and algorithms for monitoring, detection and containment.

A. Framework Overview

To deal with the value-changing infection attack in a large-scale system, we use a *distributed monitoring and detection approach for early warning* and a *hierarchical cooperative response strategy for attack containment*.

Distributed Monitoring and Detection for Early Warning: Each AC monitor maintains an attack container for its sub-tree, which is updated and aggregated on receiving either sensor readings or attack containers from the children. It performs a non-parametric CUSUM to detect value-changing attacks. Each monitor further classifies, in its sub-tree, attack according to the abnormality metric representing the fraction of sensors having abnormal behaviors and the severity metric representing how severe the attack is. These two metrics, embedded inside the ACIS and exchanged among the nodes, present the metric plane (see Figure 2) and help in characterizing the attacks and in cooperative response for containment purposes.

Hierarchical Cooperative Response Strategy for containment: Once AC monitor nodes detect a potential attack in their subtree, characterized by the pair of *(abnormality, severity)* as shown in Figure 2, they will start to react. In the early phase (i.e. abnormal region of the metric plane), AC monitors and aggregators alert their peers and parent faster depending on the evidence of the attack. In the later phase (i.e. critical region of the metric plane), they react strongly and contain the attack by immunizing healthy sensors.

The hierarchical cooperative response strategy has two important advantages: 1) A high-level node (i.e. AC aggregators) will have a broader knowledge about its sub-tree since it has

[7]"Infected" sensors refer to those that were erroneously upgraded.

an aggregated view from its children and an aggregated view of its neighbors. False alarms due to the detection algorithm can be fused and suppressed to have only a small effect. 2) The attack can be contained faster due to the early alert propagation at each hierarchy level.

Before going into the detail of the strategy, we present the *attack container* that will be the core of our framework.

B. Attack Container

Attack Container is an information structure (ACIS), defined per sensor group, that keeps track of the trust behavior of sensor nodes in the group[8]. ACIS data includes meta-data that is built from readings of sensors or others' ACIS data. For example, AC will store abnormality and severity values for each sensor data reading as well as summary of (abnormality, severity) pairs over longer period of time. Furthermore, AC will store timestamps of vendor upgrades, names of vendors, and other attack-relevant data. Note that in this paper we concentrate on two information in the AC, the abnormality and severity metrics. AC monitor builds its ACIS from sensors' readings. The aggregators can build the ACIS by *summarizing* the ACIS data of its children and neighbors. The details of ACIS construction, ACF operations and the ACF framework will be discussed in subsequent sections.

C. Severity

The severity metric indicates how severe the attack is in terms of erroneous sensor readings. Severity metric must be aggregatable, i.e. it is possible to compute a severity of a node from other severities. To be precise, we define the severity of a sensor and intermediate nodes as follows.

1) Severity of a sensor: A severity $S(i)$ of a sensor i is measured by the ratio of the deviation of its mean values under attack from the mean values in normal condition over the upper bound of that deviation. Formally, $S(i) = h_i/h_i^{max}$, where h_i, h_i^{max} are the deviation and upper bound of the shifted amount of mean, respectively (see section II-C). $S(i)$ takes only values in the interval $[0..1]$.

2) Severity of an intermediate node: A severity of an intermediate node k is an average of severity of its children. Formally,

$$S(k) = \sum_{j=1}^{|children(k)|} S(j)/|children(k)| \qquad (1)$$

$S(k) \in [0..1]$. It is now becoming obvious that severity could be aggregatable by the above definition.

D. Abnormality

Abnormality $A(i)$ of a sensor i is 1 if its corresponding AC monitor declares the sensor is abnormal (see Section III-E and IV-A) and is 0 otherwise.

Abnormality $A(k)$ of an intermediate node k is the fraction of abnormal children in the subtree over the total number of its children. Formally,

$$A(k) = \sum_{j=1}^{|children(k)|} A(j)/|children(k)| \qquad (2)$$

It is fairly easy to see that the abnormality metric is in the range $[0..1]$ and is aggregatable.

E. Role of Severity and Abnormality

In ACF, the two metrics: severity and abnormality play a very important role in classifying the degree of an attack. While abnormality captures the *scale* of the attacks, severity captures the *impact* of the attacks. Putting them together will help in characterizing the degree of the attack and thus help the system to react accordingly.

Figure 2 illustrates the role of severity and abnormality in characterizing attacks in a case where we assume $h_i^{max} = 1$ for all sensors i[9]. The abnormal behavior detected within a small number of sensors with slight deviation of measured values is considered as "normal" since it just might be the result of noises, normal failures or false alarms. As the number is getting larger in terms of severity and abnormality, the behavior is considered as "abnormal". For example, the situation where a large number of nodes has behaviors with small severity is abnormal since it might be the beginning phase of a large-scale attacks. Similarly, a single but severe failure can be considered as abnormal because the consequence could be catastrophic to the critical infrastructure. As the two metrics exceed a certain threshold, the attacks are declared as critical. The system must respond quickly to mitigate and contain the attacks.

As one might notice, fully characterizing the exact curves of the "abnormal" and "critical" thresholds is very challenging since it depends on the semantics and nature of the system. Therefore, we present a simple approximation of these two thresholds by treating them separately. For each metric, we define two thresholds: "abnormal" $(\widetilde{A_1}, \widetilde{S_1})$ and "critical" $(\widetilde{A_2}, \widetilde{S_2})$. The regions defined by these points are the approximations of nodes behavior.

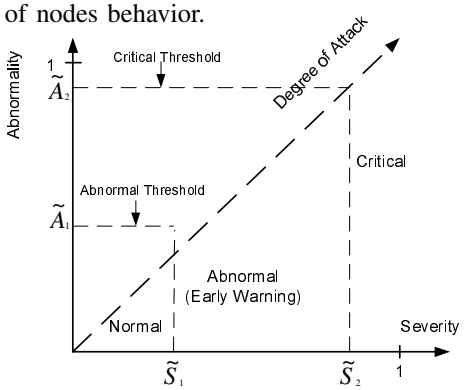


Fig. 2. Role of severity and abnormality in characterizing the degree of attacks

[8]The term AC and ACIS are used interchangeably since they have the same meaning

[9]This assumption is for the simplification of the illustration of the attack degree function. In general, the function can be much more complicated.

IV. ACF Architecture and Implementation

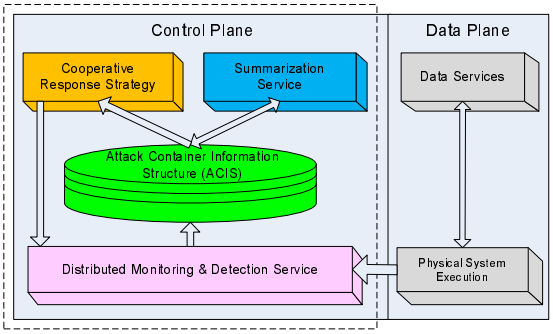


Fig. 3. Architecture

Figure 3 shows the architecture of our framework. Our framework resides within the control plane. At the bottom layer is the distributed monitoring and detection service providing monitoring and detection capabilities. It is also the first step to create ACIS for later use of other services.

Summarization Service is performed by monitors and aggregators where ACIS data from the children and neighbors is summarized to provide the summary of behaviors at each node.

Finally, ACIS are used by the Cooperative Response Strategy to provide early warnings and containment. Cooperative Response Service interacts with Distributed Monitoring and Detection Service in case of early warnings such as requesting for faster reports.

It is important to emphasize that attack container data is stored and maintained at each node, i.e. it is distributed.

A. Value-Changing Detection and Cumulative Sum Monitoring Box (CMB)

Monitoring and detection service in ACF creates the attack container. This operation, carried out at the AC monitors, will monitor the sensors and detect the anomalies. In this case, the input is the data stream from sensors and the output is abnormality and severity meta-data values, shared in AC. However, this operation could also be used at other higher level nodes such as aggregators to monitor their children or neighbors and detect abrupt changes on severity and abnormality data series.

The design of this service is base on the concept of *Cumulative Sum Monitoring Box* (CMB). A CMB is used to monitor the changes of a data stream. It processes the sensor data stream and generates alerts based on two thresholds: *abnormal threshold* and *critical threshold* (i.e. (A1, A2) for abnormality and (S1, S2) for severity as mentioned in Section III-E). The CMB could return "normal" if no change in the data stream is detected, "abnormal" if the change exceeds abnormal threshold and "critical" if the change exceeds critical threshold. The illustration of CMB is shown in Figure 4. For notation convenience, we denote $CMB(DS)$ as the current state of the data stream DS. CMB uses non-parametric Cumulative SUM change-detection algorithm as its value-changing detection algorithm. We now give a brief description of the non-parametric CUSUM. The details of the non-parametric algorithm can be found in [28] and [29].

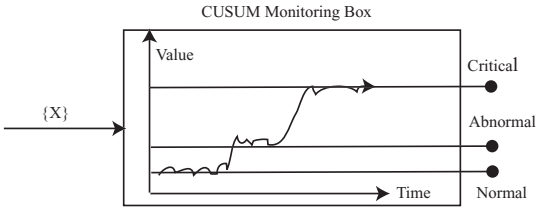


Fig. 4. CUSUM Monitoring Box

1) Non-parametric CUSUM: CUSUM algorithm was first introduced by Page [30] and has been known to be good at detecting abrupt changes [28]. Recently, it has been also applied to detect Denial-of-Service attacks [29][31] due to its simplicity and light-weight while still achieving a good performance in terms of false alarm rate and average detection delay.

Essentially, the Cumulative Sum Change-Point keeps the positive part of the log-likelihood ratio and triggers an alarm if the cumulation exceeds the threshold. The threshold is set according to the required false alarm rate. The non-parametric version extends the parametric method by estimating the changes based on historical observation.

Formally, consider a sensor i with the data model defined as in Section II-C. Let us define "accumulator" $Y_t^i = (Y_{t-1}^i + \xi_t^i)^+, Y_0 = 0$ where $X^+ = \max(0, X)$ and $\xi_t^i = X_t^i - \mu_0^i - \delta E[\mu_1^i | X_0, X_1 ... X_{t-1}]$ where δ is the sensitive factor.

The rule for CUSUM to declare the change that happens at time k^i is

$$k^i = \min\{t : Y_t^i \geq T\}, t = 0 \rightarrow \infty$$

where T is the threshold and is normally set to $\log \frac{1}{FAR}$ and FAR is the desirable false alarm rate.

The estimation of μ_1^i based on historical observation could be an average or a simple linear estimator. However, as shown in [29], the adaptive exponentially weighted estimator is preferred due to the ability to "forget" observations that are far in the past. This estimator predicts based on sequential inputs as follows.

Let $\hat{\theta}$ be the current prediction of the mean μ_1^i. If $\xi_t^i = 0$, we just set $\hat{\theta}_n = \mu_0^i$ because there is no change occurring. If $\xi_t^i > 0$,

$$\hat{\theta}_t = \frac{1}{\beta_{t-1} + 1}(\beta_{t-1}\hat{\theta}_{t-1} + X_t^i) \quad (3)$$

where the weight is $\beta_n = 1 + \beta_{n-1}$ and is reset to zero when $\xi_t^i = 0$. This weight is similar to the weight of the exponential moving average[10]. The only difference is that it depends on historical observations.

B. Summarization Operation

Because each non-sensor node receives attack container from its children and its neighbors to update its own ACIS, the ACIS metadata values have to be aggregatable. Due to the way we define the attack container including severity (Equation

[10]The moving average has the form $\hat{\theta}_t = \alpha X_t + (1-\alpha)\hat{\theta}_{t-1}$. α is similar to $\frac{1}{\beta_{t-1}+1}$ as in (3)

1) and abnormality (Equation 2) values, the summarization operation becomes feasible [11].

C. Early Warning and Hierarchical Cooperative Response Protocol

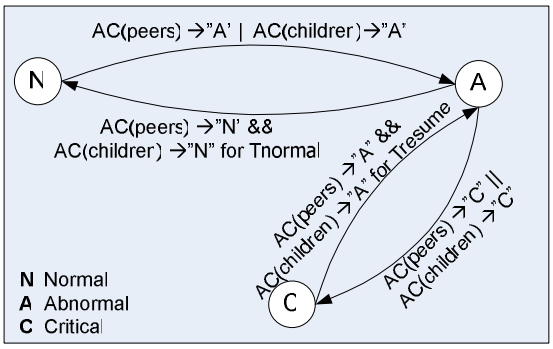


Fig. 5. State diagram of the protocol

We have just defined monitoring and detection operation used to create ACIS and summarization operation used to update ACIS at monitors and aggregators. This section will describe how ACIS are exchanged to form early warning and hierarchical cooperative response.

To enable the hierarchical cooperative response, each non-sensor node i keeps two ACIS: one for its children and the other one for its peers. We denote them as $AC_{children}(i)$ and $AC_{peers}(i)$, respectively.

ACIS Update: Each AC monitor or AC aggregator i, when receiving an ACIS (or a sensor measurement) from a child or a neighbor, will use the CMB Box to detect anomalies. If anomalies are detected, the node calculates how much deviation is from the expected mean. Finally, it updates the abnormality and severity value of $AC_{children}(i)$ and $AC_{neighbors}(i)$.

AC Node state diagram: As mentioned above, an attack container AC contains abnormality $A(i)$ and severity $S(i)$ that are used to detect the abnormal/critical behaviors by using CUSUM box.

Figure 5 shows the state diagram of a AC node. At the system bootstrap stage and in the normal condition, the node stays at "N" state. If either $AC_{children} \rightarrow$ "A" or $AC_{peers} \rightarrow$ "A" (i.e. (A(i), S(i)) moves into abnormal region), the node state transits to "A" state. This is the early warning state of the node. When a node i is in "A" state, it will update the attack container for its peers and parents at the rate $R = R_{max} \max(A(i), S(i))$ where R_{max} is the pre-defined maximum rate that a node allows to send. The intuition for this rate update is that as the abnormality and severity values in a sub-tree get larger, the update rate will be faster to react to the signal of large-scale attacks. R cannot exceed R_{max} because abnormality $A(t)$ and severity $S(t)$ are always less than 1. Also, if $A(t) = 0$ or $S(t) = 0$, R will be a pre-defined rate $R_{default}$.

[11]For numerical values, ACIS will be aggregatable. For textual information or other non-numerical information, values will be summerizable, not aggregatable

If node stays at "A" state and $AC_{children} \rightarrow$ "N" $\&\& AC_{peers} \rightarrow$ "N" for an interval time T_{normal}, node will switch back to state "N". This behavior will help to reduce the update rate for communication efficiency.

When either $AC_{children} \rightarrow$ "C" or $AC_{peers} \rightarrow$ "C", the state transits to "C" state. This is the state where the node starts to react strongly. It will issue the "CONTAINMENT" messages to all of its children. This message will be relayed at each level and eventually reach the healthy sensors who will become immune to the value-changing worm-like attack.

If the AC monitor stays at "C" state and $AC_{children} \rightarrow$ "A" $\&\& AC_{peers} \rightarrow$"A" for an interval time T_{resume}, it will switch back to state "A", issue the "RESUME" command to sensors and updates its parent and peers. This behavior happens when the false alarms happen and monitors try to get sensors resumed to the normal operation.

V. Simulation Study

We modify the simulator used in [16] to evaluate our framework. We discretize time into slots with length $t_{slot}(seconds)$. For each time slot t_{slot}, we simulate the effect of automated upgrade distribution protocols, the vendor movement and ACF framework on sensors.

Goals: The goals of the simulation study are 1) to show that our attack containment framework really helps in terms of speed and false alarms and 2) to show the effect of "abnormal region" and "critical region". Specifically, what the effects of abnormal thresholds $(A1, A2)$ and critical thresholds $(S1, S2)$ are.

Simulation Setup: We evaluate our scheme in a hierarchical network with five levels. Level 0 is sensor level. Level 1 is the monitoring level. The rest are aggregation levels. Nodes at each level have the same number of children and number of peers. Links at each level will also be assigned different delay. The parameters are shown in Table I. We also would like to note that since we are only interested in the speed and false alarms, we will not simulate the transition from "abnormal" to "normal" and "critical" to "abnormal" discussed in section IV-C.

As one might see that finding of optimal threshold setting is a very challenging problem in both simulation and analysis. Therefore, we simplify this problem by representing thresholds as 4 pairs (T_1^i, T_2^i) for each level i, $i = 1..4$. At each level i, T_1^i is used for abnormal threshold (i.e. $A_1 = S_1 = T_1^i$) and T_2^i is used for critical threshold (i.e. $A_2 = S_2 = T_2^i$).

Metrics: We use two metrics for the evaluation. The first metric is the number of infected sensors and the number of immunized sensors. The second metric is false alarm rate.

Simulation of attacks without any containment: Figure 6 shows the simulation of attacks without any containments with various pairs of (K, T_{vendor}) where K is the parameter for K-multicast infection model described in II-B and T_{vendor} is the speed of the vendor. Figure 6 clearly shows that the attack we described in section II-B can be really fast. The containment system must be automatic to deal with this attack.

Parameter	Value	Description
t_{slot}	1/20 (s)	length of a time slot
#Nodes	[5000, 100, 10, 5, 1]	Number of nodes at each level.
#Children	[0, 50, 10, 2, 1]	Number of children of each node at each level.
Children Link Delay	[0, 0.5, 0.5, 0.5, 0.5]	delays of link from parent to children at each level
Peers link weights	[0, 0.5, 0.5, 0.5, 0.5, 0.5]	delays of peer-to-peer link at each level
$R_{default}$	1 msg/sec	Default update rate
R_{max}	10 msgs/sec	Maximum update rate
T_{vendor}	5 seconds	Speed of the vendor II-B

TABLE I

SIMULATION PARAMETERS

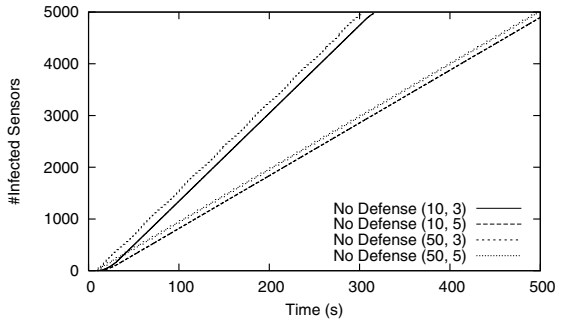


Fig. 6. Infection without containment

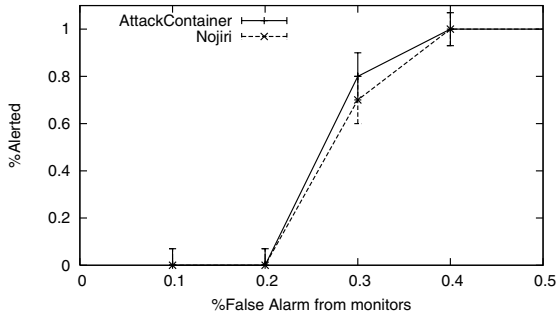


Fig. 9. False Alarm

Effect of Containment: Figure 7(a) shows the typical behavior of our system for threshold setting in {(0.3, 0.5), (0.2, 0.3), (0.15, 0.3), (0.1, 0.3)}. The attack is detected at the early phase and is contained quickly due to the cooperative mechanism. Figure 7(b) further shows that AttackContainer outperforms the mechanism proposed by Nojiri et al. [20] with different thresholds. AttackContainer could immunize more sensors than that of Nojiri's. The reason for faster speed is because hierarchical cooperative response allows faster alert propagation at high-level of the hierarchy.

Effect of Abnormal and Critical Thresholds: In this section, we want to see the effect of abnormal and critical thresholds because they define the "abnormal region" and "critical region" which directly affect the behaviors of the response mechanism.

We choose the threshold setting {(T_1^1,0.7), (0.2, 0.2), (0.15, 0.1), (0.1, 0.1) } and vary the abnormal threshold T_1^1 from 0.1 to 0.7. Figure 8(a) shows the effect of abnormal thresholds. When the threshold is low, the abnormal region is large and therefore the system can collect the evidence of attacks faster, of course, at the cost of message overhead. This explains why the system could immunize more sensors than the cases of high abnormal threshold.

The effect of critical thresholds is similar to that of abnormal threshold, shown in Figure 8(b). We vary T_2^1 in the tuple {(0.3, T_2^1),(0.2, 0.8),(0.1, 0.7),(0.1, 0.6)} from 0.3 to 0.9. Figure 8(b) also shows that critical threshold has a stronger effect than abnormal threshold since it directly affects when the sensors are blocked.

In summary, abnormal and critical thresholds are absolutely important to the behavior of the containment framework. More importantly, Figure 8 shows that our approach of distributed early warning (i.e. abnormal region) truly has effect on the attack containment.

False Alarms: We evaluate the false alarms as follows. We assume that the monitors, the ones that directly detect the abnormal behaviors of sensors, could generate false alarm with probability p_{fa}. We use the same thresholds as in experiment of the effect of containment shown in Figure 7. Figure 9 shows that even when the monitors have false alarm rate $p_{fa} = 0.2$, our scheme could still tolerate them. Furthermore, it also shows that compared to Nojiri's scheme, our ACF framework has a significant faster speed while keeping the false alarms at a reasonable rate. We believe an adaptive adjustment of thresholds at each level, such as the one proposed on [25], will further yield better false alarms. However, we leave this problem for the future work.

VI. CONCLUSION

We have presented attack containment framework for large-scale value-changing attacks in critical infrastructures. The concept of attack container provides the uniform view for each node about the behavior of its group as well as other peer's groups. Hence, attack containment framework enables the convergence of distributed early warning and hierarchical cooperative response mechanism.

We also give an integrated protocol for early warning and cooperative containment mechanism in our ACF framework. The simulation results clearly show that our scheme can alert and contain large-scale attacks under various scenarios.

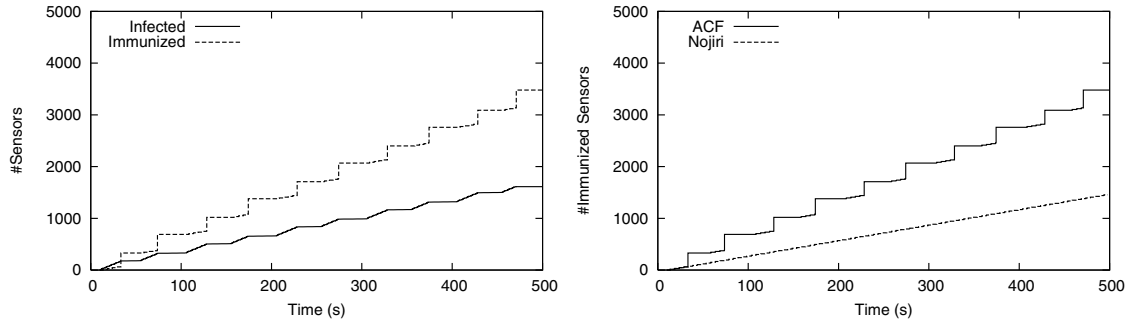
REFERENCES

[1] J. Stamp, J. Dillinger, and W. Young, "Common vulnerabilities in critical infrastructure control systems," *Sandia National Laboratories Report*, November 2003.

[2] "Critical infrastructure protection: Challenges and efforts to secure control systems," GAO-04-354, Tech. Rep.

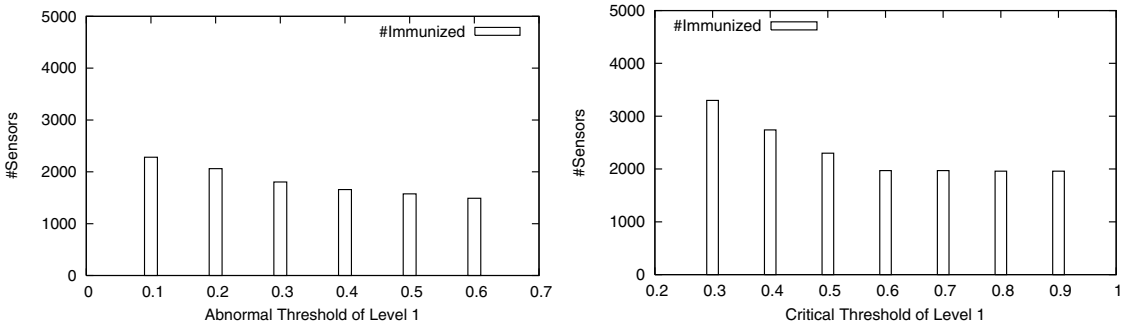
[3] K. Knight, M. Elder, J. Flinn, and P. Marx, "Summaries of three critical infrastructure applications," University of Virginia, Tech. Rep., November 1997.

[4] C. Wang, J. Knight, and M. Elder, "On computer viral infection and the effect of immunization," *ACM Annual Computer Applications Conference*, December 2002.



(a) Containment of ACF (b) ACF and Nojiri peer-to-peer mechanism

Fig. 7. Containment Speed of ACF



(a) Effect of abnormal thresholds (b) Effect of critical thresholds

Fig. 8. Effect of thresholds in ACF

[5] D. Bailey and E. Wright, *Practical SCADA for Industry*. Newnes, 2003.

[6] "Enervista." [Online]. Available: http://www.geindustrial.com/multilin/enervista/launchpad/

[7] T. Nash, "An undirected attack against critical infrastructure," US-CERT, Tech. Rep., 2005.

[8] D. Moore, C. Shannon, and J. Brown, "Code-red: a case study on the spread and victims of an internet worm," in *ACM/USENIX Internet Measurement Workshop*, 2002.

[9] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," in *IEEE Security and Privacy*, 2003.

[10] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Internet quarantine: Requirements for containing self-propagating code," in *INFOCOM*, 2003.

[11] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the internet in your spare time," in *Proceedings of the 11th USENIX Security Symposium*, 2002.

[12] J. O. Kephart and S. R. White, "Directed-graph epidemiological models of computer viruses," *IEEE Computer Society Symposium on Research in Security and Privacy*, 1991.

[13] J. O. Kephart, S. R. White, and Chess, "Computers and epidemiology," *IEEE Spectrum*, May 1993.

[14] Z. Chen, L. Gao, and K. Kwiat, "Modeling the spread of active worms," in *IEEE INFOCOM*, 2003.

[15] S. Staniford, D. Moore, V. Paxson, and N. Weaver, "The top speed of flash worms," in *WORM*, 2004.

[16] C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for internet worms," in *Proceedings of the 10th ACM conference on Computer and communication security*, 2003.

[17] D. Moore, "Network telescopes: Observing small or distant security events," in *USENIX Security*, 2002.

[18] V. Yegneswaran, P. Barford, and D. Plonka, "On the design and use of internet sinks for network abuse monitoring," in *Proceedings of Recent Advances in Intrusion Detection*, 2004.

[19] "Distributed intrusion detection system." [Online]. Available: http://www.dshield.org

[20] D. Nojiri, J. Rowe, and K. Levitt, "Cooperative response strategies for large scale attack mitigation," *DARPA Information Survivability Conference and Exposition*, 2003.

[21] C. G. Senthilkumar and K. Levitt, "Hierarchically controlled cooperative response strategies for internet scale attacks," in *Proceedings of The International Conference on Dependable Systems and Networks*, 2003.

[22] K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, A. D. Keromytis, and D. Li, "A cooperative immunization system for an untrusting internet," *ICON*, 2003.

[23] K. G. Anagnostakis, S. Ioannidis, A. D. Keromytis, and M. B. Greenwald, "Robust reactions to potential day-zero worms through cooperation and validation," in *Information Security Conference (ISC)*, 2006.

[24] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham, "Vigilante: End-to-end containment of internet worms," in *ACM Symposium on Operating Systems Principles (SOSP)*, 2005.

[25] M. Treaster, W. Conner, I. Gupta, and K. Narhstedt, "Contagalert: Using contagion theory for adaptive, distributed alert propagation," in *IEEE International Symposium on Network Computing and Applications (NCA)*, 2006.

[26] D. Lie, J. Mitchell, C. Thekkath, and M. Horowitz, "Specifying and verifying hardware for tamper-resistant software," in *In Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[27] "Trusted computing group." [Online]. Available: https://www.trustedcomputinggroup.org/groups/tpm/

[28] B. E. Brodsky and B. S. Darkhovsky, *Nonparametric Methods in Change-Point problems*. Kluwer Acadamic Publishers, 1993.

[29] H. Kim, B. Rozovskii, and A. Tartakovsky, "A nonparametric multichart cusum test for rapid detection of dos attacks in computer networks," *International Journal of Computing and Information Sciences*, vol. 2, 2004.

[30] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 1, pp. 100–115, 1954.

[31] H. Wang, M.-D. Zhang, and F.-K. G. Shin, "Change-point monitoring for the detection of dos attacks," in *IEEE Trans. Dependable Secur. Comput.*, 2004.