# Correlate the Advanced Persistent Threat Alerts and Logs for Cyber Situation Comprehension

Xiang Cheng, Jiale Zhang, and Bing Chen(✉)

College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, Nanjing 21106, China
cb_china@nuaa.edu.cn

**Abstract.** With the emerging of the Advanced Persistent Threat (APT) attacks, many high-level information systems have faced a large number of serious threats with characteristics of concealment, permeability, and pertinence. However, existing methods and technologies cannot provide comprehensive and promptly recognition for APT attack activities. To address this problem, we propose an APT Alerts and Logs Correlation Method, named APTALCM, to achieve the cyber situation comprehension. We firstly proposed a cyber situation ontology for modeling the concepts and properties to formalize APT attack activities; For recognize the APT attack intentions we also proposed a cyber situation instances similarity measures method based on SimRank method. Combining with instance similarity, we proposed the APT alert instances correlation method to reconstruct APT attack scenarios and the APT log instances correlation method to detect log instance communities. Through the coalescent of these methods, APTALCM can accomplish the cyber situation comprehension effectively by recognizing the APT attack intentions. The exhaustive experimental results show that the two kernel modules, i.e., Alert Instance Correlation Module (AICM) and Log Instance Correlation Module (LICM) in our APTALCM can achieve a high true positive rate and a low false positive rate.

**Keywords:** Cyber situation comprehension · APT attack · Alert correlation · Log correlation

## 1 Introduction

With the rapid advancement of the Internet infrastructure and the widely emerged networking applications, the topological structure of information systems has performed complexity and vulnerability. In this situation, network security management has faced significant challenges. To cope with these increasingly complicated and potential security threaten, various detection techniques have been proposed, like vulnerability detection technology, malicious code detection method, intrusion detection system, etc. Almost all the above technologies are aiming to recognize the security issues existing in the information systems. However, the biggest shortcoming of these methods is that they cannot provide real-time recognition of the real threatens in a


W. Meng and S. Furnell (Eds.): SocialSec 2019, CCIS 1095, pp. 123–138, 2019.
https://doi.org/10.1007/978-981-15-0758-8_10

comprehensive scope, which limiting the ability of network security administrators to make the most responsive decisions. Recently, to solve the above problem, the concept of Cyber Situation Awareness (CSA) [1] has emerged. The main idea of CSA is to recognize the attack activities scattering among a large amount of the noised data and grasp the whole network security situation macroscopically. In this way, the information system can make the responses appropriately and effectively to reduce the damage caused by the various network attacks as possible. Among these powerful network attacks, Advanced Persistent Threat (APT) is a kind of multiple-steps attack with characteristics of concealment, permeability, and pertinence, which has caused serious threats to all kinds of high-level information systems. To mitigate the negative effects of APT, the key problem needs to be solved is designing the cyber situation core technologies which aiming at APT attack recognition, comprehension and prediction.

To address the problems existing in the previous works, we propose an APT Alerts and Logs Correlation Method, named APTALCM, to achieve the cyber situation comprehension. The main contributions can be summarized as follows: (1) we propose the cyber situation ontology for modeling the concepts and properties that are appropriate for cyber situation awareness; (2) then, we introduce a similarity measures method based on cyber situation ontology providing to situation instances correlation; (3) at last, according to the instance type differences (i.e., alert instance & log instance), we present an APT alert instances correlation method to reconstruct APT attack scenarios and an APT log instances correlation method to detect log instance communities to recognize APT attack intentions.

The rest of the paper organizes as follows. Section 2 summaries the related works of Cyber Situation Awareness. Section 3 presents a cyber situation ontology for modeling the concepts and properties to formalize APT attack activities. Then, we introduce a cyber situation instances similarity measures method based on SimRank method. At last, combining with instance similarity, we further propose the APT alert instances correlation method to reconstruct APT attack scenarios and the APT log instances correlation method to detect the log instance communities. Section 4 gives a view of our experiments and analysis. Section 5 presents some conclusions.

## 2 Related Work

The situation is a key factor of CSA means the states of various objects in the cyber systems represented by a set of measurement values. In other words, situation is a global concept and all the objects in the cyber systems are synthesized. Any sole states cannot be regarded as a situation that it focuses on the systematic perspective and relationships between the objects in systems. Cyber situation awareness is a cognitive process applied to cyber systems consisted of three phases. First, the original data generated in the system fused and processed gradually to accomplish the semantics extraction of the system states and activities. Then, the recognition procedure will execute to obtain the exiting cyberspace activities and intentions of abnormal activities in the cyberspace. At last, the representational cyber situations acquired based on the effects of recognizing activities and intentions of abnormal activities in the cyber systems. According to the definition and illustration of CSA, we can summarize the

whole CSA processes into three specific operations: cyber situation perception, cyber situation comprehension, and cyber situation projection. Among these operations, cyber situation perception completes the measurement data fusion, semantics extraction, and activities recognition. Cyber situation comprehension achieves the recognized activities intentions acquisition. Cyber situation projection estimates the threats incurred by the activities intentions within the cyber systems. There are dialectical unification relations between the three phases of CSA, means that they not only depend on each other but also their outputs used respectively for diverse levels of security management requirement.

Attack intention recognition is one of the primary objectives of cyber situation comprehension and our work is mainly focused on the attack intention recognition of APT. Existing works on attack intention recognition mostly focused on attack scenarios reconstruction while ignoring the related non-aggressive activities' contributions to APT attack and multiple-step attack implementation. In addition to this, the intrusion ontology used in the field of intrusion detection cannot be applied to the CSA paradigm directly. Recently, statistical analysis mechanisms proposed to discover the relationships between attack steps while it can only perform on static databases because of these approaches depend on expert knowledge.

At present, the hot topics of cyber situation comprehension focus on two branches: (1) match the alerts with acquired attack activities based on the priori knowledge; (2) analyze the relationships between the alerts without prior knowledge. Cuppens et al. [2] developed the LAMBDA programming to accomplish the description of templates and matching process. The researchers [3–6] usually divide an attack activity into several stages, like IKC Model [7]. At present, there are mainly two types of methods based on similarity measure: The attribute similarity method and timing sequence method. The key of these methods is definition of suitable similarity measure metric. [8, 9] defined a similarity function and clustered the IDS alerts based on the similarities between attributes. [10] proposed an alert correlation method based on Hidden Markov Model to get the attack sequences with highest possibilities.

## 3 APT Alerts and Logs Correlation Method

The basic duty of cyber situation comprehension is analyzing the activities (include attack activities) and recognizing the attack intention. We can acquire the activities in two forms: attack alerts and host logs. As the quantity of activities in the information systems is too large, it is impossible to correlate all the activities in the information systems. Thus, we use the following two benchmarks to improve the efficiency of activities correlation: (1) APT attack alerts are essential to be correlated generating the APT attack scenarios; (2) the logs generated in the hosts infected by APT attacks are essential to be correlated detecting the unaggressive malicious activities. According to these two standpoints, the results of cyber situation comprehension contain APT attack scenarios and log instance communities. Moreover, we propose an APT alerts and logs correlation method (APTALCM) to achieve the cyber situation comprehension. The architecture of APTALCM is shown in Fig. 1.

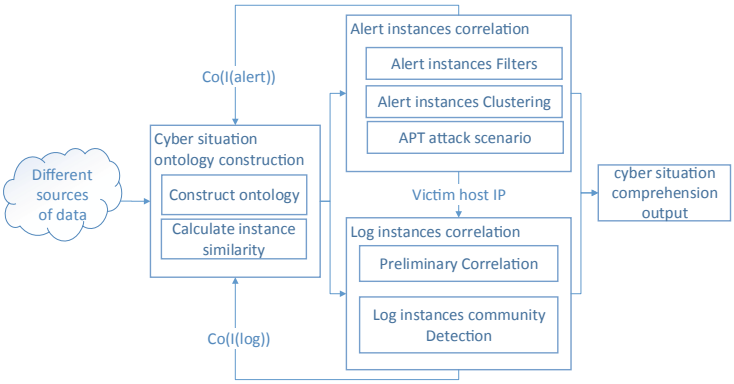


**Fig. 1.** The architecture of APTALCM

Cyber situation ontology is proposed for modeling the concepts and properties of cyber situation awareness paradigm. At present, there are no ontologies which are enough mature to satisfy the requirement of cyber situation awareness. Therefore, in this paper, we proposed APTALCM that defines a set of representational primitives to model the domain of cyber situation as the cyber situation ontology. The input of this phase is APT alerts and logs affected by APT attacks from various detect sensors and the corresponding output is the cyber situation ontology instances.

After the cyber situation ontology construction, the situation ontology instances will face two operation options according to the different instance types (alert instance & log instance). That is, the alert instances raised from the preceding phase will be fed to the alert instance correlation module (AICM) and the log instances will transmit to the log instance correlation module (LICM). Specifically, the aim of the alert instance correlation module is to recognize APT alert instances could be one part of an APT attack scenario. The log instances correlation module enforces on the log instances generate within the victim host to detect the log instance community and recognize the potential malicious activities.

### 3.1 Cyber Situation Ontology Construction

According to the APTALCM architecture, the first module of APTALCM will convert the received APT alerts and host logs into cyber situation ontology instances. Thus, our first work is to propose a formal definition of cyber situation ontology. Different from the previously proposed methods directly send the ontology instances to attack scenarios reconstruction module, we introduce a method based on the SimRank method to calculate the similarity between cyber situation instances.

**Cyber Situation Ontology Initialization.** A widely accepted definition of ontology is shown as follow: the ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or concepts), attributes (or properties), and relationships (or relations between class members).

According to the ontology definition and the combination of the characteristics of cyber situation, we introduce a formalized definition of the cyber situation ontology as follow: $\mathbf{O} = (\mathbf{C}, \mathbf{A}, \mathbf{D}, \mathbf{R}, \mathbf{S})$. The elements $(\mathbf{C}, \mathbf{A}, \mathbf{D}, \mathbf{R}, \mathbf{S})$ represent the set of classes (alert or log), the set of attributes, the domain of the cyber situation ontology, the set of relationships of the instances, and the set of similarity between the instances (alert instances or log instances), respectively. Then, we define $\mathrm{A}(c_i)$ to represent the attributes of a class $c_i$, $\mathrm{I}(c_i)_m$ as an instance of the class $c_i$. Besides, the attribute value of an instance can be represented as $\mathrm{A}(\mathrm{I}(c_i)_m)$ and $\mathrm{SIM}(\mathrm{I}(c_i)_m, \mathrm{I}(c_i)_n)$ presents the similarity between instances $\mathrm{I}(c_i)_m$ and $\mathrm{I}(c_i)_n$.

The APT alert class consists of alert instances converted from APT alert detected by various attack detection sensors and each alert instance represents a suspicious attack step of an APT attack. In this paper, we set seven attributes for APT alert class to analyze the characteristics of APT alerts output from different attack detection sensors: *Timestamp*, *Alert_Type*, *Src_Ip, Dest_Ip*, *Src_Port*, *Dest_Port*, and *Victim_HostIp*. The attributes values of an alert instance stored in a 7-dimensional vector, $\mathrm{A}(\mathrm{I}(alert)_m) = (a_1, a_2, a_3, a_4, a_5, a_6, a_7)$.

The host log class consists of the log instances transformed from the log data providing by the application programs, such as *HTTP*, *Object access*, *Authentication*, *Process create* and *WFP connect*. The intrinsic of transforming the log data to log instances is extracting representative attributes. In this work, we select 19 attributes from the log data such as *Tinestamp*, *Q_Domain*, *R_Ip*, *Pid*, *Ppid*, *Pname* etc. As the data generate from different application programs, not all the log instances have the same attributes, we also give the log instance type. The attributes value of a log instance can be presented as a 19-dimensional vector:$\mathrm{A}(\mathrm{I}(log)_m) = (a_1, a_2 \ldots a_{18}, a_{19})$. If a log instance only contains attributes a1, a3, a5, the other elements of its attribute vector are zero.

**Calculate Instance Similarity.** We proposed a cyber situation instance similarity calculation method to provide a correlation basis for alert correlation module and log correlation module. Each alert or log is an instance of cyber situation ontology and the relationship between them can be presented as a labeled directed graph with similarity. As its graphic character, the proposed method is based on the SimRank mechanism.

The SimRank mechanism provides a similarity measure of structural context where the related objects are connected by directed edges. It defines a recursive function calculating the similarity between object pairs based on the concept of context. The key is objects are similar on the condition that referenced by similar objects.

We measure the similarity between the cyber situation instances, which belong to the same class. In other words, we measure the similarity within the alert class and host log class. To compare the similarity between two cyber situation instances $\mathrm{I}(c)_m$ and $\mathrm{I}(c)_n$ within the same class, we use the following two sets of parameters:

Attributes: The attributes of each cyber situation instance, $\mathrm{A}(\mathrm{I}(c)_m)$ and $\mathrm{A}(\mathrm{I}(c)_n)$

Correlated instances: The instances have already correlated to each cyber situation instance $\mathrm{I}(c)_m$ and $\mathrm{I}(c)_n$ presented as $\mathrm{Co}(\mathrm{I}(c)_m)$ and $\mathrm{Co}(\mathrm{I}(c)_n)$.

The basic similarity measure of cyber situation instances is calculating the similarity between their attributes and the correlated instances. The formalized representation of the similarity between two cyber situation instances is shown in Eq. (1).

$$\mathrm{SIM}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) = \gamma \mathrm{SIMA}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) + \beta \mathrm{SIMCo}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) \tag{1}$$

$$\gamma = \frac{\left|\mathrm{A}(\mathrm{I}(c)_m) \cup \mathrm{A}(\mathrm{I}(c)_n)\right|}{\left|\mathrm{A}(\mathrm{I}(c)_m) \cup \mathrm{A}(\mathrm{I}(c)_n)\right| + \left|\mathrm{Co}(\mathrm{I}(c)_m) \cup \mathrm{Co}(\mathrm{I}(c)_n)\right|}$$

$$\beta = \frac{\left|\mathrm{Co}(\mathrm{I}(c)_m) \cup \mathrm{Co}(\mathrm{I}(c)_n)\right|}{\left|\mathrm{A}(\mathrm{I}(c)_m) \cup \mathrm{A}(\mathrm{I}(c)_n)\right| + \left|\mathrm{Co}(\mathrm{I}(c)_m) \cup \mathrm{Co}(\mathrm{I}(c)_n)\right|}$$

It indicates that $\mathrm{SIMA}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) \in [0, 1]$ and $\mathrm{SIMCo}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) \in [0, 1]$, respectively. The two parameters $\gamma$ and $\beta$ are defined to normalize the impact degree where $\gamma + \beta = 1$. Therefore, we can draw the conclusion that $\mathrm{SIM}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) \in [0, 1]$. Then, we will give the formalized representation of $\mathrm{SIMA}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right)$ in a mutually recursive method. $\mathrm{SIMA}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right)$ measures the similarity between cyber situation instances based on attribute similarity, $\mathrm{SIMA}\left(A_i(\mathrm{I}(c)_m), A_i(\mathrm{I}(c)_n)\right)$ measures the similarity between the attributes of each cyber situation instance. Note that the similarity between same instances is 1 and other conditions can be calculated in Eq. (2).

$$\mathrm{SIMA}\left((\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) = \frac{\partial}{|\mathrm{A}(\mathrm{I}(c))|} \sum\nolimits_{i=1}^{|\mathrm{A}(\mathrm{I}(c))|} \mathrm{SIMA}\left(A_i(\mathrm{I}(c)_m), A_i(\mathrm{I}(c)_n)\right) \tag{2}$$

The similarity between two attributes can be set as 1 on the condition of $A_i\left(\mathrm{I}(c)_m\right) = A_i(\mathrm{I}(c)_n)$. If none of the attributes is the same, $\mathrm{SIMA}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right)$ will be calculated based on Eq. (3) where $\mathrm{Co}_i(\mathrm{I}(c)_m)$ is the correlated instance to $\mathrm{I}(c)_m$. The method of acquiring them will be discussed in Sects. 3.2 and 3.3.

$$\mathrm{SIMA}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) = \frac{\partial \sum_{i=1}^{\left|\mathrm{Co}(\mathrm{I}(c)_m)\right|} \sum_{j=1}^{\left|\mathrm{Co}(\mathrm{I}(c)_n)\right|} \mathrm{SIMA}\left(\mathrm{Co}_i(\mathrm{I}(c)_m), \mathrm{Co}_j(\mathrm{I}(c)_n)\right)}{\left|\mathrm{Co}(\mathrm{I}(c)_m)\right|\left|\mathrm{Co}(\mathrm{I}(c)_n)\right|} \tag{3}$$

The $\mathrm{SIMCo}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right)$ measures the similarity between cyber situation instances based on its correlated instances similarity and calculated in Eq. (4). On the condition that either $\mathrm{I}(c)_m$ or $\mathrm{I}(c)_n$ does not have any correlated instance, we will hardly infer any similarity between them.

$$\mathrm{SIMCo}\left(\mathrm{I}(c)_m, \mathrm{I}(c)_n\right) = \begin{cases} 1 & \mathrm{I}(c)_m = \mathrm{I}(c)_n \\ \dfrac{\partial \sum_{i=1}^{\left|\mathrm{Co}(\mathrm{I}(c)_m)\right|} \sum_{j=1}^{\left|\mathrm{Co}(\mathrm{I}(c)_n)\right|} \mathrm{SIMCo}\left(\mathrm{Co}_i(\mathrm{I}(c)_m), \mathrm{Co}_j(\mathrm{I}(c)_n)\right)}{\left|\mathrm{Co}(\mathrm{I}(c)_m)\right|\left|\mathrm{Co}(\mathrm{I}(c)_n)\right|} & \mathrm{I}(c)_m \neq \mathrm{I}(c)_n \end{cases} \tag{4}$$

### 3.2 Alert Instances Correlation

As APT attack alerts are essential to be correlated generating the APT attack scenarios, we focus on the APT attack scenarios construction. The APT attack usually performs through a few steps with characteristics of persistent, targeted and aiming at the specific object. The ultimate goal of APT is obtaining confidential data in the information systems. To achieve this goal the attack process usually contains complex multi-step. The alert instances constructed by cyber situation ontology construction module belong to different APT attack step, Table 1 summarizes the matchup between APT attack scenario steps and alert instances.

**Table 1.** The matchup between APT attack scenario steps and alert instances

| Step number | APT Step | Alerts instance |
|---|---|---|
| Step 2 | (P) Point of entry | I($p_1$) Domain_instance |
| | | I($p_2$) Disguised_exe_instance |
| | | I($p_3$) Hash_instance |
| Step 3 | (C) C&C communication | I($c_1$) Domain_flux_instance |
| | | I($c_2$) Ip_instance |
| | | I($c_3$) Ssl_instance |
| Step 5 | (A) Asset/data discovery | I($a_1$) Scan_instance |
| Step 6 | (D) Data exfiltration | I($d_1$) Tor_intance |

The first step (Intelligence Gathering) contains some passive process and the corresponding alerts are not easily be detected by network traffic sensors. The fourth step (Lateral Movement) is internal traffic within the information system while the APT alerts are generated based on inbound and outbound traffic. Based on the above facts we only correlate the APT alert instances generated in Step 2, Step 3, Step 5 and Step 6 of an APT attack scenario.

The AICM outputs two kinds of correlated alert instance clusters:$Cluster_{full}$ and $Cluster_{sub}$. $Cluster_{full}$ will be generated when AICM has correlated a full APT attack scenario during the correlation duration has every step of an APT attack scenario. $Cluster_{sub}$ will be produced when AICM has correlated two or three rather than all steps of an APT attack scenario during the correlation duration.

**Alert Instance Filter (AIF).** As the ATP alerts are produced by various detection sensors, the same alert instances have the possibility of generating during a correlation duration. The alert instance filter (AIF) discards the repeated and redundant alert instances. It checks whether the new arriving alert instance has been constructed during the correlation duration through compare the alert instance type and instance attributes value of it with the previous instances. It is obvious that ignoring the invalid alert instances can reduce computation cost of AICM.

**Alert Instance Cluster (AIC).** Alert instance cluster (AIC) module assigns the alert instances those are most similar to a certain cluster. An APT full steps scenario or sub-steps scenario can present as alert instance clusters. As AIC is based on APT attack scenario it restricts to three rules:

*Rule 1.* Alert instances, which belong to the same APT attack step, should not be assigned to the same cluster.

*Rule 2.* The same types of alert instances should not be assigned to the same cluster.

*Rule 3.* The APT attack alert instances trigger time span should be within the correlation duration and alert instances order should be in accord with the APT attack life cycle.

All the generated alert instance clusters are presented as a directed graph; the scattered alert instances are linked by directed edges based on the similarity. Then the clusters will be consumed by the correlation indexing module. Each cluster is consisted of maximum four ordered alert instances and recorded in an instance_cluster_dataset (ICD). When a new alert instance arrives in the AIC module, we first check to which APT attack step it belongs. Based on the different alert instance type AIC operates different options. We have the conclusion: Point of entry (P) which the second step of APT attack scenario is the first detectable attack step. When AIC get an alert instance $\mathrm{I}(p_i)$ it generates a new cluster and sets $\mathrm{I}(p_i)$ to instance_1 recorded in ICD. When AIC get an alert instance $\mathrm{I}(c_i)$, AIC inquires the similarity $\mathrm{SIM}(\mathrm{I}(c_i), \mathrm{I}(p_i))$ where $\mathrm{I}(p_i)$ is the instances already recorded in ICD in the order instance_1 of one existing cluster. AIC adds the $\mathrm{I}(c_i)$ to the order instance_2 of the cluster which not only has the largest value of $\mathrm{SIM}(\mathrm{I}(c_i), \mathrm{I}(p_i))$ but also comply with Rule1, Rule 2 and Rule 3. Then we can get the correlated instances of $\mathrm{I}(c_i)$:$\mathrm{Co}(\mathrm{I}(c_i)) = \mathrm{I}(p_i)$, and send the $\mathrm{Co}(\mathrm{I}(c_i))$ back to the cyber situation ontology construction module for later instance similarity calculation. If there are no suitable alert instances $\mathrm{I}(p_i)$ to be chosen, AIC generates a new cluster and sets $\mathrm{I}(c_i)$ at instance_2 recorded in ICD. The correlation operations of $\mathrm{I}(a_i)$ and $\mathrm{I}(d_i)$ are similar to above. In general, we correlate the alert instance to the most similar prior step alert instance has recorded. A correlation example is shown in Fig. 2.

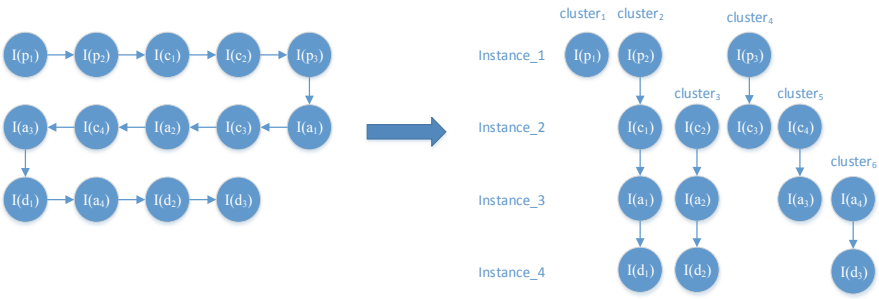


**Fig. 2.** The alert instance clusters construction

**APT Attack Scenario (AAS).** APT attack scenario (ASS) module confirms the alert instances belonging to the same alert instance cluster whether can construct a full or sectional APT attack scenario. As we knew each ATP alert instance cluster is constructed incrementally, it has the possibility that later received alert instance reforms the previously correlated alert instances. To address this problem, we add a parameter $L_{ij}$ to the correlated links between every two alert instances, which belong to the same cluster. The parameter $L_{ij}$ has two values: 1 or 0. When the alert instance_i and instance_j has the same *Victim_HostIP* the $L_{ij}$ is set to 1; otherwise, the $L_{ij}$ is set to 0. The ASS will face four state of $L_{ij}$ during the correlation. The four states and corresponding operations are shown in Fig. 3 and described as follow:

$(1, 1)$: The APT alert instances can belong to a certain AAS.
$(0, 1)$: The latest two alert instances are much more similar than the prior two alert instance. Then the first link should be disconnected and construct a new instance cluster contains the latest two alert instances waiting for coming correlation.
$(1, 0)$ : No evidences can trigger the disconnection, just waiting for later instances.
$(0, 0)$ : No evidences can trigger the disconnection, just waiting for later instances.

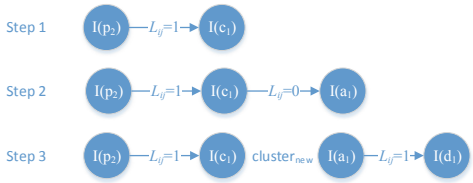


**Fig. 3.** The constructing AAS states evolution

We also introduce a parameter *LinkNum* to check the clustered APT alert instances and discard the uncorrelated alert instances. *LinkNum* can be formulated as follow.

$$LinkNum_{cluster_k} = \sum_{\substack{i=1, j=i+1 \\ instance\_i \in cluster_k \\ instance\_j \in cluster_k}}^{3} L_{ij} \tag{5}$$

$LinkNum = 0$. The APT alert instances in the cluster have no effect and cause with each other; they cannot construct an APT attack scenario.
$LinkNum = 1$. The APT alert instances in the cluster can generate a correlation between two alert instances and they can construct a $Cluster_{sub}$ with two steps.
$LinkNum = 2$. The APT alert instances in the cluster can generate correlations between three alert instances and they can construct a $Cluster_{sub}$ with three steps.
$LinkNum = 3$. The APT alert instances in the cluster can generate a correlation between four alert instances constructing a $Cluster_{full}$ and present an APT attack scenario.

Then, we define an ASS vector to record the information of each correlated clusters. The ASS vector is formally defined as follow:

$$assv_k = (cluster_k, LinkNum, Victim\_HostIp, SimDeg) \tag{6}$$

The $assv_k$ vectors are the sectional output of cyber situation comprehension to be used in future cyber situation projection. Meanwhile the LICM module can get the attribute *Victim_HostIp* from ASS vectors to set the correlation hosts range.

**Module Implementation.** We implement the algorithm of AICM module in C programming language after getting the simulation dataset. The pseudo-code of the AICM module is shown below (Fig. 4).

```
ypedef struct
{
   struct TimeStamp
   char Alert_Type[20];
   char Src_Ip[20];
   char Dest_Ip[20];
   int Src_Port;
   int Dest_Port;
}Attr;// Structure definition
main{
      readtxt()//Read the information
and convert
it into a specified format
      sim （） {//similarity degree
      do{  If ((edge[i][0] == 1 &&
edge[i][1] == 0 && edge[t][0] == 0 &&
edge[t][1] == 1) | | (edge[i][0] == 0 &&
edge[i][1] == 1 && edge[t][0] == 1 &&
edge[t][1] == 0))
     arr[i][j] = C * AB;
else
          arr[i][j] = C * (1.0 + AB) / 2.0;//
Compare each attribute and Complete the
iteration
}end;
}
      Timecmp();//Sort by TimeStamp
      Check_sort();//Group by type
      //Construct the cluster
      Linkcourt();//Calculate the linkcount
      Print();//Oput the cluster
}end
```

**Fig. 4.** The pseudo-code of AICM module

### 3.3 Log Instances Correlation

Advanced Persistent Threat (APT) attack has multiple stages for the sake of being elusive and stealthy. Besides the APT alerts generated during the multiple stages, this type of attack pattern inevitably leaves some log information spatio-temporally dispersed across victim hosts. LICM exploits the fundamental inter connections between logs and detects the log instances communities. The preliminary correlation module constructs the weighted graphs by correlating log instances generated at the victim hosts (The *Victim_HostIps* are got AICM). LICM can also discover the log instance communities hid within the weighted graphs by log instances community detection module.

**Preliminary Correlation (PC).** The input of preliminary correlation module (PC) is the log instances extracted from log data and the outputs are directed and weighted graphs. In this work, we regard the log instances as nodes and regard the relationships between them as edges. The weights of edges illustrate the similarity between log instances and the directions of edges illustrate the effect and cause relationship between them. Once the LICM gets an ASS vector from the AICM it begins to construct a preliminary correlated graph based on the log instances generated at the victim host. Some preprocessing may leave logs before the significant APT alerts occur, so we correlate the log instances generated before the first alert instance for a short time of $\tau$. As the log instances generated in a timing sequence, preliminary correlation module inquires the similarity from the cyber situation ontology construction module in the same strategy. For example, at a victim host $\mathrm{I}(log)_1$ is the first log instance generated within the *CorrelationDuration*. To construct a weighted and directed graph the similarity between the log instances are regarded as weights according to the following strategy:

$\left[\mathrm{I}(log)_1, \mathrm{I}(log)_2, \mathrm{I}(log)_3, \ldots, \mathrm{I}(log)_n\right]$ is a log instance sequence generated according to timing sequence. When PC module gets a new log instance $\mathrm{I}(log)_i$ by inquiring the $\mathrm{SIM}\left(\mathrm{I}(log)_1, \mathrm{I}(log)_i\right)$, $\mathrm{SIM}(\mathrm{I}(log)_2, \mathrm{I}(log)_i)$,$\mathrm{SIM}(\mathrm{I}(log)_3, \mathrm{I}(log)_i)$,…,$\mathrm{SIM}(\mathrm{I}(log)_{i-1}, \mathrm{I}(log)_i)$ from the cyber situation ontology construction module. On the condition that $\mathrm{SIM}(\mathrm{I}(log)_i, \mathrm{I}(log)_j) \neq 0$, create a directed edge from $\mathrm{I}(log)_i$ to $\mathrm{I}(log)_j$ set $\omega_{ij} = \mathrm{SIM}(\mathrm{I}(log)_i, \mathrm{I}(log)_j)$, generate a correlated instance of $\mathrm{I}(log)_j$ set $\mathrm{Co}_n\left(\mathrm{I}(log)_j\right) = \mathrm{I}(log)_i$ and pass back $\mathrm{Co}_n\left(\mathrm{I}(log)_j\right)$ to cyber situation ontology construction module. Otherwise, $\omega_{ij} = 0$. There are no correlated relationships (edges) between $\mathrm{I}(log)_i$ and $\mathrm{I}(log)_j$.

The preliminary correlation module acquires the weighted graphs of log instances and sends them to log instance community detection module for acquiring the cyber situation comprehension output.

**Log Instance Community Detection (LICD).** Taking account for the log instance graphs scale log instance community detection module has the demand of proposing an efficient community detection method to extract the log instance communities from the intricate directed and weighted correlated instance graph. Comparing the existing diverse machine learning method used in communities detection, LICD module owns a log instance community detection method based on the Louvain method, which has the advantage of managing large-scale nodes networks.

At the initial phase of LICD, each log instance in the graph constructed from PC module represents a single log instance community. $\sum_m \omega_{im}$ and $\sum_m \omega_{mj}$ respectively presents the totality weights added on edges associate to log instances $\mathrm{I}(log)_i$ and $\mathrm{I}(log)_j$,$c_{\mathrm{I}(log)_i}$ and $c_{\mathrm{I}(log)_j}$ respectively represents the log instance community $\mathrm{I}(log)_i$ and $\mathrm{I}(log)_j$ belong to. The $\theta - function$ in the LICD module is used to separate the log instances which belong to the different log instance communities, means that

$\theta(i,j) = 1$ if $i = j$, $\theta(i,j) = 0$ otherwise. To compare the density degree of the correlations within the log instance communities with the correlations cross the log instance communities LICD introduces an evaluation index *DenDeg* defined as follow.

$$DenDeg = \frac{\sum_{\mathrm{I}(log)_i,\mathrm{I}(log)_j}\left[\omega_{ij} - \frac{\sum_m \omega_{im}\sum_m \omega_{mj}}{\sum_{\mathrm{I}(log)_i,\mathrm{I}(log)_j}\omega_{ij}}\right]\theta\left(c_{\mathrm{I}(log)_i}, c_{\mathrm{I}(log)_j}\right)}{\sum_{\mathrm{I}(log)_i,\mathrm{I}(log)_j}\omega_{ij}} \tag{7}$$

As soon as the LICD module finishes the log instance initialization, it repeats actions to optimize the index *DenDeg* in the following strategies: for each log instance $\mathrm{I}(log)_k$, shift $\mathrm{I}(log)_k$ from its attached log instance community $c_{\mathrm{I}(log)_k}$ into its correlated log instance $\mathrm{Co}_n\left(\mathrm{I}(log)_k\right)$ attached communities. LICD evaluates the value change of index *DenDeg* and places $\mathrm{I}(log)_k$ into the log instance community $c_{\mathrm{I}(log)_k\_in}$, which has the most obvious index *DenDeg* increase. If the maximum increase is not positive, the log instance $\mathrm{I}(log)_k$ will not be shifted from its original log instance community. LICD applies this process repeatedly and sequentially to each log instance until no $\Delta DenDeg$ occurs, gets the detected log instance communities as the segmental output of cyber situation comprehension.

**Module Implementation.** We implement the algorithm of LICM module in Python after getting the log data. The LICM module pseudo-code is provided as follow (Fig. 5):

```
#define MAX_VERTEX_NUM 20
typedef struct ArcBox
{   int tailvex,headvex;
   struct ArcBox *hlink,*tlink;
   InfoType *info;
}ArcBox; //The struct of edge definition
 typedef struct VexNode
{   VertexType data[20];
   ArcBox *firstin,*firstout;
}VexNode; //The struct of log nodes definition
 typedef struct
{   VexNode xlist[MAX_VERTEX_NUM];
   int vexnum,arcnum;
}OLGraph; //The struct of graph definition
//Three main algorithms
get_attributes() //Convert the log data into log instance
{         key_word log[6]; //Five kinds of log data
          init_keyword(); //Initialization of the log data
          while()
          { getline();//Read the log data
          find_keyword()
          { if(key_word in log)// Set the attribute vector value
                  {set 1;
                  }
            else set 0;
          }
          }
          return  A[20];//Output the log instances
}

simrank() //Calculation the similarity degree
{  calc_similarity();
          init_graph(); // Initialization of the graph
          while(new  node  occur)//When  new  nodes  arrive,
renew the graph
          {  ergodic_graph();
                  calc_similarity();
                  change_graph();
          }
          print(); //Output the result
}

louvain()//Clustering analysis
{     calc_modularity(){
            for(each node) // Evaluation the clustering degree
                  {
                   get_neighbor();
                   calc_modularity();
                  }
          }
          if(new modularity occurs) // Iterative computations
            calc_modularity();
          else
            print();
}
main()
{
          get_attirbutes();
          simrank();
          louvain();
}
```

**Fig. 5.** The pseudo-code of LICM module

# 4 Experimental Evaluation of APTALCM

## 4.1 Evaluation of the Alert Instance Correlation Module

As there is no available public data set can provide enough APT attack alerts, we adapt to construct a specialized simulation data set. The function of the alert instance correlation module is to recognize various alert instances could belong to a certain APT attack scenario. To significantly evaluate the AICM module, the simulation data set consists of APT alerts belong to APT attack scenarios and other general alerts do not belong to APT attack scenario. The aim of this experiment is to verify whether the AICM module can reconstruct the APT scenarios hidden in the constructed data set.

**Data Generation.** To construct the simulation data set we use Python to write a script, which constructs two classes of alert: Correlative alerts be part of a $Cluster_{full}$ or $Cluster_{sub}$; scattered alerts do not belong to any of the alert instance cluster.

We set seven attributes to each alert: *Alert_Type*, *Timestamp*, *Src_Ip*, *Dest_Ip*, *Src_Port*, *Dest_Port* and *Victim_HostIp*. To guarantee the randomness of the generated alert we select the *Alert_Type* from the provided eight ATP alert type there in before. We assign a random value start from 01 February 2019 00:00:01 to 30 March 2019 23:59:59 to *Timestamp*. The *Src_Ip* value is assigned based on the selected *Alert_Type*.

The *Dest_Ip* assigned randomly with an IP address on an enterprise network. We select the *Src_Port* randomly from the 49 140 to 65 521 range which is usually allocated dynamically to initiate a connection. We assign a random port number to *Dest_Port* based on the *Alert_Type*. The *Victim_HostIp* is assigned randomly with an IP address on an enterprise network. We have generated 5000 APT alerts for the simulation data set consisted with 150 $Cluster_{full}$, 150 $Cluster_{sub}$ and 4000 random isolated alerts. Then, the simulation data set for evaluate the AICM module has accomplished.

**Correlation Performance.** We have applied the AICM module algorithm on the constructed simulation data set. The correlation result is shown in Table 2. We choose the False Positive Rate (FPR) and the True Positive Rate (TPR) as the correlation effect measurement parameters. We can find the TPR of the two steps $Cluster_{sub}$ is higher than any other clusters. It is obvious that the TPR is lower with the alert instance cluster steps quantity increaser. This is mainly because more alert instances correlation process will increase the possibility of the random isolated alert instances to be unexpected correlated. When decreasing the TPR, the unexpected random isolated alert instances correlation can also incur the False Positive correlation result. As the larger step

**Table 2.** Evaluation AICM module correlation result.

| APT attack cluster | Correlated quantity | FP | TP | N | P | FPR | TPR |
|---|---|---|---|---|---|---|---|
| APT two steps cluster | 2*83 | 2*35 | 2*48 | 4900 | 100 | 1.4% | 96% |
| APT three steps cluster | 3*106 | 3*19 | 3*87 | 4700 | 300 | 1.2% | 87% |
| APT full scenario | 4*121 | 4*10 | 4*120 | 4400 | 600 | 0.9% | 80% |
| Total APT cluster | 968 | 167 | 837 | 4000 | 1000 | 4.2% | 83.7% |

quantity of the clusters the stronger ability they equipped to amend the previous correlation by the later alert instances, the FPR is lower with the alert instance cluster steps quantity increaser. In general, holistic TPR and FPR are well satisfied.

**Performance Comparison Between AICM and Existing APT Detection Systems.** We compared the performance of the developed AICM module with the three typical reconstruction methods of the APT scenarios to show the advantage of our method. The comparative results are shown in Table 3.

**Table 3.** The comparative results between AICM and other APT scenarios reconstruct method.

| APT scenarios reconstruct method | Efficiency | Step quantity | FPR | TPR |
|---|---|---|---|---|
| AICM | Real-time | Four steps | 4.2% | 83.7% |
| Spear phishing based | Real-time | One steps | 15.9% | 94.3% |
| TerminAPTor | Real-time | Four steps | 25.64% | 98.8% |
| C&C-based | Off-line | One steps | 1.2% | 79.6% |

We can get the obvious results that other three proposed APT scenarios reconstruct method are not able to handle the problem that how to balance the higher TPR and lower FPR properly. As the only method can approximately get to the performance of AICM, the C&C-based method has a disadvantage of failing to accomplish the real-time APT scenarios reconstruct. It is nice to see the AICM has higher TPR with not too high FPR.

### 4.2 Evaluation of the Log Instance Correlation Module

We implement the algorithm of LICM in Python and take full advantage of the convenience of package python-Louvain to accomplish log instance community detection.

The experiment environment described as follow: (1) a victim Windows 10 64-bit operating system running on a host with an Intel Core i5-7200u 2.0 GHz CPU, 8 GB RAM. (2) an attack Windows 10 64-bit operating system running on a host with an Intel Core i7-8550u 2.53 GHz CPU, 16 GB RAM. We also assign extra roles to the attack host: the FTP server, C&C server and Apache server.

**Data Generation.** To construct the log data set and evaluate the LICM module algorithm we record the log data from the log providers and the log data quantity of each provider shown as Table 4. We record the log data while using the experimental machine by routinely work without perceiving that some operations have triggered some APT attack activities. We get these log data after a particular attack has launched on the computer system to simulate the APT attack scenario.

**Correlation Performance.** We have applied the LICM module algorithm on the recorded log dataset to evaluate the performance of log community detection on 7 APT attack scenarios such as Attack on Aerospace (AA), Hacking Team (HT), Tibetan and HK (TH), Russian Campaign (RC), Op-Tropic Trooper (OTT), APT on Taiwan (AT), and Op-Clandestine Fox (OCF). We also choose the FPR and TPR as the correlation

**Table 4.** APT log instance size.

| Log | Quantity | Size (KB) |
|---|---|---|
| HTTP | 3345 | 16530 |
| Object access | 282 | 5789 |
| Process create | 261 | 6420 |
| DNS | 510 | 72 |
| WFP | 734 | 18453 |

effect measurement parameters. The log instances within any detected community are regarded as malicious ones and the others as benign ones. The TRP is the portion of the correctly classified malicious log instance or benign log instance. The FPR is the portion of the actual benign log instances unexpectedly classified to the malicious cluster. From the results shown in Fig. 6, we can see that TPRs of LICM module works well on the 7 typical APT scenarios and the FPRs are also medium.

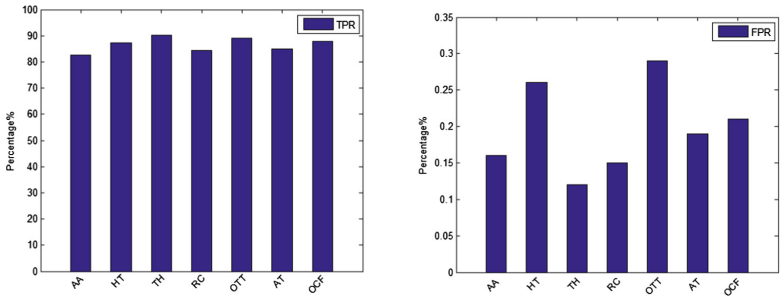


**Fig. 6.** Evaluation LICM module detection result

## 5 Conclusion

In this paper, we proposed an APT alerts and logs correlation method to accomplish the cyber situation comprehension. To recognize attack intentions, a similarity measures method based on SimRank is proposed. We also proposed an APT alert instances correlation method to reconstruct APT attack scenarios and an APT log instances correlation method to detect log instance communities. The experimental results show that the APTALCM has higher TPR with acceptable FPR.

## References

1. Bass, T.: Intrusion detection systems and multisensor data fusion: Creating cyberspace situational awareness. Commun. ACM **43**(4), 99–105 (2000)
2. Cuppens, F., Ortalo, R.: Lambda: a language to model a database for detection of attacks. In: Proceedings of the 3rd International Workshop on Recent Advances in Intrusion Detection (RAID 2000), Toulouse, vol. 1907, pp. 197–216 (2000)

3. Bhatt, P., Yano, E.T., Gustavsson, P.M.: Towards a framework to detect multi-stage advanced persistent threats attacks. In: Proc. of the IEEE Intel Symposium on Service Oriented System Engineering, Toronto, pp. 390–395 (2014)
4. Roschke, S., Cheng, F., Meinel, C.: A new alert correlation algorithm based on attack graph. CISIS **6694**(11), 58–67 (2017)
5. Albanese, M.: Subrahmanian vs. scalable detection of cyberattacks. CISIM **245**, 9–18 (2016)
6. Mathew, S., Upadhyaya, S., et al.: Situation awareness of multistage cyber attacks by semantic event fusion. In: Proceedings of the Military Communications Conference, London, pp. 1286–1291 (2018)
7. Aleroud, A., Karabatis, G., et al.: Context and semantics for detection of cyber attacks. Int. J. Inf. Comput. Secur. **6**(1), 63–92 (2014)
8. Hutchins, E.M., et al.: Intelligence driven computer network defense informed analysis of adversary campaigns intrusion kill chains. In: Proceedings of the ICIW, Chicago, pp. 113–127 (2011)
9. Julisch, K.: Clustering intrusion detection alarms to support root cause analysis. ACM Trans Inf. Syst. Secur. **48**(4), 443–471 (2016)
10. Ourston, D., et al.: Applications of hidden Markov models to detecting multi-stage network attacks. In: Proceedings of the Hawaii International Conference on System Sciences, Hawaii, pp. 73–76 (2016)