Flow-based Anomaly Detection Using Access Behavior Profiling and Time-sequenced Relation Mining

Weixin Liu[1], Kangfeng Zheng [1], Bin Wu [1], Chunhua Wu[1], Xinxin Niu[1]
[1] Information Security Center, Beijing University of Posts and Telecommunications
Beijing 100876, China
[e-mail: jack18jack@gmail.com]
*Corresponding author: Weixin Liu

Received December 20, 2015; revised April 3, 2015; revised May 7, 2016; accepted May 16, 2016; published June 30, 2016

Abstract

Emerging attacks aim to access proprietary assets and steal data for business or political motives, such as Operation Aurora and Operation Shady RAT. Skilled Intruders would likely remove their traces on targeted hosts, but their network movements, which are continuously recorded by network devices, cannot be easily eliminated by themselves. However, without complete knowledge about both inbound/outbound and internal traffic, it is difficult for security team to unveil hidden traces of intruders. In this paper, we propose an autonomous anomaly detection system based on behavior profiling and relation mining. The single-hop access profiling model employ a novel linear grouping algorithm PSOLGA to create behavior profiles for each individual server application discovered automatically in historical flow analysis. Besides that, the double-hop access relation model utilizes in-memory graph to mine time-sequenced access relations between different server applications. Using the behavior profiles and relation rules, this approach is able to detect possible anomalies and violations in real-time detection. Finally, the experimental results demonstrate that the designed models are promising in terms of accuracy and computational efficiency.

Keywords: flow analysis, intrusion detection, behavior profiling, time-sequenced correlation, anomaly detection

http://dx.doi.org/10.3837/tiis.2016.06.018 **ISSN : 1976-7277**

1. Introduction

Company networks and their complexity have been evolving rapidly over the past decade, and emerging threats raise a global concern about security issues. More and more sophisticated and multi-stage attacks are uncovered by security analysts, such as Operation Aurora[1] and Operation Shady RAT[2], which are also known as Advanced Persistent Attacks (APT). Intruders use zero-day vulnerabilities and social engineering techniques to accomplish a set of stealthy and continuous hacking processes. However, traditional security systems, which mostly depend on static signatures, are difficult to keep up with the changing threat landscape and dynamic environment. In addition, due to the lack of knowledge about internal network behavior patterns, intrusion detection systems are incapable of distinguishing malicious insiders from benign ones.

Therefore, it is necessary for intrusion detection systems to keep track of overall network activities and create profiles for network applications. Network flow, which represent by nature aggregated information, is a scalable approach of passive network monitoring and behavior analysis in high-speed networks [3]. Flow-based behavior profiling is a promising approach to detect traffic anomalies and protect network from unknown exploits [4]. Thus, this research's proposal is to create an autonomous flow-based network monitoring system capable of identifying the normal behavior of network applications and detecting anomalies in enterprise network.

The entirety of this research is accomplished through the analysis of flow features. The proposed system is divided into three parts: autonomous discoverying server appications, access behavior profiling and access relation mining. In the first application discovery phase, we extract source and destination ip/port to distinguish clients from server applications, and convert flows into access flows towards server applications. Based on access flows, six flow features (the quantitative attributes (bits and packets in two directions, flow duration and number of flows between client and server application in analysis interval) and two tag features (flow direction and occurrence period) are selected to create access behavior profile for each individual application. Many experts today believe that the disregarded relationships between applications are the major weak spot abused by attackers to compromise systems[5], thus we generate relation rules from frequently related access behavior of different server applications. Based on the previous behavior profiles and relation rules, this approach is able to detect deviation from historical network behavior.

The key contributions of this article are as follows: First, we provide an applicable method to discover active server applications without pre-defined knowledge, and create behavior profiles for each server application by applying a novel linear grouping algorithm PSOLGA. PSOLGA is a PSO-based [6] clustering algorithm, with better grouping stability and time complexity than LGA [7].In addition, we use in-memory graph model to establish anomaly detection rules from time-dependent access flows, such as clients→ web server→ database. To evaluate the proposed system, a variety of tests is performed using simulation data and real-world data from an enterprise network.

The remainder of this paper is organized as follows: Section 2 reviews prior literatures related to flow analysis and anomaly detection. Methodology and major algorithms are given in Section 3. Section 4 explains the two access models and corresponding anomaly detection

approaches. The results of evaluation are presented in Section 5. Finally, Section 6 concludes this article.

2. Related Work

Flow analysis is broadly used in large-scale network behavior profiling[8-10] ,Qos optimization[11-13] and intusion dectection[14, 15]. As is shown in [16], netflow is utilized to profile block-level network activities, as well as track and quantify changes in blocks. Gilberto[8] introduces a profile-based anomaly detection system PCADS-AD which is able to detect DDoS and Flash Crowds by using PCA. In [10], the authors developed a flow-based anomaly detector by using ANN-based classifier and selective sampling.

Clustering is often used as an unsupervised techniqe to discover traffic patterns [17]. A netwok access control machanism, basing on X-means[18] and majority voting, was studied by Frias-Martinez. As an important evolutionary computation technique, PSO[6] is applied to many subject areas such as clustering optimization and multiobjective optimization. LI et.al. [19] uncovered the host members of Botnet in the organizational network by using a combination of PSO and Kmeans. Although K-means algorithm may be deemed as the most important flat clustering algorithm due to its simplicity, it has serveral drawbacks, such as uncapable to deal with non-spherical data. In our pratical experience, network behavior of many server application is distributed in linear strutures, thus K-means may not be our best choice of clustering algorithm. Linear Grouping Algorithm(LGA), first introduced by Van Aelst[7] in 2006, can be useful for investigating subsets that follow different linear relationships in data sets. Garcia[20, 21] introduced Robust LGA to obtain better grouping results against outliers in 2009, by optimizing LGA through trimming methology. In our approach, we build behavior profiles from historical normal access flows, where no observation should be considered as an outlier, thus trimming is not applicable.

Regarding event correlation and self-learning system model, Friedberg et al.[22] aimed to detect anomalies by generating rules from log-information. As the authors assume no prior knowledge about the structure of event logs, all different combination of log atoms should be extracted for covering potential hypothesises H, and additional refinement is required to drop the trival ones.

The proposed approach distinguishes from the abovementioned literatures due to auto discovery of server applications. Besides that, while previous researches focused on detecting major changes and anomalies for overall network behavior, such as DDoS, Scan and Trojan activities[8, 9, 17], our approach is able to build profiles for different server applications in passive monitoring and detect deviation from historical behavior of the specific server application, such as illegal data dump and malicious insider activities. Moreover, this paper contributes by using PSOLGA to find best groups of access behavior towards individual server application. We are first to apply PSO in linear grouping algorithm, optimizing the grouping stability and time complexity.And then, we use the grouping result to generate anomaly alarms in real-time detection.

In addition, compared to [22],our rule generation of correlated access flows avoid additional rule refinement due to the clear struture of flow attributes and the application of in-memory graph model. Besides that, in Friedberg's approach, only situation that the condition event does not triger implication event is considered anomaly. However, it failed to consider the situation that the implication event occurs without the preceding condition event,

which should be included in anomaly detection for completeness. Thus, we introduce two different conditional probability for evaluating both of the previously mentioned situations.

3. Methodology and Background

As is shown in **0**, we collect netflow data from all switches and routers of a small enterprise network, which is divided into a DMZ zone and an intranet zone. Two web servers F and G are deployed in DMZ zone, which are exposed to external users. In intranet zone, there are two different databases(Mysql and Redis) for web servers(F and G) , two HDFS(Hadoop Distributed File System) nodes for storing flow records, and other internal rserves. Two group of external users are used for evaluation, one of which is labeled as normal users(A,B,C) and the other as attackers(D and E). The recording for flow dataset last for 140 hours, of which the first 136 hours for training phase and no attacks are injected. Attackers launch their attacks from the 137th hour. Our system model is presented in **0**. Historical flow traces are automatically converted to access flows and ingested by the two models:single-hop access profiling model and double-hop access relation model. The single-hop model extracts access behavior profile of each specific server application from historical access flows,while the double-hop model generates relation rules between access flows towards different server applications. Both the behavior profiles and relation rules are applied in realtime anomaly detection afterwards.

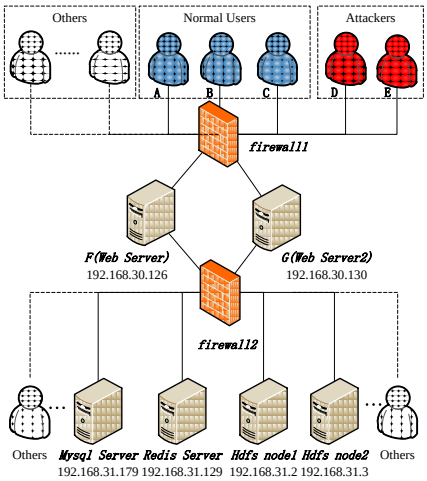


Fig. 1. Evaluation evironment

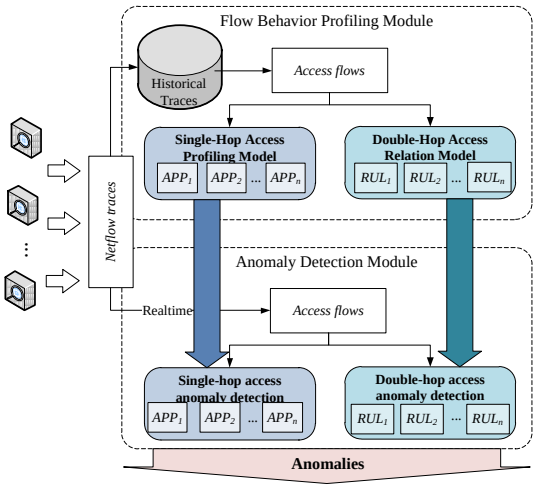


Fig. 2. System model

3.1. Automatic server discovery and access flows

Few studies have placed attention on how to identify servers in flows, while routers and switches mostly export unidirection flows, such as Netflow. Some efforts have been made to identify client/server basing on packet-level analysis[23], or to convert unidirectional flows to bi-directional flows[24]. In order to adapt to any flow export protocol, we develop a methodology for converting input flows to bidirectional flows towards server side, without packet-level attributes or pre-defined server ports. As different configuration of flow caching timeout may break up long-lived flows into fragments of different duration, we merge input raw flows into $flow\_feature\_set_T$ to ensure source/destination key are unique in each analysis interval T. $flow\_feature\_set_T$ is the aggregated form of input flows, which can be derived from any type of flow protocols. *pkt_cnt_in*, *byte_cnt_in*, *pkt_cnt_out* and *byte_cnt_out* are the sum

of corresponding attributes in all associated packets with the same specific key (*proto*, *ip_src*, *port_src*, *ip_dst*, *port_dst*) in two directions. *start_time* is the minimum start time and *end_time* is the maximum end time for packets in a flow.

Definition 1. $flow\_feature\_set_T$ = {**key:** (*proto, ip_src, port_src, ip_dst, port_dst*), **value:** (*port_dst, pkt_cnt_in, byte_cnt_in, pkt_cnt_out, byte_cnt_out, start_time, end_time*)}

Definition 2. $OED[ip,port]$ = #(*unique pairs of* ip_src_i *and* $port\_src_i$) + #(*unique pairs of* ip_dst_j *and* $port\_dst_j$) | $i, j, \epsilon N$, $ip= ip\_dst_i= ip\_src_j$, $port= port\_dst_i= port\_src_j$, N is the totoal count of $flow_feature_set_T$s

$OED[ip,port]$ is denoted as the opposite-end divergence of specific pair of *ip* and *port*. Under the assumption "clients always appear with multiple IPs and random source ports, while servers mostly use unique set of IPs and listening ports", opposite-end divergence of server-side pairs of ip/port are more likely larger than that of client-side pairs, which is also proved in real-world network traces we captured. Thus, we are able to distinguish servers from clients in historical flow data via comparing opposite-end divergence of *ip_src/port_src* pair and *ip_dst/port_dst* pair.

Definition 3. $access\_flow\_feature\_set_T$= {**key**: (*proto, ip_server, port_server, ip_client,* $t_{interval}$), **value**: (*pkt_cnt_to, byte_cnt_to, pkt_cnt_from, byte_cnt_from, flowscount,* $start\_time_{min}$, end_time_{max})}

Based on the auto-discovered servers, we aggregate $flow\_feature\_set_T$ into bidirectional access flows $access\_flow\_feature\_set_T$ from client towards server within a certain interval *T*. We merge flows with the same pair of (*proto, ip_server, port_server, ip_client,* $t_{interval}$). $t_{interval}$=*start_time*/*T*, denoting the time slot flow occurs in. *pkt_cnt_to* and *byte_cnt_to* are #packets and #bytes towards server side of a specific key, and *pkt_cnt_from* and *byte_cnt_from* are for the opposite direction. $start\_time_{min}$ and end_time_{max} are the minimum *start_time* and the maximum *end_time* within $t_{interval}$. *flowscount* is #flows from the client-end to server-end within $t_{interval}$.

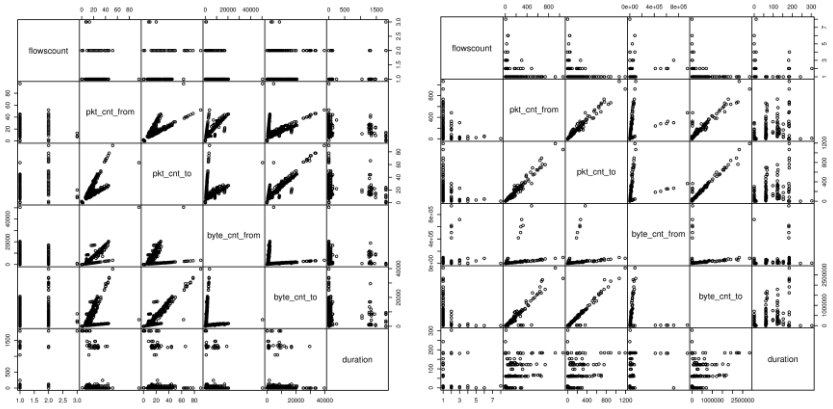


Fig. 3. scatter plots of hdfs control exchange(left) and web server(right)

Euclidean Distance is the most common choice in clustering flow behavior[19, 25, 26], as authors usually assume data is ditributed in spherical structures. However, we have discovered that most serverside applications constrain the size or type of their returning content of different requests and linear structures dominates our feature space in flow monitoring. For expample, web servers return limited textual content, hypertexts or multimedia content when

normal clients request for online articles, web links or personal photos. Clients commonly establish limited flows towards servers within a certain interval. **0** shows the multi-dimensional visualization of normal access behavior of two different server applications(HDFS control message exchange and web server), which is a scatterplot matrix for pairs of each two different dimensions. It is obvious that access flows follow a certain set of linear grouping strutures. Thus we are inspired to use linear grouping algorithm to cluster behavioral features of access profiles.

3.2 PSOLGA algorithm

LGA combines ideas from principal components, clustering methods and resampling algorithms, with the objective to find the grouping result with the minimal sum of square regression residuals(ROSS). Square regression residual is the square distance between a point and its associated hyperplane, measuring how far a point lies from this hyperplane. Resampling is the key of LGA to search for the best grouping result, which needs to take enough starting samples. Van Aelst offered a funtion to calculate *m*[7] as the minimal number of starting values, which is sometimes insufficient to guarantee the fittest result.

Particle swarm optimization(PSO)[6] is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It is broadly used in optimizing classification[27] and clustering approaches[28, 29]. In this section ,we propose a novel linear grouping method PSOLGA with a combination of PSO and LGA, which is able to optimize the resampling process in LGA and output more stable grouping result.

The fundamental expression of PSO is as follows:

$$V_i^{(t)} = w \times V_i^{(t-1)} + c_1 \times r_1 (P_i - X_i^{(t-1)}) + c_2 \times r_2 (G - X_i^{(t-1)}) \quad (1)$$

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)} \quad (2)$$

LGA has five major steps: scaling, generation of the starting values, initializaiton of the groups, iterative refinement and resampling[7]. Considering a data set of size n in d dimensions, LGA need to generate several starting sample groups, each of which contains $k \times d$ points(k is the desired number of clusters). We consider each starting sample group as a particle in $k \times d \times d$ dimensions. $X_i^{(t)}$ denotes the starting position in each swarm iteration. *Location_LGA_Iteration* function is introduced to fundamental PSO, which means paritcles fly in both location iteration and global swarm iteration. For each particle in *Location_LGA_Iteration*, initializaiton of the groups and iterative refinement are firstly finished as in LGA, and then new samples of d-subsets are taken from each of the local final k groups, which are formulated to $XE_i^{(t)}$. $XE_i^{(t)}$ is the ending position in each location iteration. Taking samples of d-subset from a specific ouput group increases the convergence towards the fittest hyperplane, as they have already been assigned to the same linear group. We modify Eq.(1-2) into Eq.(3-5):

$$V_i^{(t)} = w \times V_i^{(t-1)} + c_1 \times r_1 (P_i - XE_i^{(t-1)}) + c_2 \times r_2 (G - XE_i^{(t-1)}) \quad (3)$$

$$XE_i^{(t)} = Local\_LGA\_Iteration(X_i^{(t)}) \quad (4)$$

$$X_i^{(t)} = XE_i^{(t-1)} + V_i^{(t)} \quad (5)$$

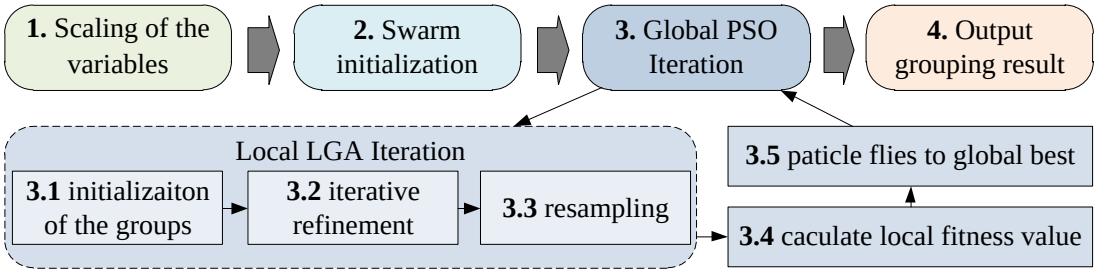


Fig. 4. PSOLGA workflow

As is shown in **0**, PSOLGA is described as follows:

Step 1. **Scaling of the variables**. Considering x_j is one of the totoal n observations, and $x_j[l]$ denotes the value of x_j in the lst dimension, where $j \in [1,n]$, $l \in [1,d]$.The scalabe expression of $x_j[l]$ is caluated as follows:

$$x'_j[l] = (x_j[l] - x_j[l]_{\min}) / (x_j[l]_{\max} - x_j[l]_{\min})$$

Step 2. **Swarm initialization**. The initial swarm consists of m' particles, each of which is generated by ramdomly selecting k exclusive subsets of d points(d-subsets[7]). Each particle is a vector in $k \times d \times d$ dimensions.

$$X_i^{(t)} = (z^{(t)}_{i1}, z^{(t)}_{i2}, ..., z^{(t)}_{ik}), t \in [1, Iteration_{\max}], i \in [1, m']$$

$$z^{(t)}_{im} = (x_1, x_1, ..., x_d), m \in [1, k]$$

Step 3. **Global PSO Iteration**. For each particle in each global iteration, *Location_Iteration* function is firstly applied to get grouping results and ending position $XE_i^{(t)}$ of the specific particle. We choose ROSS as the fitness value of current iteration based on the previous grouping result and update the new starting position of the specific particle based on Eq.(3-5). Global iterations continue until the best fitness value has stayed unchanged for $Stage_{max}$ or total number of iteration has exceeded $Iteration_{max}$. If the position of particle exceed the range of $[X_{min}, X_{max}]$, then its position will be reassigend to X_{min} or X_{max}. Similarly, if the velocity of a particle exceeds the range of $[V_{min}, V_{max}]$, then its velocity will be reassigned to V_{min} or V_{max}. X_{min}, X_{max}, V_{min}, V_{max} are derived from data scope, while $Stage_{max}$ and $Iteration_{max}$ are set to limit computational time.

Step 4. **Ouput grouping result:** Output the grouping result with the best fitness value, along with the hyperplane coefficients.

4. Access Models and Anomaly Detection

Anomalies in access flows can be divided into two parts: deviation from historical behavior and violation of access time sequence.The single-hop access profiling model offers a method to model direct access towards server applications and detect anomalies through outlier-based and tag-based approaches. Intruders may act like normal clients without any deviation in flow level attributes. For example, in a common portal environment, the web sever normally access database server after client request for personal information or upload some specific data, which can be expressed in time-sequenced pattern : (clients→web server→database). Either web server accessing database server with no previous client-request for its service, or no follwing access to database server from web server when clients request for their personal data, should be considered as abnormal. Thus, double-hop access relation modelis a reasonable

complement for the single-hop access profiling model. Both models and their detection approaches will be detailed in the following sub-sections.

4.1 Single-Hop Access Profiling Model

We formulate six features from access flows, which are *BYTE_CNT_TO*, *PKT_CNT_TO*, *BYTE_CNT_FROM*, *PKT_CNT_FROM*, *FLOWSCOUNT*, *DURATION*. The first five features are directly derived from the corresponding attributes of *access_flow_feature_set*$_T$s with the same key in a specific interval $t_{interval}$. *DURATION*=end_time_{max}-$start\_time_{min}$, is derived from the time difference between $start\_time_{min}$ and end_time_{max}. We disregard client ports in analysising access flows as they are always random and useless, thus a client is only denoted by its ip address, while a server is denoted by its listening ip and listening port. We remove *ip_client* and $t_{interval}$ from key of *access_flow_feature_set*$_T$, as we profile access behavior towards a server application for all clients. Expression of merged access profiles is shown in Definition 4.

Definition 4. *merged_access_profile* = {**key**: (*proto*, *ip_server*, *port_server*), **value**: {*Features*: (*BYTE_CNT_TO*, *PKT_CNT_TO*, *BYTE_CNT_FROM*, *PKT_CNT_FROM*, *FLOWSCOUNT*, *DURATION*) , Tags :(*DIRECTION*, *TIME*)}}

DIRECTION is a 4-bit binary digit, each bit of which is denoted as a diffent direction of access flows, including external→intra(0001), intra→external (0010), external→external (0100), intra→intra(1000). An intra ip belongs to the intra-network or public ip addresses owned by the coorperation, while external ip means the others. *TIME* is a *t*-bit binary, every bit of which is denoted as a diffent time slot of 24 hours, for example, if we split 24 hours into 4 time slot(*t*=4), then 0001 means the the specific profile occurs in 0:00-6:00.

After feature formulation of traning flow data, *merged_access_profile*s are stored,as well as the *OED* map for opposite-end divergence of ip/port pairs, which will be used in further realtime identificaiton. Each unique key of *merged_access_profile*s is considered as a specific server application, while the value shows flow behavior of a certain client request in analysis interver *T*. Clients of different server application may appear in dissimilar flow patterns. Thus, we employ PSOLGA to get access flow behavior of distinct server applications, by analysing observations of *merged_access_profile* with distinct *key*(*proto, ip_server, port_server*).

After PSOLGA clustering, results for each *key*(*proto, ip_server, port_server*) are obtained,as is shown in **Definition 5.** *cluster*[*key*] consists of *k* clusters, each of which contains four elements and two tags to describe the cluster. $hyperplane_i$ shows the orthogonal hyperplanes for the *i*th linear group. $residual_i$ is the maximal absolute value of orthogonal residual of the *i*th cluster. $center_i$ is the average of observation associated with the *i*th cluster. $radius_i$ is the maximal Euclidean Distance between intra-cluster points p^i_c and $center_i$.

Definition 5. Clustering result: *cluster*[*key*] = {($hyperplane_i$, $residual_i$, $center_i$, $radius_i$, $Direction_i$, $Time_i$)| $i \in [1,k]$}. $hyperplane_i$ = {(w^i_j,e_i)| $j \in [1:d]$， d is the dimension of features， e_i is the orthogonal residual for the *i*th hyperplane and w^i_j is the jth coefficient). $residual_i = max(|w^i p^i_c + e_i|)$, p^i_c is the scalable observations assigned to this cluster, $c \in [1, \#$ of observations associated with the *i*th cluster of *cluster*[*key*]. $center_i$ = {$avg(p^i_c[j]) \mid j \in [1:d]$}. $radius_i = max(distance_{Euclidean}(p^i_c, center_i))$. $Direction_i$ and $Time_i$ are tags of the *i*th cluster of *key*, which describe the direction and time occurance of the specific cluster.

Both $Direction_i$ and $Time_i$ are derived from OR operations of corresponding tag of every single observation asscociated with the specific cluster. For example, $Direction_i$ is 1001 when intra server application is accessed by external and intra clients. Similarly, if we split 24 hours

into 4 time slot, then 0101 means the behavior in the specific cluster occurs in both 0:00-6:00 and 12:00-18:00.

4.2 Single-hop Access Anomaly Detection

Four steps are taken to analysing incoming flow batches, using cluster results from the single-hop access profiling model:

Step 1. **Flow merging**. Servers and clients are identified by using *OED* map previously stored. Incoming flow batches will be then merged into form of $merged\_access\_profile_T$, along with the *Direction* tag and *Time* tag.

Step 2. **Normalization**. Basing on the maximal and minimal value in each dimention of training data, incoming *merged_access_profile*s are projected to the training feature space. y_j is the *j*th observation in incoming *merged_access_profile*s, $x^{training}[l]_{\min}$ is the minimal value of training data in the *l*st dimension, while $x^{training}[l]_{\max}$ is the maximal value.

$$y'_j[l] = (y_j[l] - x^{training}[l]_{\min}) / (x^{training}[l]_{\max} - x^{training}[l]_{\min})$$

Step 3. **Outiler-based anomaly detection**. Two different distance are used to determine whether the incoming *merged_access_profile* is anomalous. The incoming profile y_j is assigned to the closest *cluster*[key of y_j]$_m$, of which $hyperplane_m$ and y_j has the minimal orthogonal distance. The specific incoming profile y_j is consider as anomalous, if either the orthogonal residual of y_j to $hyperplane_i$ is larger than the associated $residual_i$, or the Euclidean Distance to $center_i$ is larger than the associated $radius_i$.

Step 4. **Tag-based anomaly detection**. We can derive the *DIRECTION* and *TIME* tag of the incoming profile from ownership of the associated ip addresses and occurrence time of access flows. The tags are then applied with OR operation with $Direction_i$ and $Time_i$, while $cluster_i$ is the cluster that the profile is assigned to.If the result differs from the corresponding tag of $cluster_i$, the incoming profile is also regarded as abnormal. For example, the $Direction_k$ tag of $cluster_k$ is 1000,which means historical clients accessing the specific server application in this pattern are external users. If the incoming profile assigned to $cluster_k$ with the Direction tag of 0001, it will be considered as an anomaly , which may be caused by internal fake-ip attacks or modifications of firewall rules.

4.3 Double-Hop Access Relation Model

In this section, we propose in-memory graph model to extract time-sequenced correlation from access flows, without repeating scanning training dataset. In-memory graph model *M* and double-hop access correlation rules *DHA_RULE* are formulized in **Definition 6-8**. *F* is the merge result of $access\_flow\_feature\_set_T$ with the same protocol, server ip address, server port, client ip address and occurrence time slot. $t^i_{interval}$ denotes the *i*th time slot $[i \times T : (i+1) \times T]$ *F* occurs in. $start\_time^i_{min}$ and $start\_time^i_{max}$ are the minimal and maximal start time of access flows in $t^i_{interval}$. *F* is for further use in rules generation. *Server*=(*proto*, *ip_server*, *port_server*), $Server_{pre}$ is the connected server application in pre order, denoted by its protocol, listening ip address and listening port. $Server_{post}$ is the connected server application in post order. Precedent access flow f_{pre} is the access flow towards $Server_{pre}$ from any client ip addresses in a certain interval $t_{interval}$. Posterior access flow f_{post} is the access flow towards $Server_{post}$ from ip_server_{pre} in the same $t_{interval}$.

Definition 6. *F*= {**key**: (*proto*, *ip_server*, *port_server*, *ip_client*), **value**: ($t^i_{interval}$, $start\_time_{min}$, $start\_time_{max}$)}, f_{pre}, $f_{post} \in F$

Definition 7. In-memory graph model $M=\{V_{ip}, V_{server}, E_{timespan}\}$

• $V_{ip}=\{V_{cip}, V_{sip} | v_{cip}=(ip, type), v_{sip}=(ip, type)\}$ are the verticles representing for either a client ip address or a server ip address. *type*=(*Client*| *Sever*| *Client and Server*) denotes the role of a corresponding ip address in access flows.

• $V_{server}=\{v_{server} | v_{server}=(proto, ip\_server, port\_server, tr\_table)\}$ represent for server applications, $tr\_table=\{(t_{interval}, start\_time_{min}, start\_time_{max})\}$ collects all access time records towards the specific server application.

• $E_{timespan}=\{E_s, E_c | e_s = (v_{server} \rightarrow v_{sip}), e_c=(v_{cip} \rightarrow v_{server}, tr\_table)\}$ are edges connecting V_{ip} and V_{server}. E_s connect V_{server} and V_{sip} with the same *ip_server* and no value is attached. E_c connect V_{cip} and V_{server}, representing a unique access pair between server application(*proto*, *ip_server*, *port_server*) and a specific client ip address(*ip_client*), of which *tr_table* collects all time records v_{cip} accessing v_{server}.

Definition 8. $DHA\_RULE= \{(Server_{pre} \rightarrow Server_{post}, Prob_{pre}, Prob_{post}, Cnt_{pre}, Cnt_{post}, Cnt_{co}) \}$ is the form of double-hop access rules.

- $Server_{pre}$ is the server application in f_{pre}. $Server_{post}$ is the server application in f_{post}.
- Cnt_{pre} is the distinct count of $t_{interval}$, during each of which f_{pre} occurs.
- Cnt_{post} is the distinct count of $t_{interval}$, during each of which f_{post} occurs.
- Cnt_{co} is the distinct count of $t_{interval}$, during each of which $Server_{post}$ is accessed by ip_server_{pre} after $Server_{pre}$ being accessed by any client.
- $Prob_{pre}= Cnt_{co} / Cnt_{pre}$, is the probability that f_{post} occurs after the first f_{pre} in the same analysis interval $t_{interval}$.
- $Prob_{post}= Cnt_{co} / Cnt_{post}$, is the probability that f_{post} occurs and at least one f_{pre} occurs before f_{post} in the same analysis interval $t_{interval}$.
- $Prob_{pre}$, $Prob_{post} > TH_{prob}$, Cnt_{pre}, $Cnt_{post} > TH_{cnt}$. (6)
- TH_{prob} and TH_{cnt} are filtering threshold, used to filter out rules of strong confidence.

Three major steps to generate *DHA_RULE*s are described as follows:

Step 1. **Merging**. Merge $access_flow_feature_set_T$s within all unique analysis interval $t_{interval}$ into *F*. *F*s are then used to initialize *M*.

Step 2. **Initialization of model *M* and graph computing**. Insert all unique ip addresses in keys of *F*s into V_{ip}, and update *type* of V_{ip} basing on whether it is a client ip, or a server ip, or both. Insert all unique pair(*proto*, *ip_server*, *port_server*) into V_{server}, and insert or update *tr_table* of the specific v_{server}. Similarly, E_s and E_c are inserted into *M*. After Initialization, connections and time records are utilized to extract rules. In-edges of a v_{sip} are connected to v_{server}s represented as its server applications, while the out-edges connecting to v_{server}s which have been requested for service from v_{sip} during the whole traning period.

Step 3. **Rule Extraction**. Inner join operation is done to the associated time records of v^{pre}_{server} and e^{post}_{c} to calculate the co-occurrence times.Only rules satisfying Eq.(6) are outputed.

In **0**, the workflow for rule generation is shown in detail. **Table 1** explains the *DHA_RULE* Generation Algorithm in further detail. This algorithm benefits from the compact struture of graph models and hash methods, with computational complexity of which is reduced to $O(N+I\times s\times n\times m)$. *N* is record count for training access flows. *s* is the number of v_{sip} with

type=(*Client and Server*). *n* is the maximum server ports associated with a server ip address. *m* is the maximum count of out-connected server ports by a specific client ip address. *I* is distinct count of $t_{interval}$ in inputing access flows. As *s*, *n* and *m* are far less than *N*, the computational complexity can be approximate to $O(N+I)$ in pratice.

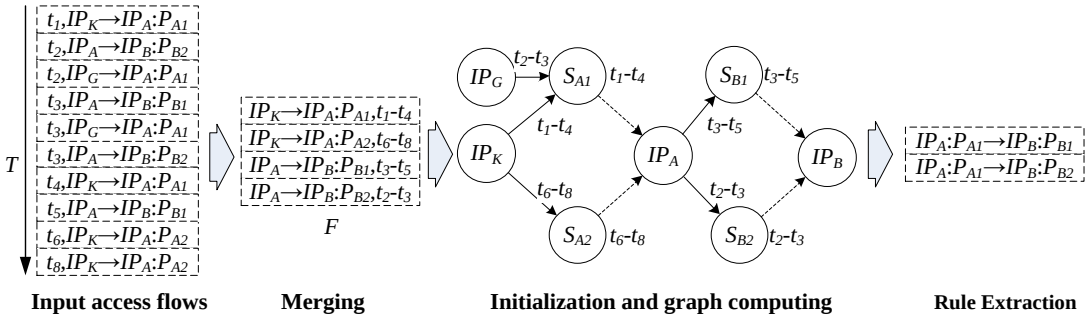


Fig. 5. workflow for rule generation

Table 1. *DHA_RULE* Generation Algorithm

Input: dataset of access flows $access_flow_feature_set_T$s, TH_{cnt}, TH_{prob}
Output: *R* as a set of *DHA_RULE*s

1. initialize R as an empty set, *M* as an empty graph
2. merge $access_flow_feature_set_T$s into *F*s
3. **for each** *F* **in** *F*s **do**
4. insert and update *M* with V_{ip}, V_{server} and $E_{timespan}$
5. **end for**
6. **for each** v_{sip} with *type* of (*Client and Server*) **in** *M* **do**
7. v^{pre}_{server} represents for the connected v_{server} by in-edges
8. v^{post}_{server} represents for the connected v_{server} by out-edges
9. e^{post}_{c} represents for out-edges connected to v^{post}_{server}
10. **for each** v^{pre}_{server} and e^{post}_{c} **do**
11. inner join $v^{pre}_{server}.tr\_table$ and $v^{post}_{server}.tr\_table$ by key($t_{interval}$) into record set tr_{co}={($t_{interval}$, $start\_time^{pre}_{min}$, $start\_time^{pre}_{max}$, $start\_time^{post}_{min}$, $start\_time^{post}_{max}$)}
12. $pair_{co}$={($proto^{pre}$, ip_server^{pre}, $port\_server^{pre}$) → ($proto^{post}$, ip_server^{post}, $port\_server^{post}$)}
13. $pair_{pre}$=($proto^{pre}$, ip_server^{pre}, $port\_server^{pre}$)
14. $pair_{post}$=($proto^{post}$, ip_server^{post}, $port\_server^{post}$)
15. $cnt_{pre}[pair_{pre}]$ = # of records in $v^{pre}_{server}.tr\_table$
16. $cnt_{post}[pair_{post}]$ = # of records in $e^{post}_{c}.tr\_table$
17. $cnt_{co}[pair_{co}]$=0
18. **for each** *tr* **in** tr_{co} **do**
19. **if** $start\_time^{post}_{max} > start\_time^{pre}_{min}$ **then** increment $cnt_{co}[pair_{server}]$ by 1
20. **end for**
21. $prob_{pre}[pair_{co}]= cnt_{co}[pair_{co}]/cnt_{pre}[pair_{pre}]$
22. $prob_{post}[pair_{co}]= cnt_{co}[pair_{co}]/cnt_{post}[pair_{post}]$

23. **if** $prob_{pre}[pair_{co}]$, $prob_{pre}[pair_{co}] > TH_{prob}$ and $cnt_{pre}[pair_{pre}]$, $cnt_{post}[pair_{post}] > TH_{cnt}$ **then** insert ($pair_{co}$, $prob_{pre}[pair_{co}]$, $prob_{post}[pair_{co}]$, $cnt_{co}[pair_{co}]$, $cnt_{pre}[pair_{pre}]$, $cnt_{post}[pair_{post}]$) into R
24. **end for**
25. **end for**
26. output R

4.4 Double-hop access anomaly detection

The extracted *DHA_RULE*s are used in anomaly detection. Each rule has two evaluation streams: EV_R^{pre} and EV_R^{post}. EV_R^{pre} is considered as a binomial trial of size SL in descrete time, with the given probability $Prob_{pre}$, while $Prob_{post}$ is for EV_R^{post}, as shown in Definition 9. SL is the length of both evaluation streams.

Definition 9. $EV_R^{pre} \sim b(Prob_{pre}, SL)$, $EV_R^{post} \sim b(Prob_{post}, SL)$

When realtime access flows matched with a specific *DHA_RULE* show up, a new evaluation value will be append to the corresponding evaluation stream and only the latest SL values will be kept in the stream. Evaluation values for different situations are listed in **Table 2**. Evaluation values of two evaluation streams are set to 1 only when f_{pre} and f_{post} both occur in the same time slot $t_{invertal}$ and f_{post} occurs after at least one f_{pre}. If no f_{post} occurs after f_{pre} , only ev_{pre} is set to 0 and no value to be inserted into EV_R^{post}. Similarly, only ev_{post} is set to 0, when f_{post} occurs without any f_{pre}. If no f_{post} or f_{pre} shows up, no value will be inserted into evaluation streams.

Table 2. Evaluation values

Situation	Evaluation value
$f_{pre} \wedge \neg f_{post}$	$ev_{pre}=0$, $ev_{post}=\emptyset$
$f_{pre} \wedge f_{post}$	$ev_{pre}=1$, $ev_{post}=1$
$\neg f_{pre} \wedge f_{post}$	$ev_{pre}=\emptyset$,$ev_{post}=0$,

Anomaly value is defined in **Definition 10**. Cumulative probability distribution is used to distinguish where an evaluation stream should be considered as an anomaly. *AValue* is the probability anomaly happens. Only when *AValue* is higher than α is considered as a valid anomaly. R is a specific *DHA_RULE* under evaluation. n is the count of positive evaluation in an evaluation stream. p is $Prob_{pre}$ for EV_R^{pre} and $Prob_{post}$ for EV_R^{post}.α is the anomaly detection threhold.

Definition 10. $AValue=1-\sum_{i}^{n} b(i \mid p, SL)$, $AGap=AValue-\alpha$, $isAnomalous(R) = AGap \geq 0$.

5. Experiment Evaluation

5.1 Evaluation of PSOLGA algorithm

We use a sample of synthetic data of 4 groups of linear distributed 2-dimensional points and the hockey data set(nhl194) for evaluating the improvement of PSOLGA compared to LGA. nhl194 contains information on the performance of players in the Canadian National Hockey

League for the 94–95 competition[7]. Four features(PTS,P/M,PIM,PP) of nhl194 is under consideration and the best group number of nhl194 is 3 as mentioned in [7]. Hence ,we try to divide the synthetic data into 4 groups and nhl194 into 3 groups, by applying both LGA and PSOLGA. Both grouping results are listed and discussed as below.

0 shows contrasive results of using LGA and PSOLGA in the two mensioned datasets. The left ones are plot/scatter plot of gouping results and the right ones show the minimum ROSS of both grouping algorithms in 20 resampling rounds. With the minimal starting value *m* suggested in [7], we notice that LGA cannot always find the best ROSS while PSOLGA shows good stablility towards the fittest result. In totoal 20 resampling tests, LGA achieves the best ROSS for 16 times in synthetic data and 9 times in nhl194, while PSOLGA achieves 100% success in both testing datasets.

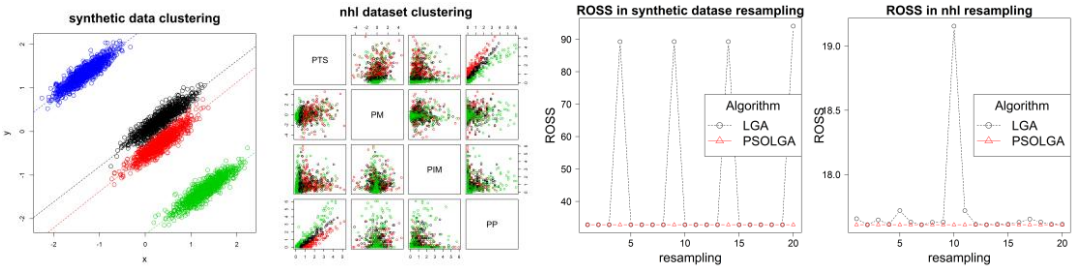


Fig. 6. grouping results of synthetic data(*#starting value*=77) and nhl194(*#starting value* =44)

The computional efficiency of both algorithms is listed in **Table 3**. *#starting hyperplane* is the starting value of LGA and the size of swarm in PSOLGA. *resample* is the times for repeating the corresponding algorithm. *calc* is the times algorithm scans the dataset. $ROSS_{min}$ is the best found fittness value. With the same starting value, PSOLGA requires more caculation than LGA, as PSOLGA has to iterate grouping for multiple rounds until the best fitness value stay unchanged for certain rounds or exceed the upper limit of totoal optimizaition rounds. There is a tradeoff between calulational complexity and clustering stability. Benefited from the strategic random search mechanism brought in by PSO, PSOLGA is able to reduce computational load by choosing a smaller starting value while still succeed to find the best clustering result.

Table 3. comparation of computational complexity

	#starting hyperplane	*resamples*	*calc*	$ROSS_{min}$
LGA (nhl194)	44	3	6154	17.6133
	44	10	20588	17.60981
	44	100	205464	17.60981
PSOLGA (nhl194)	10	2	9506(IT:13)	17.60981
	20	1	5373(IT:7)	17.60981
	44	1	13181(IT:8)	17.60981

As shown in **Table 3**, PSOLGA can still find the minimum ROSS value after 7 iterations when the size of swarm is reduced by half to 20. Compared to result of PSOLGA with 44 particles, more than 50% computational efforts are reduced. However, PSOLGA cannot gurantee for finding the best result when the swarm size is too small, which is 10 for dataset

nhl194. Experience shows swarm of half the size of the starting groups suggested by LGA is sufficient to find the best grouping result.

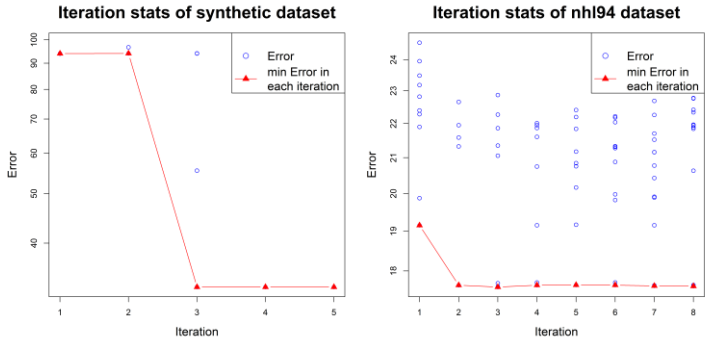


Fig. 7. iteration status of PSOLGA

0 shows the status of PSOLGA within each interation for both datasets. Blue dots denote error(objective value) of particles in a certain iteration, and the red triangles are the best/minimum error in each iteration. We can see process of particles flying toward the best location, until the best grouping result is obtained.

5.2 Evaluation of single-hop access anomaly detection

As is shown in **0**, we collect normal access flows from normal user *A*, *B* and *C*, as well as attacking access flows from attackers *D* and *E*. Web server *F* is their target, which offers 3 kinds of services: log-in, insert and query personal data. Attackers use the tool sqlmap for sql injection attempts, such as guessing authentication information, acquiring version of database and structure of tables. Raw flows are first converted to access flows.The analysis interval *T* is set to be 10 second.

To get a glimpse of feature space of the training dataset, we take 200 random samples from the training dataset, in which the first 100 observations are from nomarl access flows and the other 100 are abnormal access flows originated from attackers. The similarity matrix in **0** is a heat map of similarity between two observations in euclidean distance. The darker the color is, the more similar the two observations are. We can notice that abnormal flows are scattered over a wide area, as attackers launched multiple different attacks which show dissimilarities in flow features. For example, attackers can acquire version of database by a single connection towards the webserver, but multiple flows need to be initialized to get the column names. Besides that, some abnormal access flows are similar to benign ones in euclidean distance.

We split the training dataset in two equal subset to do cross exmination. We compare PSOLGA and Kmeans in detecting abnormal access flows. As we assume no knowledge about the actrual groups of normal access flows, we test grouping from 1 to 7 clusters, and choose the cluster number with the best detection rate. Accuracy rate is count of true classified samples divided by overall sample count. As is shown in **0**, PSOLGA achieves the best detection rate 98.45% when #clusters=3, while the best result of the Kmeans approach shows at #clusters=6. True negative rate and true positive rate are shown in **0** and **0**. In **0**, Gap analysis[7] is used for estimating the number of linear groups, which also suggest 3 clusters. It means Kmeans tend to overestimate group numbers and PSOLGA is able to achieve best accuracy rate when clustering data with the correct group number. From **0** and **0**, we can see

that Kmeans is insufficient to group data distributed in linear strutures, while PSOLGA is able fit data into 3 groups in different colors.

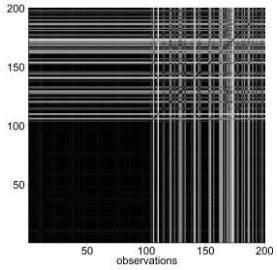


Fig. 8. Similarity matrix

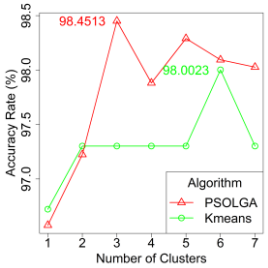


Fig. 9. Accuracy rate(%)

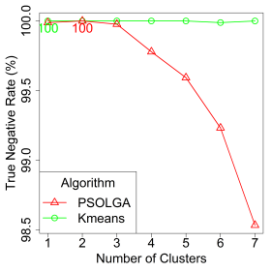


Fig. 10. True negative rate(%)

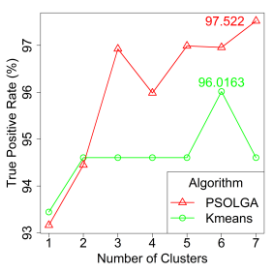


Fig. 11. True positive rate(%)

Our approach use both orthogonal distance and euclidean distance to detect outliers, thus it is able to distinguish abnormal flows which are similar to normal ones in euclidean distance (as shown in **0**) but deviate from the corresponding linear grouping struture. Howerver, the Kmeans approach uses merely euclidean distance, so it is incapable to detect these abnormal flows. When PSOLGA try to group data into more clusters than the correct cluster number(3,suggested by GAP analysis), it leads to overfitting. Overfitting will bring down the overall accuracy rate and some unpreditable small fluctuation of curves. For example, the fluctuation in accuracy rate and true positive rate occurs when cluster number is increased from 3 to 4 and from 4 to 5, as shown in **0** and **0**, due to dataset distribution and wrong clustering groups, but the downtrend of accuracy rate would not be affected.

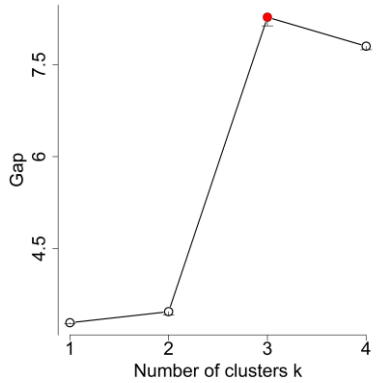


Fig. 12. Gap Analysis

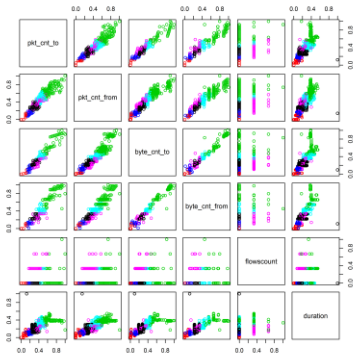


Fig. 13. Groups of normal access flows (Kmeans, #cluster = 6)

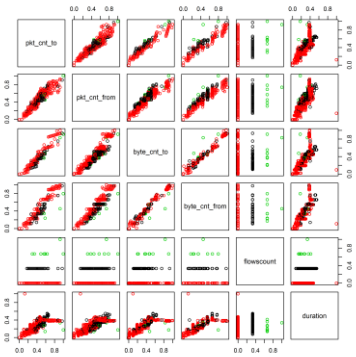


Fig. 14. Groups of normal access flows (PSOLGA, #cluster = 3)

5.3 Evaluation of double-hop access anomaly detection

We generate 58 rules from the training dataset without knowledge about predifined server applications. Five typical rules are listed in **Table 4**. Rules *webmysql* and *webredis* are rules for web servers and corresponding databases. *hdfsctrl*,*hdfsctrl* and *hdfsdbctrl* are rules for HDFS nodes. 50010 is the listening port of HDFS nodes for control messages. 9000 is the

webservice port for the master node of HDFS. We conduct two different attacks: illegal database dump and directory traversal/path traversal, to show the use of the two evaluation streams: EV_R^{pre} and EV_R^{post}. The first attack results in anomaly "Illegal Database Access", and the second one leads to anomaly "Abnormal Web Access". We set the size of evaluation stream SL=10 and anomaly detection threhold α=0.99.

Situation "Illegal Database Access": The attacker got root control of the web server F and the authenticaiton information for the mysql database from the previous sql injection. After logging onto F, the attacker try to dump data from the database from 13:55:00 to 14:01:40. As is shown in rule *webmysql*, $prob_{post}$ is 0.73, meaning within a certain interval(10s in this article), f_{post} towards mysql database follows the f_{pre} towards web server F for the probability of 73%. We can see from the second trend chart of **0** that the $AValue_{post}$ starts to arise right after database dump started.When $AValue_{post}$ reaches the anomaly detection threhold α(0.99) and $AGap_{post}$ gets larger than 0, a valid anomaly will be reported.

Situation "Abnormal Web Access": The attacker succeed to locate a vulnerbility of directory traversal/path traversal on the web server F, and launch attacks to access files on F and execute system commands. During the attack(14:36:00-14:41:40), F do not need to access the mysql database and thus no access flows between F and mysql database appear, which is abnormal for normal users. Negative value starts to be append to EV_R^{pre}, and $AValue_{pre}$ gets higher afterwards. After $AValue_{pre}$ gets higher than the anomaly detection threhold α(0.99), a valid anomaly is triggered.

Table 4. Double-hop access rules

RuleName	$proto_{pre}$	ip_{pre}	$port_{pre}$	$proto_{post}$	ip_{post}	$port_{post}$	cnt_{co}	cnt_{pre}	cnt_{post}	$prob_{pre}$	$prob_{post}$
webmysql	TCP	192.168.30.126	80	TCP	192.168.31.179	3306	401	676	542	0.99	0.73
webredis	TCP	192.168.30.130	80	TCP	192.168.31.129	6379	708	714	713	0.99	0.99
hdfsctrl	TCP	192.168.31.2	50010	TCP	192.168.31.3	50010	587	587	591	1	0.99
hdfsctrl	TCP	192.168.31.3	50010	TCP	192.168.31.2	50010	586	591	587	0.99	1
hdfsdbctrl	TCP	192.168.31.2	9000	TCP	192.168.31.3	50010	586	587	591	0.99	0.99

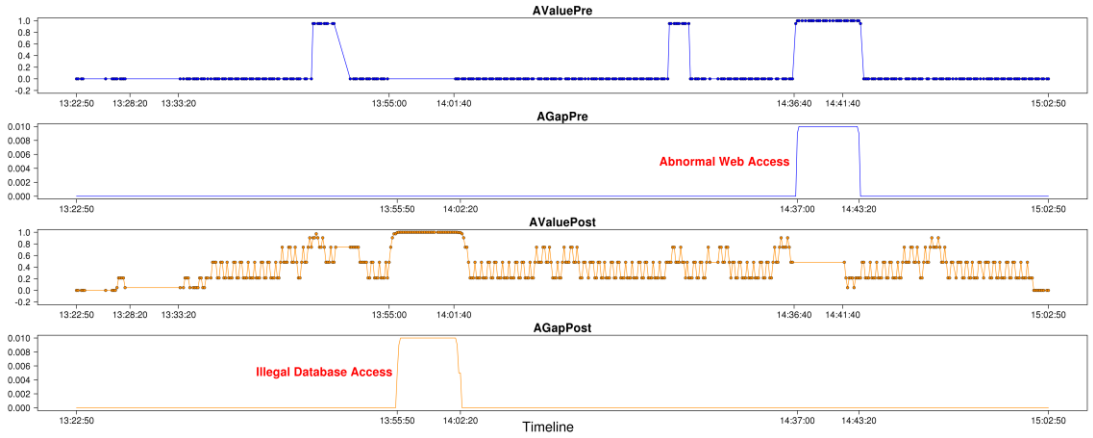


Fig. 15. Evaluation streams

6. Conclusion

Advanced persistent threats and insider threats remain a serious concern to organisations. Lack of appropriate methods to keep track of overall network activities makes it difficult for security team to uncover unknown exploits and malicous insiders. Thus, it is neccesary to arm network administrators with autonomous inventory of netwok assets and behavior analysis technique.

In this paper, we investigate autonomous flow-based anomaly detection in enterprise network. Compared with existing anomaly detection methods, this work has the following differences: First of all, we propose a methodology of discovering server applications in the targeted network without prior knowledge and merge flows into access flows towards server applications. Besides that, we introduce a novel linear grouping algorithm PSOLGA for mining the significant linear strutures in access flows, which are then used to build behavior profiles for each indivual server application. PSOLGA achieves better grouping stability and computational efficiency than traditional LGA. In addition, we use in-memeroy graph model to search for highly dependent access flows in time series and reduce the overall computational workload. These dependent flow sequences are formulated into rules for the detection of violation in access relations. Finally, we conduct experiments with both simulation data and real-world flow dataset. Performance and accuracy of our model are verified to be promising.

Acknowledgements

The authors would like to thank the anonymous reviewers for their detailed reviews and constructive comments, which help improve the quality of this paper. This work was supported in part by National Natural Science Foundation of China under Grant No.61101108 and No. 61121061.

References

[1] Binde, Beth, Russ McRee, and Terrence J. O'Connor, "Assessing outbound traffic to uncover advanced persistent threat," *SANS Institute*, 2011. Article (CrossRef Link)

[2] Alperovitch, Dmitri, "Revealed: operation shady RAT," *McAfee*, vol. 3, 2011. Article (CrossRef Link)

[3] Claise, Benoit, Brian Trammell, and Paul Aitken, "Specification of the IP Flow Information Export (IPFIX) protocol for the exchange of flow information," *draft-ietf-ipfix-protocol-rfc5101bis-08 (work in progress)*, 2013. Article (CrossRef Link)

[4] Sheikhan, Mansour, and Zahra Jadidi, "Flow-based anomaly detection in high-speed links using modified GSA-optimized neural network," *Neural Computing and Applications*, vol. 24, no. 3-4, pp. 599-611, 2014. Article (CrossRef Link)

[5] Animesh Patchaand Jung-Min Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*,vol. 51,no. 12, pp.3448-3470,2007. Article (CrossRef Link)

[6] Riccardo Poli , James Kennedy and Tim Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol 1, no 1, pp.33-57, June. 2007. Article (CrossRef Link)

[7] Stefan Van Aelst, Xiaogang Steven Wang, Ruben H. Zamarand Rong Zhu, "Linear grouping using orthogonal regression," *Computational Statistics & Data Analysis*, vol. 50, no. 5, pp.1287-1312, 2006.Article (CrossRef Link)

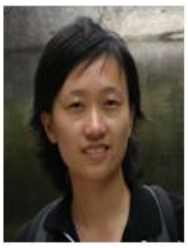
[8] Gilberto Fernandes, Joel J. P. C. Rodriguesand Mario Lemes Proença, "Autonomous profile-based anomaly detection system using principal component analysis and flow analysis," *Applied Soft Computing*,vol. 34, pp.513-525,2015.Article (CrossRef Link)
[9] Gilberto Fernandes Jr., Luiz F. Carvalho, Joel J. P. C. Rodriguesand Mario Lemes Proença Jr., "Network anomaly detection using IP flows with Principal Component Analysis and Ant Colony Optimization," *Journal of Network and Computer Applications*, vol. 64, pp.1-11, 2016. Article (CrossRef Link)
[10] Zahra Jadidi, Vallipuram Muthukkumarasamy, Elankayer Sithirasenanand Kalvinder Singh, "Performance of Flow-based Anomaly Detection in Sampled Traffic," *Journal of Networks*, vol. 10, No. 9, 2016. Article (CrossRef Link)
[11] M. Shojafar, N. Cordeschi; E. Baccarelli, "Energy-efficient Adaptive Resource Management for Real-time Vehicular Cloud Services," in *Proc. of IEEE Transactions on Cloud Computing* , vol.PP, no.99, pp.1-1. 2016. Article (CrossRef Link)
[12] Quan Guo, Jia Jiaand, Guangyao Shen et al., "Learning robust uniform features for cross-media social data by using cross autoencoders," *Knowledge-Based Systems*,vol. 102, pp.64-75, June 15, 2016. Article (CrossRef Link)
[13] S. Beheshti, F. Alajaji and T. Linder, "Optimal Joint Decoding of Correlated Data Over Orthogonal Multiple Access Channels with Memory," *IEEE Transactions on Vehicular Technology* , vol.PP, no.99, pp.1-1. Mar.17, 2016.Article (CrossRef Link)
[14] Bingdong Li, Jeff Springer, George Bebis and Mehmet Hadi Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567-581, March , 2013. Article (CrossRef Link)
[15] Minho Jo, Longzhe Han, Dohoon Kimand Hoh Peter In, "Selfish attacks and detection in cognitive radio Ad-Hoc networks," *IEEE Network*, vol. 27, no. 3, pp.46-50, 2013.Article (CrossRef Link)
[16] E. Sharafuddin, N. Jiang, Y. Jin and Z. L. Zhang, "Know Your Enemy, Know Yourself: Block-Level Network Behavior Profiling and Tracking," in *Global Telecommunications Conference (GLOBECOM 2010)* , pp. 1-6, Dec, 2010. Article (CrossRef Link)
[17] Zhang Xiaochen, Liu Shengli, "Meng Leiand Shi Yunfang. Trojan Detection Based on Network Flow Clustering," *Multimedia Information Networking and Security (MINES)*, pp. 947-950, 2012. Article (CrossRef Link).
[18] Pelleg, Dan, and Andrew W. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters." *ICML*, vol. 1, 2000. Article (CrossRef Link)
[19] Shing-Han Li, Yu-Cheng Kao, Zong-Cyuan Zhang, Ying-Ping Chuang and David C. Yen, "A network behavior-based botnet detection mechanism using PSO and K-means," *ACM Transactions on Management Information Systems*, vol. 6, no. 1, 2015.Article (CrossRef Link)
[20] L. A. García-Escudero, A. Gordaliza, R. San Martín, S. Van Aelst and R. Zamar," Robust linear clustering," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, no. 1, pp.301-318, 2009.Article (CrossRef Link)
[21] L. A. García-Escudero, A. Gordaliza, A. Mayo-Iscarand R. San Martín, "Robust clusterwise linear regression through trimming," *Computational Statistics & Data Analysis*, vol. 54, no. 12, pp.3057-3069, 2010.Article (CrossRef Link)
[22] Ivo Friedberg, Florian Skopik, Giuseppe Settanni and Roman Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Computers & Security*, vol. 48, pp.35-57, 2015.Article (CrossRef Link)
[23] Barsamian and Alexander V, "Network characterization for botnet detection using statistical-behavioral methods. *Doctoral dissertation, Dartmouth College*, 2009. Article (CrossRef Link)
[24] Haag Peter, "Watch your Flows with NfSen and NFDUMP," in *50th RIPE Meeting*. 2005. Article (CrossRef Link)
[25] Peng Bichen, Guo Wei, Liu Daipingand Fu Jianming, "Dynamic application flow cluster based on traffic behavior distance," in *Proc. of IEEE Computer Society*, vol. 1, pp. 1291-1296 2010. Article (CrossRef Link)

[26] V. Frias-Martinez, J. Sherrick, S. J. Stolfo and A. D. Keromytis, "A Network Access Control Mechanism Based on Behavior Profiles," in *Proc. of Computer Security Applications Conference*, pp. 3-12, 2009. Article (CrossRef Link)
[27] Gintautas Garsvaand Paulius Danenas, "Particle swarm optimization for linear support vector machines based classifier selection," *Nonlinear Analysis-Modelling and Control*, vol. 19, no. 1, pp.26-42, 2014. Article (CrossRef Link)
[28] Ying Tan, Yuhui Shiand, Fernando Buarque et al., "A Population-Based Clustering Technique Using Particle Swarm Optimization and K-Means," *Advances in Swarm and Computational Intelligence*, pp. 145-152, 2015. Article (CrossRef Link)
[29] Z. p. Yan, C. Deng, J. j. Zhou and D. n. Chi, "A novel two-subpopulation particle swarm optimization," in *Proc. of 10th World Congress on Intelligent Control and Automation (WCICA)*, pp. 4113-4117, 2012. Article (CrossRef Link)

Weixin Liu is currently a Ph.D. candidate at the School of Computer, Beijing University of Posts and Telecommunications, Beijing, China. He received his B.E. degree in communication engineering from University of Science and Technology Beijing in 2006. His research interests include network security, intrusion detection, behavior modeling, vulnerability analysis and risk assessment. *The corresponding author. Email:jack18jack@gmail.com

Kangfeng Zheng is a vice-professor, Ph.D supervisor at the School of Computer, Beijing University of Posts and Telecommunications, China. He received his Ph.D. in 2006 from School of Information Engineering in Beijing University of Posts and Telecommunications. His research interests are network security, intrusion detection, malicious code analysis, honey-net system, cyber-attack modeling and Advanced Persistent Threat.

Bin Wu Ph.D. in signal and information processing, lecturer at the School of Computer, Beijing University of Posts and Telecommunications, China. He received his Ph. D. in 2008 from Beijing University of Posts and Telecommunications. His research is centered on active defense technology in various computer and communication networks, including intrusion detection, honey-net systems and secure gateway.



Chunhua Wu is a lecturer in Beijing University of Posts and Telecommunications. She received the B.S degree in communication engineering from Chengdu University of Information Technology in 1999 and the Ph.D degree in signal and information processing from Beijing University of Posts and Telecommunications in 2008.

Xinxin Niu received the BS and MS degree from the Beijing University of Posts and Telecommunications (BUPT) in 1985 and 1988, and the PhD degree from the Department of Electronic Engineering of the Chinese University of Hong Kong. She is a professor and doctoral supervisor in School of Computer Science of BUPT. Her research areas include Information and Network Security, Information Hiding and Digital Watermark, Digital Content and Security