AD^2: Anomaly Detection on Active Directory Log Data for Insider Threat Monitoring

Chih-Hung Hsieh
Institute of Informaiton Industry
Taipei, Taiwan
Email: chhsieh@iii.org.tw

Chia-Min Lai, Ching-Hao Mao, and Tien-Cheu Kao
Institute of Informaiton Industry
Taipei, Taiwan
Email: {senalai, chmao, tckao}@iii.org.tw

Kuo-Chen Lee*
Institute of Informaiton Industry
Taipei, Taiwan
Email: kclee@iii.org.tw
*: Corresponding Author

***Abstract*—What you see is not definitely believable is not a rare case in the cyber security monitoring. However, due to various tricks of camouflages, such as packing or virutal private network (VPN), detecting "advanced persistent threat"(APT) by only signature based malware detection system becomes more and more intractable. On the other hand, by carefully modeling users' subsequent behaviors of daily routines, probability for one account to generate certain operations can be estimated and used in anomaly detection. To the best of our knowledge so far, a novel behavioral analytic framework, which is dedicated to analyze Active Directory domain service logs and to monitor potential inside threat, is now first proposed in this project. Experiments on real dataset not only show that the proposed idea indeed explores a new feasible direction for cyber security monitoring, but also gives a guideline on how to deploy this framework to various environments.**

***Keywords*—Active Directory Log Analysis, Anomaly Detection, Behavioral Modeling, Machine Learning, Advanced Persistent Threat.**

I. Introduction

The so called "advanced persistent threat"(APT) is a set of stealthy and continuous computer hacking processes. Typically, APT processes take a high degree of covertness over a long period of time, use malware of sophisticated techniques to exploit vulnerabilities in systems, and continuously monitor or extract confidential data from specific targets, once attackers get the control of victim systems. According to cyber security technical reports from various organizations or companies, it takes averagely at least 346 days for more than 81% victims to aware that they have been hacked [1]. Yet, due to various tricks of camouflages, such as packing, obfuscated shellcode, or virtual private network (VPN), signature-based malware detection technique, such as intrusion detection system (IDS), is getting less useful, especially on identifying advanced persistent threat(APT) at post-compromised stage. Facing the above challenges, considering account's subsequent behaviors may gives cyber security engineers additional context-based evidences and smart chance to detect threats from insiders [2].

In this report, instead of expert rules with only a few pre-defined signatures, a threat detection method regarding account's behavior sequences recorded as Active Directory (AD) logs was proposed. The AD domain controller of an organization monitors all related information when any intranet accounts try to allocate or acquire various resources and services. To the best of our knowledge so far, none of previous works are dedicated to threat identification by using sequential AD data modeling. The proposed framework 1) takes the AD logs as time-series input data providing chronological evidences, 2) emphasizes on sequential context mining from collected AD log, and 3) for each account, build the probability Markov model where best depicts the corresponding likelihoods of different behaviors occurring.

For real world feasibility concerning, a real dataset of AD logs, from real organization in Taiwan containing large amount of employees, was collected. After proper pre-processing and behavior learning, the performance of the proposed framework is measured with cross-validation manner for the sake of objectively evaluating the effectiveness and robustness. A fair N-fold cross validation experiment shows that the learnt behavioral Markov model gives successful results in terms of outstanding recall rates as well as fair precisions. The merits and contributions of this paper are fourfold. 1) the inside threat detection problem is first formularized as a sequential behavior modeling problem regarding with time-series AD log data mining. 2) The learnt model has good ability at monitoring the post-compromised anomaly behavior in terms of 66.6% recall and 99.0% precision rate. 3) the useful domain knowledge provided by well-known cyber security company, TrendMicro, was encoded as an annotation profile and indeed helps building accurate model; 4) a practical guideline for future users doing parameter selection is also proposed based on series discussions of how the framework parameters effect the detection results.

In the remaining parts of this report, section II briefly introduces the target Active Directory domain service. Section III describes the kernel methods of the proposed framework, including Markov model, the encoded representation for domain knowledge, and the designed anomaly detection mechanism. The performance of the proposed framework is evaluated in Section IV. At last, Section V concludes this project.

II. Backgrounds & Materials

A. Active Directory Domain Service

The Active Directory domain service is a directory service that Microsoft developed for Windows domain networks [3][4]. An AD domain controller authenticates and authorizes all users and computers in a Windows domain type networkassigning and enforcing security policies for all computers and installing or updating software. For example, when a user logs into a computer that is part of a Windows domain, Active Directory checks the submitted password and determines whether the

user is a system administrator or normal user [5]. The AD domain controller of an organization monitors all related information when any intra-net accounts try to allocate or acquire various resources and services. To the best of our knowledge so far, none of previous works are dedicated to threat identification by using sequential AD data modeling. Figure 1 is an illustration example of how a "Kerberos" process, one variant of AD domain, works and interacts with intra-net accounts. The process goes as the following steps, detail about this "Kerberos" process example can be referenced from [6]:

Stage 1 The Authentication Service Exchange
- **Step 1** Kerberos authentication service request (KRB_AS_REQ)
- **Step 2** Kerberos authentication service response (KRB_AS_REP)

Stage 2 The Ticket-Granting Service Exchange
- **Step 3** Kerberos ticket-granting service request (KRB_TGS_REQ)
- **Step 4** Kerberos ticket-granting service response (KRB_TGS_REP)

Stage 3 The Client/Server Exchange
- **Step 5** Kerberos application server request (KRB_AP_REQ)
- **Step 6** Kerberos application server response (optional) (KRB_AP_REP)

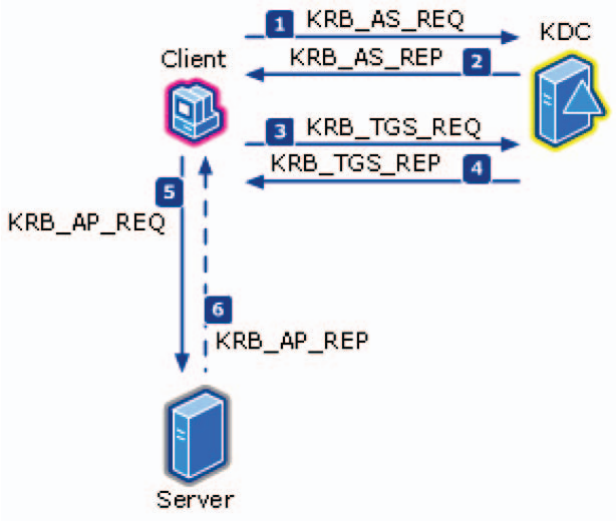

Fig. 1. An illustration of AD domain controller works and interact with intra-net accounts.

B. The Real Dataset

The performance of the proposed framework is evaluated in this section. In this experiment, a certain government organization of 95 employees in Taiwan was selected as the deployment environment of the proposed method. The Active Directory domain service using version of Windows Servers 2008 R2 is mounted on this organization's domain network and keeps monitoring all service requests and resource allocations raised from intra-net account. In this circumstance, Total 12,310,519 logs with size of 22.5 giga-bytes (GBs) was collected during two months, from 2014/11/26 to 2015/01/02. Among the 22.5 GBs data, logs with respect to accounts of real employees was left to be analyzed and forms an original dataset D in the following experiment. At last, D consists of 2,887,504 logs recorded in a 5.2 GBs file from 95 employees. It shoud be mentioned that because the number of event code "4624" and "4634" extremely dominate than those of others, and may lead to a biased Markov model. For the dataset D in the following experiment, those two event codes are safely ignored as considering model state.

III. Method

The ultimate goals of this project are as followings: 1) to collect Active Directory log data which is generated once any account try to access the Active Directory domain service; 2) to build behaivoral model capable of describing personal tendency for each user; and 3) to estimate the resulted likelihood given one's model to generate certain subsequent event codes. Based on the requirements mentioned above, the structure of the proposed framework can be made up of three major modules. The first is responsible for pre-processing raw data composed of whole access logs caused by all intra-net accounts and for forming the input dataset of following analytic usages. In the second module, The Markov model, the famous machine learning algorithm and well-known as a consequent state changing modeling tool, is then adopted to be the kernel approach to summarizing the user's behaviors. The last one is designated as an anomaly detection process to determine the likelihood that a certain account's model produce the given input sequences of event codes. Figure 2 shows the whole framework comprising of the three modules. The remain parts of this section give the brief introduction of adopted Markov model, the usage of prior knowledge, and the details about those three major modules.

A. Markov Model

In probability theory, a Markov model is a stochastic approach to model randomly changing systems where it is assumed that future states depend only on the present state and not on the sequence of events that preceded it (that is, it assumes the Markov property)[7]. Generally, the reason of taking this assumption is because it enables subsequent reasoning and computation regarding the model that would otherwise be intractable. The simplest Markov model is the Markov chain. It models the state of a system with a random variable that changes through time. In this context, the tendency of every transition form one state to another is described by a probability. The Markov property suggests that the distribution of this probability depends only on the distribution of the previous state. Due to the advantage of being good at describing consequent state changing, there are lots of applications, such as speech recognition [8], hand-written text recognition [9], gesture recognition [10], and cyber security intrusion detection[2], based on Markov model or its popular variant, hidden Markov model. Figure 3 is an illustration example of a Markov model with 3 states [11].

B. Generic Markov-model-state Annotation profile

Because the proposed method try to build the behavioral model for each account and to detect anomaly once account

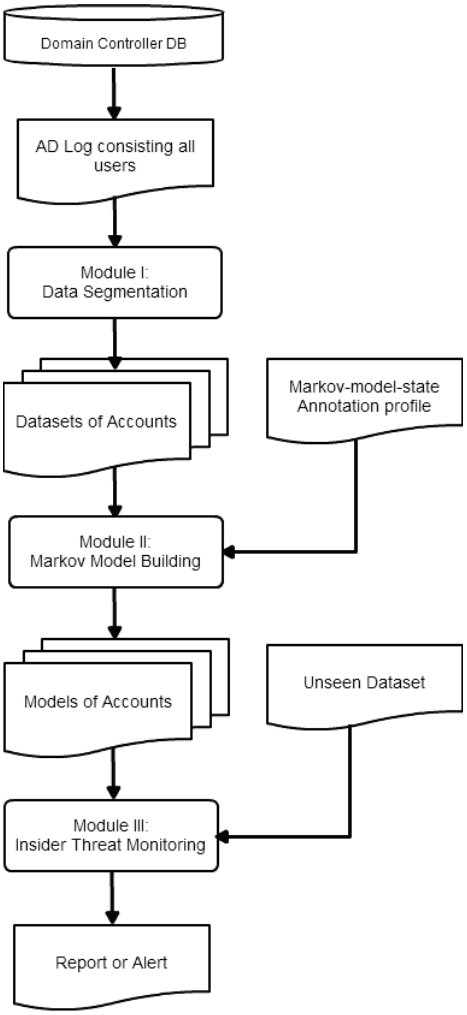

Fig. 2. The proposed framework of inside threats monitoring

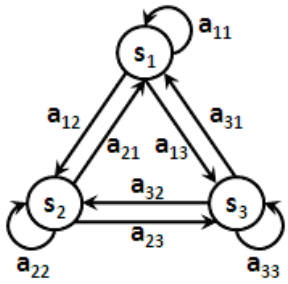

Fig. 3. An example of 3-state Markov model.

do not act like themselves compared to their historical daily routine. For this purpose, instead of considering only Active Directory code, making use of additional information as more as possible may significantly improve the model's representative degree. In this subsection, we define a Markov-model-state annotation (MMA) profile to encode proper prior knowledge by describing that which event code should be co-considered with certain specific attributes as a complete Markov state. For example, according to the Active Directory domain service specification provided by Microsoft, event codes "4771" represents that an account was pre-authentication failed for some reason. Therefore, it will provide much useful information about details of failure when event code "4771" be co-considered with the attribute "result code". Figure 4 is an illustration of a Markov-model-state annotation profile. It includes three annotations for three different event codes, respectively. For example, event code of no.4623 should be co-considered with both two fields, "xxx" and "yyy", to be formed as one state of Markov model. Note that for any event code which is unlisted in profile, it means that this event code is adopted default setting where implies using only event code itself as Markov state.

Annotation 1:

$$Event\_Code = 4623\ with$$
$$Field\_Name_1 = xxx,$$
$$Field\_Name_2 = yyy.$$

Annotation 2:

$$Event\_Code = 4624\ with$$
$$Field\_Name_1 = xxx,$$
$$Field\_Name_2 = yyy,$$
$$Field\_Name_3 = zzz.$$

Annotation 3:

$$Event\_Code = 4723\ with$$
$$Field\_Name_1 = aaa,$$
$$Field\_Name_2 = bbb,$$
$$Field\_Name_3 = ccc.$$

Fig. 4. An example of Markov-model-state annotation profile.

C. Module I: Generate Dataset of Event Code Chains.

Because the collected raw data is mixed with AD log data coming from multiple different accounts and the proposed method try to build separate behavior model for each account, one function of Module I is to partition original AD log data into different files of dataset, one for each account. Besides, the adopted Markov approach models one account's multiple instances of different operations as a probabilistic model. Different operation instances also need to be divided from a long consequent log sequence of event code into multiple shorter event code segments. The hypothesis used here to separate out those event code segments is that considering idle time in naive but real circumstance, the time intervals inter two independent segments of operations are usually longer than those intra one operation. For this reason, during the idle time, the time intervals between two subsequent AD logs are usually longer than those in working periods. In module I, a real-valued parameter θ is adopted as a cutting threshold. θ is longest allowed time interval and is used to divide a event sequence into two segments. If a time interval of two subsequent AD logs are longer than θ, it will lead two different event segments generated by cutting out the idle time.

D. Module II: Build Markov Model given an Event Chains Dataset

This section defines what components constitute adopted Markov model and how to build Markov model given the dataset consisting of event chains of an employee.

Given the dataset D_i containing event chains of the i^{th} employee, $i = 1, ..., E$, the resulted Markov model based on the dataset should include following components:

1) A finite state set $S = \{s_1, s_2, ..., s_{ns}\}$ that contains all possible states of Markov model defined by the MMA mentioned in previous section and derived from D_i. Note that ns is the total number of derived states in Markov model;

2) An $1 \times ns$ initial probability (IP) vector, $IP = [ip_1, ..., ip_{ns}]$ where ip_i represents the probability that i^{th} state is the initial state of an event code segment, and $\sum_{i=1}^{ns} ip_i = 1$.

$$\forall\ i = 1, ..., ns,$$
$$ip_i = \frac{\text{\#event chains starting with } s_i \text{ in } D_i}{\text{\#event chains in } D_i}$$

3) An $ns \times ns$ transition probability (TP) matrix, as following:

$$TP = \begin{bmatrix} tp_{1,1} & \dots & tp_{1,j} & \dots & tp_{1,ns} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ tp_{i,1} & \dots & tp_{i,j} & \dots & tp_{i,ns} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ tp_{ns,1} & \dots & tp_{ns,j} & \dots & tp_{ns,ns} \end{bmatrix}$$

where for each i and j, $tp_{i,j}$ represents the transition probability from i^{th} state to j^{th} state, with constraints that $\sum_{j=1}^{ns} tp_{i,j} = 1, i = 1, ..., ns$.

$$\forall\ i, j = 1, ..., ns,$$
$$tp_{i,j} = \frac{\text{\#transitions from } s_i \text{ to } s_j \text{ in } D_i}{\text{\#transitions starting from } s_i \text{ in } D_i}$$

Given an observed event code segment, $c = [o_1, o_2, ..., o_T]$ of length T, o_t means the t^{th} observed Markov state in c. And the model probability (MP) that Markov model of (S, IP, TP) generates event code segment c can be calculated by:

$$MP(c) = ip_{o_1} \prod_{t=1}^{T-1} tp_{o_t, o_{t+1}} \ .$$

E. Module III: Probability Estimating given Markov Model

The anomaly detection mechanism for inside threat monitoring is implemented in Module III. After generating training dataset by using Module I, the output of Module II is personal behavioral model accompanied with a referenced probability, P_{ref}. This referenced probability is calculated by estimating the likelihood that this Markov model generates the training dataset of itself which is used to build the corresponding model. P_{ref} provides referenced usages that how well this resulted model fits the used training dataset and what is the likelihood that this employee produce corresponding logs when he did his daily routine jobs. Based on the P_{ref} and a user-defined threshold parameter, δ, the following equation (1) is designed to detect the anomaly.

$$Cond._i : \frac{NMP_i(Tr_i) - NMP_i(Dataset_{unseen})}{NMP_i(Tr_i)} \geq \delta$$

$$NMP_i(Dataset) = \sqrt[s(Dataset)]{\prod_{c \in Dataset} \sqrt[l(c)]{MP_i(c)}} \quad (1)$$

$$s(Dataset) = \text{size of } Dataset$$
$$l(c) = \text{length of } c$$

Tr_i represents training dataset for model building of employee $i = 1, ..., E$ and E is the maximum index of employees. Assume that $Dataset$ is a set of event segments given as input of employee i's learnt Markov model and c is any event segment belonging to $Dataset$. $NMP_i(\cdot)$ and $MP_i(\cdot)$ return the normalized and original model probabilities where i's learnt Markov model can generate the whole $Dataset$, respectively. The reason of using normalized probability is that Markov model has two characteristics: 1) the longer the length of an event segment has, the smaller resulted probability is; and 2) the more segments a dataset consists of, the smaller resulted probability of this dataset is. To cope with this situation and provide a fair evaluation between datasets or event segments with different sizes, we normalize the resulted probability, $MP_i(\cdot)$, of a $dataset$ to $NMP_i(\cdot)$ not only according to length of each event segment but also according to size of each of dataset with equation (1). Note that $NMP_i(Tr_i)$ is exactly the referenced probability P_{ref} for i^{th} employee, mentioned in the beginning of this subsection.

The idea of using equation (1) as anomaly detection mechanism is relative intuitive. Once the condition $cond._i$ is true, it means the probability of i^{th} employee's Markov model generating unseen dataset ($NMP_i(Dataset_{unseen})$) is relative lower than the i^{th} employee's referenced probability (P_{ref} or $NMP_i(\cdot)$) by a given ratio threshold δ. In this circumstance, it is quite unlikely that this $Dataset_{unseen}$ came from the i^{th} employee. Due to this hypothesis, when the $cond._i$ is true, it should trigger an alert of anomaly.

IV. Parameter Selection & Performance Evaluation

A. Experiment Settings

Considering both effectiveness and robustness of proposed method, the performance is fairly measured with a N-fold cross validation manner. Assume that D is the dataset containing event chains of all employees, then $D_1 \cup D_2 \cup ... \cup D_E = D$, where D_i is the dataset of event chains for i^{th} employee, $i = 1, ..., E$, and E is now 95 in current experiment. In N-fold cross validation, each D_i will first be partitioned into N folds. In each fold, one of the N parts is used for validation, named as $InnerVa_{ij}$, while the other $N-1$ parts are combined and formed as the so called $InnerTr_{ij}$ for model building. Note that $InnerTr_{ij} \cup InnerVa_{ij} = D_i$ for $j = 1, ..., N$.

According to our N-fold cross validation setting, in each fold j, for each employee i, the corresponding $InnerVa_{ij}$ will be used as the $Dataset_{unseen}$ in equation (1) to evaluate the model trained by $InnderTr_{ij}$. The model trained $InnderTr_{ij}$ will then try to differentiate whetehr the given input $Dataset_{unseen}$ belongs to corresponding account or not. In this experiment, $i = 1, ..., E$, $j = 1, ..., N$, while E and N are set to be 95 and 5. It will results in total $5 \times 95 \times 95 = 45,125$ testing cases.

In our proposed framework, there are three kinds of parameters needed to be determined. Following are the brief reviews of them, and the ranges of parameter value to be tuned in the following parameter selection.

1) θ: This is the longest allowed time interval between two subsequent event codes in one event code segments, and is used to divide a event sequence into two segments. In this experiment, θ is set to be 3 minutes, 6 minutes, and 9 minutes.
2) δ: The usage of this parameter is a anomaly detection threshold included in equation (1). In this experiment, δ is set to be 1%, 5%, and 10%.
3) MMA: a Markov-model-state annotation profile specifies which event code should be co-considered with certain attributes as a state in the Markov model. Note that for event code not to be listed in MMA means that use the default setting. The default setting now is using event code itself only. Note that, in this experiment, the possibilites of candidated MMA can be classified into two categories. The first category is without using any domain knowledge, such that MMA can be: a) for every event code using none of attribute (None Used MMA, i.e. default setting), b) for every event code using all kinds of attribute (All Used MMA), and c) for every event code using randomly selected attributes (Random MMA). The second one is based on the domain knowledge given by our cooperated domain experts working in TrendMicro company (TrendMicro MMA). Figure 5 shows the used domain knowledge-based TrendMicro MMA which annotates event codes 4768, 4769, and 4771 with additional attributes, respectively.

Annotation 1:

$$Event\_Code = 4768\ with$$
$$Field\_Name_1 = \text{“}return\ code\text{”},$$
$$Field\_Name_2 = \text{“}failure\ code\text{”},$$
$$Field\_Name_3 = \text{“}service\ name\text{”}.$$

Annotation 2:

$$Event\_Code = 4769\ with$$
$$Field\_Name_1 = \text{“}return\ code\text{”},$$
$$Field\_Name_3 = \text{“}service\ name\text{”}.$$

Annotation 3:

$$Event\_Code = 4771\ with$$
$$Field\_Name_1 = \text{“}return\ code\text{”},$$
$$Field\_Name_2 = \text{“}failure\ code\text{”},$$
$$Field\_Name_3 = \text{“}service\ name\text{”}.$$

Fig. 5. The Markov-model-state annotation profile from TrendMicro company.

B. Results & Discussions

As mentioned above, there are total 45,125 cases testing if the given input $Dataset_{unseen}$ belongs to Markov model being evaluated. The predicted result could be positive or negative. The positive case means the predicted label is "$Dataset_{unseen}$ does not belong to this account" and is also the anomaly case where the Management Information System (MIS) engineers are interested. On the other hand, the negative one represents that the input data belongs to this account. The following measurements are used to evaluate the proposed method and corresponding parameter setting. Table I, Table II, and Table III shows the different performance under different settings of maximum interval time θ = 3, 6, and 9 minutes. Based on the results of 5-fold cross validation , it can be observed that:

1) The behavioral modeling seems to take advantage of additional annotations when co-considering event code with annotated attributes as a Markov state. The effect is shown by that the TrendMicro MMA and All Used MMA obviously outperform than the other two, None Used MMA and Random MMA, in terms of recall and accuracy. Although the TrendMicro and All Used MMA perform almost exactly the same, the former is still more feasible than the latter in the realistic deployed environment. Because the space complexity of Markov model state is about $O(n^m)$ while n is the number of attributes, and there are m different values in each attribute. The MMA co-considering with all possible attributes may result in the number of Markov states exponentially increasing. Therefore, the TrendMicro MMA, incorporating significant prior knowledge, provides the best and rational results.
2) In this experiment, the longer the interval time θ is, the better performance of Markov model built can deliver. The idea of the proposed framework is to model user's complete operations as multiple instances of possible behaviors. The time interval of 3 minutes may be more likely too short to contain a complete operation than using 9 minutes as maximum allowed idle time. However, this setting should be customized according to the scenario of different deployed environments based on appropriate experiments for parameter selection.
3) It is no wonder that smaller values of δ will cause the whole anomaly detection mechanism more sensitive by making the $cond._i$ in equation (1) more easily to be trigger (i.e. increasing recall rate). On the other hand, despite larger values of $delta$ increase the threshold of trigger an anomaly alert, it still indeed enhances the certainty grades once any anomaly alerts are triggered (i.e. increasing precision rate). When setting the values of δ, it should concern the inevitable trade-off situation between recall and precision. Generally speaking, because the cyber security attacking causes huge amount of damage in the most cases, to make sure an acceptable recall rate is our first thumb rule.
4) Although, combining the prior domain knowledge form TrendMicro, AD^2 can not only produce highest performance in terms of recall and accuracy, but also may significantly reduce the number of possible states in Markov model state set (S) compared to all-used MMA. However,

AD^2 with TrendMicro MMA still can only produce about 66% recall rate or accuracy. It shows us that anomaly detection only based on AD log may be limited. Due to this reason, the future work of this study is inspired with that combining AD log with other various logs or contexts has opportunity to improve the performance of detecting anomaly.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$TP : \text{True Positive}, \quad TN : \text{True Negative}, \tag{5}$$

$$FP : \text{False Positive}, \quad FN : \text{False Negative}.$$

TABLE I. PERFORMANCE EVALUATION WITH θ = 3 MINUTES.

MMA	TrendMicro			All Used		
δ	1%	5%	10%	1%	5%	10%
Recall	64.81%	61.99%	57.98%	64.82%	61.99%	57.98%
Precision	99.09%	99.19%	99.29%	99.09%	99.19%	99.29%
Accuracy	64.60%	61.89%	58.01%	64.60%	61.89%	58.02%
MMA	None Used			Random		
δ	1%	5%	10%	1%	5%	10%
Recall	54.00%	51.77%	48.90%	56.86%	51.70%	45.14%
Precision	99.01%	99.10%	99.21%	99.02%	99.13%	99.21%
Accuracy	53.95%	51.82%	49.06%	56.76%	51.77%	45.36%

TABLE II. PERFORMANCE EVALUATION WITH θ = 6 MINUTES.

MMA	TrendMicro			All Used		
δ	1%	5%	10%	1%	5%	10%
Recall	64.88%	62,15%	58.30%	64.88%	62,15%	58.30%
Precision	99.10%	99.15%	99.28%	99.10%	99.15%	99.28%
Accuracy	64.67%	62.02%	58.32%	64.67%	62.02%	58.32%
MMA	None Used			Random		
δ	1%	5%	10%	1%	5%	10%
Recall	54.27%	52.17%	49.44%	56.20%	50.15%	42.11%
Precision	98.99%	99.04%	99.18%	99.13%	99.18%	99.26%
Accuracy	54.20%	52.18%	49.56%	56.17%	50.26%	42.40%

TABLE III. PERFORMANCE EVALUATION WITH θ = 9 MINUTES.

MMA	TrendMicro			All Used		
δ	1%	5%	10%	1%	5%	10%
Recall	66.60%	64.38%	61.37%	66.60%	64.38%	61.37%
Precision	99.07%	99.16%	99.25%	99.07%	99.16%	99.25%
Accuracy	66.34%	64.21%	61.32%	66.34%	64.21%	61.32%
MMA	None Used			Random		
δ	1%	5%	10%	1%	5%	10%
Recall	54.55%	52.92%	50.59%	53.22%	48.11%	41.64%
Precision	98.98%	99.07%	99.15%	99.10%	99.12%	99.20%
Accuracy	54.47%	52.92%	50.68%	53.24%	48.24%	41.93%

V. CONCLUSION

Not only because APT attacking takes a high degree of covertness over a long period of time, but also it usually cause lots of human efforts or financial damage. Efficient and effective inside threat monitoring becomes a hot issue during recent decade. In this project, unlike just using of expert rules with only a few pre-defined signatures, the idea of most likely state changing estimation is leveraged as a behavioral modeling technique. The logs of Active Directory domain service from every intra-net accounts was collected. And an anomaly detection framework based on famous Markov model algorithm was proposed to analyze AD log and to build the personal model for each account. Further a novel Markov-model state annotation (MMA) profile was also be incorporated during the model training and testing phases. Experiments on a dataset from a real environment of 95 employees shows that the proposed Markov-model based approach combined with TrendMicro prior knowledge will give the best performance of about 66.6% recall and 99.0% precision rates compared to model without using domain knowledge. The major advantages of of using TrendMicro MMA than using all-used MMA is that TrendMicro MMA consists of only a few Markov-model-state annotations such that it can significantly reduced the number of possible Markov states being concerned, compared to all-used MMA. However, even combining the prior domain knowledge, AD^2 only can produce about 66% recall rate or accuracy. That may gives us another conjecture that anomaly detection based on analyzing AD log may be limited by information which AD log can tell. This observation inspires our team that combining AD log with other various logs or contexts may be helpful to detect anomaly. A brief guideline of how to set up the parameters included in this framework is also provided according to the experimental result. The future works include: 1) keep improving the recall rate without sacrificing accompanied precision; 2) make a clustering analysis on the intra-net accounts to see whether different people behave like a group or not.

REFERENCES

[1] "Trend micro white paper on advanced persistent threat(apt)," Trend Micro Inc., Tech. Rep., 2013.

[2] H.-K. Pao, C.-H. Mao, H.-M. Lee, C.-D. Chen, and C. Faloutsos, "An intrinsic graphical signature based on alert correlation analysis for intrusion detection," in *Technologies and Applications of Artificial Intelligence (TAAI), 2010 International Conference on*. IEEE, 2010, pp. 102–109.

[3] "Directory system agent," Microsoft, MSDN Library, Tech. Rep., 2014, [Online; accessed: 6-May-2014]. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms675902(v=vs.85).aspx

[4] M. E. Russinovich and D. A. Solomon, *Microsoft Windows Internals: Microsoft Windows Server (TM) 2003, Windows XP, and Windows 2000 (Pro-Developer)*. Microsoft Press, 2004.

[5] "Active directory collection: Active directory on a windows server 2003 network," Microsoft, TechNet Library, Tech. Rep., 2015, [Online; accessed: 6-May-2015]. [Online]. Available: https://technet.microsoft.com/en-us/library/cc780036(WS.10).aspx

[6] "How the kerberos version 5 authentication protocol works," Microsoft, TechNet Library, Tech. Rep., 2015, [Online; accessed: 6-May-2015]. [Online]. Available: https://technet.microsoft.com/en-us/library/cc772815(v=ws.10).aspx

[7] J. R. Norris, *Markov chains*. Cambridge university press, 1998, no. 2.

[8] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[9] M.-Y. Chen, A. Kundu, and J. Zhou, "Off-line handwritten word recognition using a hidden markov model type stochastic network," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 5, pp. 481–496, 1994.

[10] A. D. Wilson and A. F. Bobick, "Parametric hidden markov models for gesture recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 9, pp. 884–900, 1999.

[11] "Markov model and hidden markov model," 2015, [Online; accessed: 1-May-2015]. [Online]. Available: http://www.csie.ntnu.edu.tw/ u91029/HiddenMarkovModel.html