ChainSpot: Mining Service Logs for Cyber Security Threat Detection

Jain-Shing Wu[1,2], Yuh-Jye Lee[1], Te-En Wei[1,2], Chih-Hung Hsieh[2*], Chia-Min Lai[1,2]
1: Department of Computer Science and Information Engineering,
National Taiwan University of Science and Technology, Taipei, Taiwan
2: Institute for Information Industry, Taipei, Taiwan
*: Corresponding Author
Email: jsw@iii.org.tw, yuh-jye@mail.ntust.edu.tw, iversonwei@iii.org.tw,
chhieh@iii.org.tw, senalai@iii.org.tw

***Abstract*—Given service logs of who used what service, and when, how can we find intrusions and anomalies? In this paper, a cyber threat detection framework - *ChainSpot* was proposed, in which the novelty is to build graphical patterns by summarizing user's sequential behaviors of using application-layer services, and to discover deviations against one's normal patterns. Besides modeling, the issue of justifying trade-off between feature explicity and computation complexity is properly addressed, as well. Effectiveness and performance of proposed method are evaluated using dataset collected in real circumstance. Experiments show that *ChainSpot* can provide very good supports for awaring abnormal behaivors which is starting point of threat detection. The detection results are highly correlated to expert-labeled ground truth, therefore, *ChainSpot* is proven helpful for saving forensics efforts significantly. Even more, case investigations demonstrate that the differences between benign and suspicious patterns can be further interpreted to reconstruct the attack scenarios. Then the analytic findings may be treated as indicators of compromise for threat detection and in-depth clues for digital forensics.**

I. Introduction

Detecting cyber threats inside enterprise is like finding needles in a haystack. "Advanced persistent threat"(APT) is a set of stealthy and continuous computer hacking processes. APT attackings 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. According to surveys from well-known security institutes and companies [1][2], only 31% victims have ability to discover they have been compromised or data breach internally, by themselves. It takes 205 days in average that threat groups would present on a victims network before being detected, where the worst case takes even about eight years. For such emergent and huge demand, companies and industries devote lots of time and resources to products for preventing users from being victims of APT, for instance, firewall rule or malicious website blacklist. Yet, due to various tricks of camouflages, such as packing, obfuscated shellcode, or virtual private network (VPN), signature-based malware detection technique, such as traditional intrusion detection system (IDS), is getting less useful, especially on identifying advanced persistent threat(APT) at post-compromised stage.

To complement the blind spot of traditional detection method and to alleviate the loading upon human forensics, novel APT-monitoring approaches, such as mining attacks based on huge amount of security device logs, have emerged in recent years [3][4][5][6][7]. T.-F. Yen et al. first used principle component analysis finding significant intrusion signatures based on large-scaled logs of security information and event management (SIEM) system to detect suspicious activities on hosts in an enterprise at 2013 [3]. Leveraging T.-F. Yen's research at 2013, the same team extended targeted scale to proxy logs, VPN logs, employee database, and client-side anti-virus report to decide risks of hosts in enterprise where hosts being dynamically logged by multiple accounts were ruled out according to Windows active directory authentication logs [4]. Research of A. Opera, et al. [5], considering information recorded by domain name system (DNS) and proxy, used known malicious domain names or infected hosts as seeds and adopted belief propagation algorithm to detect other potential comprised hosts and malicious domains. L. Invernizzi, et al. further investigated full HTTP requests from networks owned by internet service provider (ISP) and tried solving problem of detection malwares's distribtuions and infrastructures with global connection graph among malwares, hosts, accounts, or other entities involved in malware distribution [6]. Not only focusing on signature-based statistics measurements, H.-K. Pao et al., presented a research considering causal relationships of IDS sequential alerts to detect intrusion [7].

However, to our best knowledge so far, existing malwares or infected hosts detection mechanisms mostly belong to following two categories: 1) traditional methods used for intrusion detection only focus on network-layer anomaly discovery, such as firewall and IDS/IPS consistently tracing whether any pre-defined rule is triggered or not; 2) Researches of log analysis, usually estimate statistics measurements from various logs as signature features, without taking user's sequential behavioral anomaly into consideration. When facing to insider threats and APT attacks, methods with above shortcomings might be easily evaded. On the other hand, users' intentions, habits or tendencies are usually expressed when they using application-layer service. Due to this reason, instead of using expert rules with only a few pre-defined signatures or only

2324-9013/16 $31.00 © 2016 IEEE
DOI 10.1109/TrustCom.2016.284

network-layer recorded information, this paper emphasizes on highlighting deviations of user's behaviors from the viewpoint of application-layer services accessing. To achieve this goal, the proposed method *ChainSpot* consists of following essentials and analyzes user's sequential behaviors. 1) *ChainSpot* takes Windows active directory (AD) logs and proxy logs as time-series input data providing chronological evidences. Both of input logs are from application-layer services. The Windows AD domain controller of an organization monitors all related information when any intra-net account tries to logon or acquire various services, while proxy server acts as an intermediary between the user's computer and the Internet allowing client computers to make indirect network connections to other network services. 2) probabilistic graphical models are used for summarizing patterns describing user's sequential application-layer behaviors. As modeling user's behavioral pattern, a trade-off comes out in selecting appropriate features and also keeping computation simplicity. This issue was resolved in this research based on domain experience and knowledge survey. 3) the last key point of *ChainSpot* is to adopt appropriate criterion monitoring anomaly when large deviation from user's newly observed model to his own historical benign one.

Dataset and ground truths for evaluation were collected from real environment comprising thousands of people. Experiments show that 1) the modeling method of *ChainSpot* has significant effectiveness on measuring user's behavioral deviation when data of both normal and abnormal stages were given. 2) As deploying *ChainSpot* to anomaly monitoring, it results in very good supports for threat detection in which the detection result were highly correlated to expert-labeled ground truths. Proper setting for control parameter was also discussed and suggested with an efficient evaluating manner [8]. 3) Representative case investigations demonstrate that how the mechanism of *ChainSpot* works, and that the structural differences between benign and suspicious patterns, which effect *ChainSpot*'s prediction, could be further interpreted to reconstruct the cyber attacking scenarios. As a result, these analytic findings might be novel domain knowledge and threat detection clues for digital forensics.

In the remaining parts of this report, section II briefly introduces materials form Active Directory domain controller and proxy service, as well as the real dataset used for evaluation. Section III describes the detail components of the proposed *ChainSpot*. The effectiveness, performance, and case studies of the proposed framework are evaluated and discussed in Section IV. At last, Section V concludes this project.

II. Backgrounds & Materials

A. Windows Active Directory Domain Service

The Active Directory (AD) domain controller provides a directory service that Microsoft developed for Windows domain networks [9][10]. An AD domain controller authenticates and authorizes all users in a Windows domain type network assigning. 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 specified account is legal user in this domain even a system administrator [11]. The AD domain controller of an organization also monitors all related information when any accounts of this domain try to allocate or acquire various resources and services as well as help setting security policies for all computers or installing software update. Details about how AD controller works can be found in [12].

B. Proxy Service

In computer networks, the major role played by proxy is web proxy, facilitating access to content on the World Wide Web and providing anonymity [13]. A proxy server acts as an intermediary between intra-net clients and other servers on the internet to serve requests sent by clients seeking resources from other servers. The operation procedure of proxy is such as that: 1) an intra-net client, which needs some service on internet, first connects to the proxy server and tells its requesting. The requested service may be various types, such a file, connection, or web page, etc. 2) the proxy server then parses client's request and verifies whether this request obey pre-defined policy or not, based on what being requested or which website holds the requested service. 3) Once this request being accepted by proxy, the client will built a connection to server on internet for subsequent service requesting, otherwise this connection will be denied by proxy.

C. The Real Dataset

The materials used for performance evaluations of the proposed framework are introduced in this part. In this work, a certain government organization of around 1,000 employees in Taiwan was selected as the deployment environment of the proposed method. Materials collected from this organization comprise three kinds of data. Two of them are service logs, AD and proxy logs, for user's behavior modeling and anomaly detection. While the other one is security operation center (SOC) event tickets, labeled with various security anomaly events by forensics experts providing ground truths specifying which IP related to certain accounts behaves abnormal or not. The descriptions and statistics about the used real dataset are as followings. The Active Directory domain service of Windows Servers 2008 R2 version is mounted on this organization's domain network and keeps monitoring all service requests and resource allocations raised from intra-net accounts. In this circumstance, Total 27,902,857 logs was collected during one months, from 2016/08/01 to 2016/08/31. The proxy logs in the same duration contains 78,044,332 logs. All log data supplying information of user's behaviors will be dispatched to each account's profile based on the recorded field "account name". The full size about the collected data (AD and proxy logs) is 152.299 gigabytes. On the other hand, totally 99 SOC event tickets among 6 types were generated in August/2016 and were used as source of ground truth, because SOC tickets will indicates which accounts are attacked or doing something strange at which time or in which duration. The number of

abnormal accounts correlated by SOC tickets is 1,089. Due to the confidential concern, for any consultant or interest about this real data, please contact to the corresponding author. Table I lists all numbers of SOC-tickets collected from the target organization.

TABLE I
NUMBER OF DIFFERENT SOC TICKETS

Index	Type of SOC ticket	#tickets
1^{st}	Single account failed too many times when logon	4
2^{nd}	Multiple accounts logon from single IP in short time	5
3^{rd}	Host connects to malicious domain	11
4^{th}	Buffer overflow attack from outside	19
5^{th}	DoS attack from outside	59
6^{th}	Try to logon in non-working hours	1

III. THE PROPOSED METHOD - *ChainSpot*

This section and following subsections define what components constitute adopted Markov chain and how to build it given the dataset consisting of logs from an employee. In probability theory, a Markov chain 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)[14]. 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 [15], hand-written text recognition [16], gesture recognition [17], and cyber security intrusion detection[7], based on Markov chain itself or its popular variant, hidden Markov model.

A. State Constitution of Used Markov Chain

The first subsection gives the explanation about which information, from collected AD and proxy logs, constructs the states of Markov chain. The goal of *ChainSpot* is to model user's behaviors which can express their intention or anomaly if something strange happens. However too many information without interpretable meaning may simply worsen computational complexity. For this reason, a feature selection strategy in data pre-processing of *ChainSpot* will decide which kind of information should be included in building Markov chain. The major principles consists of two parts. 1) Logs driven spontaneously by machine for protocol acknowledgement or information exchanging do not directly reflect users' intention, such that these kind of logs (for instance, AD event codes "4624" and "4634") will be ruled out and not be considered. 2) For data fields of highly describing users' intentions and habits (such as "Accessed Domain Name" in proxy logs) will be retained to form Markov states.

The final used state constitution in AD logs contains:

1) All active directory event codes, except event codes "4624" and "4634", was included [18].

2) Event code "4771" in which the meanings is "Kerberos Authentication Failed" will be considered accompanied with one corresponding field-"reply code", where "0x12" means that "this account has been locked out" and "0x18" stands for "wrong password" [19].

And the following fields are included in Markov state for proxy logs modeling:

1) "Adopted HTTP Method" represents value of "GET" or "POST" indicating user's query action.

2) "Download or Upload" represents the packet flow direction of access connection.

3) "Accessed Domain Name" stores the second-level domain name where this user visited.

4) "Access Result" records value of "allowed" or "denied", which is the final judgement whether pre-set proxy policy allows this user's connection or not.

Consequently, the Markov chain based on AD logs is a sequence of all produced windows AD event codes, where event codes "4624" and "4634" was excluded and event code "4771" was accompanied with additional field "reply code". On the other hand, the state constitution for proxy Markov chain is a sequence of integrated field information, such as "try to *GET* and *Download* on *microsoft.com* then this action was *denied*".

B. Build Markov Chain given Log Sequences Dataset

Given the dataset D_i containing log sequences of the i^{th} accounts, $i = 1, ..., E$, the resulted Markov chain based on the dataset should includes following components:

1) A finite state set $S = \{st_1, st_2, ..., st_{ns}\}$ that contains all possible states of Markov chains defined in the previous subsection and derived from D_i. Note that ns is the total number of derived states in D_i;

2) An $ns \times ns$ transition probability matrix (TPM), as following:

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

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

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

It should be noted that for general setting, there is one additional parameter describing Markov chain, a $ns \times 1$ initial probability vector. The real-valued elements in initial probability vector represents the probability for any state being the initial state of an event chain. However, due to the reality in the practical environment that it is almost impossible to precisely partition user's sequential behaviors into several independent behaivor segments, *ChainSpot* omits usage of initial probability vector by just taking all behaviors in one user's profile as a single but long log sequence to simplify data preprocessing.

C. Deviation Estimating given Different Markov Chains

The anomaly detection mechanism of *ChainSpot* relies on appropriate estimating deviation from user's new profile to his benign one. In this paper, user's profiles are summarized as patterns represented by Markov chains, as a result, deviation measuring can be realized by using graph edit distance (GED) to quantify similarity (or dissimilarity) between different Markov chain models. The formal graph edit distance between two graphs G_1 and G_2, written as $GED(G_1, G_2)$ can be defined as following:

$$GED(G_1, G_2) = \min_{(e_1,...,e_k) \in P(G_1,G_2)} \sum_{i=1}^{k} cost(e_i), \quad (1)$$

where $P(G_1, G_2)$ denotes the universal set of editing paths isomorphically transforming G_1 into G_2, and $cost(e_i)$ is the cost of each graph editing operation, e_i.

With respect to Markov chains in *ChainSpot*, calculation of GED on two Markov chains can then be implemented by following equation (2):

$$GED(TPM^1, TPM^2) = \sum_{i=1}^{ns} \sum_{j=1}^{ns} \left| tp_{i,j}^1 - tp_{i,j}^2 \right|, \quad (2)$$

where TPM^1 and TPM^2 are transition probability matrices of two Markov chains, as well as $tp_{i,j}^1$ and $tp_{i,j}^2$ are the corresponding transition probabilities in TPM^1 and TPM^2, respectively. The ns is the number of state in Markov chain.

IV. Experiments

The main objective of this section is to demonstrate the effectiveness and performance of *ChainSpot*. The first sub-section is a reconnaissance survey of the effectiveness when using *ChainSpot* to catch up the deviation deviation abnormal case to normal one for each account in each SOC ticket. The second experiment is a general performance evaluation on prediction ability of treating *ChainSpot* as an anomaly detection framework in terms of recall and prediction, compared to alternative baselines. The last sub-section explains what reasons lead *ChainSpot* to successfully differentiate abnormal cases, and how these findings will benefit to digital forensics team and used as indicators of compromised.

A. Effectiveness of behavior deviation measuring using ChainSpot

This experiment is a reconnaissance survey to check whether each Markov model of abnormal account results in larger deviation compared to this account's benign model and normal cases. For each account which is ever labeled as "abnormal" by some SOC event tickets, three phases of log data for this account with different meanings and usages can be generated:

1) "abnormal testing phase": this phase of logs came from 3-days duration before the date that SOC ticket was signed. A corresponding abnormal Markov chain, named as $M_{abnor.}^i$, is used representing i^{th} account's abnormal behaviors.

2) "training phase": this phase of logs for one account is the former half of whole logs excluded logs of "abnormal testing phase". Note that behaviors from this phase are all assumed to be benign ones, and notation, $M_{tr.}^i$, represents Markov chain for this phase of i^{th} account.

3) "normal testing phase": this phase consists of one's all logs excluded the former two phases and are also assumed to be benign. Markov chain $M_{nor.}^i$ is used for summarized i^{th} account's behaviors of this phase.

The successful condition in following reconnaissance survey over total E accounts labeled by SOC tickets is regarded as following:

$$\forall\ i = 1, ..., E,\ i^{th} \text{ account is regarded as:} \\ Successful, \text{ If } GED(M_{abnor.}^i, M_{tr.}^i) > GED(M_{nor.}^i, M_{tr.}^i). \\ Failed, \text{ otherwise.} \quad (3)$$

Table II lists all accounts labeled by SOC tickets in terms of different event names, as well as the corresponding successful rates. In the real dataset, each SOC ticket index, indicating different anomaly or attacking, leads to various number of abnormal accounts, as well as different kinds of logs will link to different types of attacking. For instance, AD logs link to 2^{nd}-type ("Multiple accounts logon from single IP in short time") SOC tickets and result in 865 accounts, while proxy logs mainly contribute 255 accounts of 3^{rd}-type ("Host connects to malicious domain") SOC tickets. Table II shows that *ChainSpot* provides significantly better effectiveness compared to uniform baseline (about 50.00%), except on cases of 1^{st}-type ("Single account failed too many times when logon") tickets (50.00%). The general effectiveness measuring gives 85.66% and 87.17% success rates in terms of averaging on various types or averaging on different accounts, respectively.

Another useful observation found from the deployed organization can be reasonably leveraged to narrow set of accounts being detected and to improve the current effectiveness of *ChainSpot.* The dynamic host configuration protocol (DHCP) used in the experimental organization allows user dynamically logging on different intra-net source IPs. Thus, using "source IP" field of SOC tickets to correlate suspicious accounts might unintentionally includes some innocent accounts logging on the same IP where abnormal accounts logon, too. To rule out

TABLE II
RECONNAISSANCE SURVEY FOR EFFECTIVENESS OF *ChainSpot*. THE EXPLANATION FOR EACH SOC TICKET INDEX CAN BE REFERRED TO TABLE I.

AD logs related			
Ticket index	#Related Accounts	#Succ. Accounts	Succ. Rate
1^{st}	2	1	50.00%
2^{nd}	865	791	91.45%
5^{th}	3	3	100.00%
6^{th}	2	2	100.00%
Proxy logs related			
Ticket index	#Related Accounts	#Succ. Accounts	Succ. Rate
3^{rd}	255	185	72.50%
4^{th}	3	3	100.00%

these mistaking, a threshold "adhesion degree" is to measure how the extent for a user routinely using same IP to logon or do web browsing is. Accounts who do not achieve preset "adhesion degree" requirement will be filtered out. Table III and table IV list the major changes on quantities of accounts related to some SOC tickets, the filtering results, and the corresponding successful accounts using different "adhesion degree" criterion. Table III adopts the probability ratio of users' mostly used IPs, while table IV calculates entropy based on all IPs of a user ever used as "adhesion degree" measuring. Generally speaking, applying "adhesion degree" thresholds improves the success rate , in spite of that different "adhesion degree" calculation mainly contributes on different SOC ticket type. Using "adhesion degree" filtering strategy is essentially another trade-off situation between recall and precision. Accounts with lower adhesion degree will be ruled out as threshold increasing, however it will leads the judgement of *ChainSpot* having more confidence such that the success rate will be improved. Figure 1 focuses on security events containing most accounts, and further depicts the performance changing when applying different threshold setting. Users of *ChainSpot* can realize the trend and convergence of performance trade off according to figure 1, and can decide the appropriate value of "adhesion degree" threshold based on their preference between precision and recall.

TABLE III
THE MAJOR CHANGES ON QUANTITIES WHEN APPLYING PROBABILITY RATIO FILTERING STRATEGY TO ACCOUNTS OF DIFFERENT SOC TICKETS.

AD logs related			
	Threshold = 0.0	Threshold = 0.7	Threshold = 0.9
Ticket index	Succ. Rate	Succ. Rate	Succ. Rate
1^{st}	1/2	1/2	1/2
2^{nd}	791/865	552/588	212/220
5^{th}	3/3	3/3	3/3
6^{th}	2/2	1/1	1/1
Proxy logs related			
	Threshold = 0.0	Threshold = 0.7	Threshold = 0.9
Ticket index	Succ. Rate	Succ. Rate	Succ. Rate
3^{rd}	185/255	108/130	48/51
4^{th}	3/3	1/1	1/1

TABLE IV
THE MAJOR CHANGES ON QUANTITIES WHEN APPLYING ENTROPY FILTERING STRATEGY TO ACCOUNTS OF DIFFERENT SOC TICKETS.

AD logs related			
	Threshold = 1.0	Threshold = 0.4	Threshold = 0.2
Ticket index	Succ. Rate	Succ. Rate	Succ. Rate
1^{st}	1/2	1/2	1/2
2^{nd}	791/865	656/700	377/392
5^{th}	3/3	3/3	3/3
6^{th}	2/2	2/2	2/2
Proxy logs related			
	Threshold = 1.0	Threshold = 0.4	Threshold = 0.2
Ticket index	Succ. Rate	Succ. Rate	Succ. Rate
3^{rd}	185/255	168/237	46/58
4^{th}	3/3	1/1	1/1

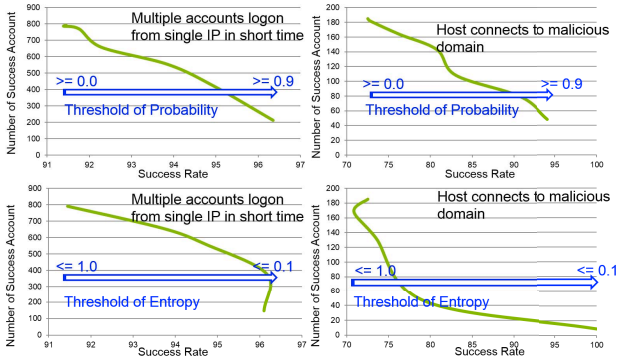


Fig. 1. Effectiveness changing given different filtering thresholds with respect to major security events, 2^{nd} and 3^{rd} types.

B. Performance of anomaly detection using ChainSpot

The ultimate goals of *ChainSpot* is to build the behavioral model for each account and to detect anomaly once accounts do not act like themselves compared to their historical daily routine. For this purpose, we formalize following experiment as a training-detection scenario. Totally, the collected dataset comprises 1,089 accounts, and 3 Markov chains were built to be instances of different usages for each account's profile. Just like that mentioned in subsection IV.Part A, for each account's profile, logs in "training phase" lead to one benign Markov chain equivalent to one training instance, while Markov chains of "abnormal testing phase" and "normal testing phase" result in two testing instances, one for positive (abnormal) case and the other for negative (normal) case, respectively. It should be noted that the usage of *ChainSpot* in this scenario is much different compared to usage in previous experiment. In training-detection framework, label ("abnormal"(negative) or "normal") of each testing instance is obviously default to be unknown, such that equation (3) is no longer appropriate. Following equation is then be defined to help *ChainSpot* making prediction when unknown instances arrive. $\forall i = 1, ..., E$, given an unseen Markov chain M^i_{unseen} which is known belonging to i^{th} account, the predicted label of M^i_{unseen} is classified as:

$$\begin{array}{l} Abnormal, \text{ once } GED(M^i_{tr.}, M^i_{unseen}) \geq \delta, \\ Normal, \text{ for otherwise,} \end{array} \quad (4)$$

where $M^i_{tr.}$ is the Markov chain representing i^{th} account's historical benign model, $GED(\cdot)$ is the graphical edit distance defined in equation (2), and δ is a user-defined threshold. The idea of using equation (4) as anomaly detection mechanism is relatively intuitive. Once the graph edit distance between $M^i_{tr.}$ and M^i_{unseen} is larger than δ, it means the unseen behaviors of i^{th} account deviate the same account's historical pattern to an unallowable extent. In this condition, anomaly alert should be triggered.

In this anomaly detection experiment, there are totally 1,089 abnormal (positive) instances, and 1,089 normal (negative) instances. Table V shows the performance results of *ChainSpot* under different deviation threshold δ of 4, 6, and 8, respectively. Taking $\delta = 6$ seems to result in a general well-balanced prediction of 0.71 $Precision$ and 0.81 $Recall$ rates. Based on that, table VI then compares *ChainSpot* of $\delta = 6$ with three alternative baselines, including predictors always output "positive", always output "negative", and output randomly. It can be obviously observed that the performance of *ChainSpot* significantly outperforms performances from those naive baselines.

It should be noted that different setting of δ will gives different preference between $Precison$ and $Recall$, and "cost curve" is an efficient technique evaluating different predictions method with a cost-sensitive manner [8]. Figure 2 depicts cost curve to gives suggestion on how to select appropriate δ value when *ChainSpot* deploys. For a binary prediction problem like this experiment, the cost curve considers performances of each candidate predictor (δ = 4, 6, and 8) on following two dimensions:

x-axis : "Probability cost" ($PC(+)$) is defined by combining the penalty costs of wrong prediction and data distribution for each class:

$$PC(+) = \frac{p(+)cost(-|+)}{p(+)cost(-|+) + (1 - p(+))cost(+|-)}, \quad (5)$$

where $p(+)$ is the probability of positive samples, $cost(-|+)$ and $cost(+|-)$ are penalty costs of false negative and false positive cases.

y-axis : "Normalized expected cost" (NEC), defined as following, indicates expected predictor performance:

$$NEC = Rate_{FN} * PC(+) + Rate_{FP} * (1 - PC(+)), \quad (6)$$

where $Rate_{FN}$ and $Rate_{FP}$ is the ratios belong to $[0, 1]$ that false negative and false positive errors occur. It should be noted that the smaller value of NEC is, the better performance of candidate predictor can provide.

When deploying *ChainSpot* to a certain organization, the member of security team should consider the value of $PC(+)$ of their own environment based on the frequencies of abnormal security events occurring as well as the cost when security event occurs. For instance, according to figure 2, a suggestion of δ value is that:

$$\begin{cases} \delta = 4, \text{ if } PC(+) \leq 0.33, \\ \delta = 6, \text{ if } 0.33 < PC(+) \leq 0.67, \\ \delta = 8, \text{ if } 0.67 < PC(+), \end{cases} \quad (7)$$

where this strategy provides expected performance of $NEC \leq 0.28$

TABLE V
PERFORMANCE OF ANOMALY DETECTION UNDER DIFFERENT DEVIATION THRESHOLD. $\#TP$, $\#TN$, $\#FP$, AND $\#FP$ ARE NUMBERS OF TRUE POSITIVE, TRUE NEGATIVE, FALSE POSITIVE, AND FALSE NEGATIVE SAMPLES, RESPECTIVELY.

Measurements	$\delta = 4$	$\delta = 6$	$\delta = 8$
$\#TP$	1050	882	645
$\#FP$	674	362	218
$\#TN$	415	727	871
$\#FN$	39	207	444
$Precision$	0.61	0.71	0.75
$Recall$	0.96	0.81	0.59

TABLE VI
PERFORMANCE COMPARISON BETWEEN *ChainSpot* AND THREE BASELINES, INCLUDING PREDICTOR ALWAYS OUTPUT "POSITIVE", ALWAYS OUTPUT "NEGATIVE", AND OUTPUT RANDOMLY

Measurements	*ChainSpot* with $\delta = 6$	always "Pos."	always "Neg."	Randomly output
$\#TP$	882	1089	0	546
$\#FP$	362	1089	0	541
$\#TN$	727	0	1089	548
$\#FN$	207	0	1089	543
$Precision$	0.71	0.50	Null	0.50
$Recall$	0.81	1.0	0	0.50

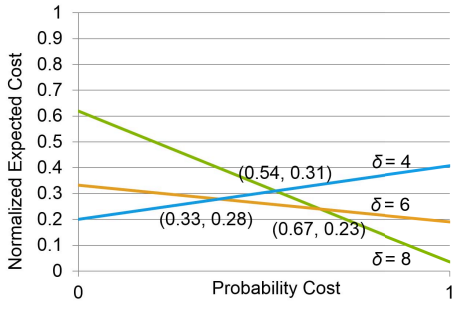


Fig. 2. Cost curve describing distributions of normalized expected cost under different δ and probability cost.

C. Representative demonstration & case study of ChainSpot

The final part of the experiment is demonstrative case studies of what essentials *ChainSpot* will catch to differentiate normal and abnormal cases, and how these essentials found by *ChainSpot* will benefit to domain experts as novel forensics knowledge. The following demonstrations on AD and

proxy logs also illustrates that how mechanism of *ChainSpot* works, and that the mechanism is quite different from existent common methods. When deploying to real circumstance, *ChainSpot* can be a complementary threat-detecting technique and be used concurrently with other alternatives.

1) Case I - Detecting "Multiple accounts logon from single IP in short time" on AD logs: The 2^{nd} event - "Multiple accounts logon from single IP in short time" means that a policy violation when too many accounts logon from an outside IP within a short duration, and it is usually a prognostic followed by account compromise. The first discussion is for one representative account A_{case1} of this abnormality. Figure 3 depicts the corresponding three patterns summarized from sequential behaviors recorded in AD logs. Each Markov chain is dedicated to user's behaviors of one phase. And vertices represent event codes included in Windows Active directory protocol [18], while directional edges indicates the transition probability among different events. The wider and darker for how an edge is depicted, the stronger probability of this transition means. In general, comparing all patterns (Markov chains) in figure 3, it can be obviously found that the structural difference between training phase (figure 3(a)) and abnormal instance (figure 3(c)) is much larger than that between training phase and normal instance (figure 3(b)). Further, after deep investigation the main differences leading *ChainsSpot*'s prediction in terms of causalities in vertices and edges, behaviors constitute abnormal pattern have the following characteristics differing from training and normal testing phases:

1) **Without "4661" to "4662":** the event "4661" means "a session handler to an object was requested", while "4662" represents "an operation was performed on a requested object".
2) **Without "4662" to "4658":** the event "4658" means "a session handler to an requested object was released".
3) **With loops from "4768" to "4771 0x18":** the event "4768" means "an Kerberos authentication ticket was requested", and the "4771 0x18" occurs when authentication fails because of entering wrong password.

Combining the original meaning of the original event, the above example shows that attackers try to logon lots intranet accounts ,one by one, in order to get access permission of a certain object or service in this organization. This kind of intelligence can help cyber security member to recommend every intra-net users to renew their password complexity, as well as enhance protection on certain assets being targeted by attackers with additional information about what resource attackers interest.

2) Case II - Detecting "Host connects to malicious domain" on proxy logs: Another illustration is about *ChainSpot* detects an account A_{case2} engaging in event of "Host connects to malicious domain" based on behavioral patterns summarized from proxy logs. Again, Markov chains representing data of A_{case2}'s training, normal testing, and abnormal testing phases was built by *ChianSpot*. According to subsection III.Part A, the states and transitions in proxy Markov chains of *ChianSpot* look like the following form: **From "Get and Download from google.com then failed" to "Get and Upload from facebook.com then failed"**. By means of investigating vertices and transitions from three-phase Markov chains, figure 4 shows accumulated counts of domains where A_{case2} accessed in abnormal phase, but never in the duration of the other two. The x-axis of figure 4 lists these domains that A_{case2} indeed accessed during abnormal stage, but never in the other two phases. The y-axis shows the accumulated accessing counts to corresponding domains. It should be noted that domain "moneydj.com" has been proven as malicious website, and never occurs in A_{case2}'s training and normal profiles. Further, implicit behavioral anomaly can also be detected by comparing the state constitutions of different A_{case2}'s profile. Figure 5(a) and 5(b) show accumulated visiting count of common domain where A_{case2} accessed during both of training and normal testing stages. It can be obviously noted that the most five of frequently visited domains is 1) "amazon.com", 2) appledaily.com.tw, 3) chartbeat.net, 4) edgesuit.net and 5) facebook.com. However, none of these well-known social platforms, news, or common websites was being accessed during tA_{case2}'s abnormal phase. This is a highly suspicious situation and the security team should carefully investigate this anomaly to check whether A_{case2} was already compromised or not.

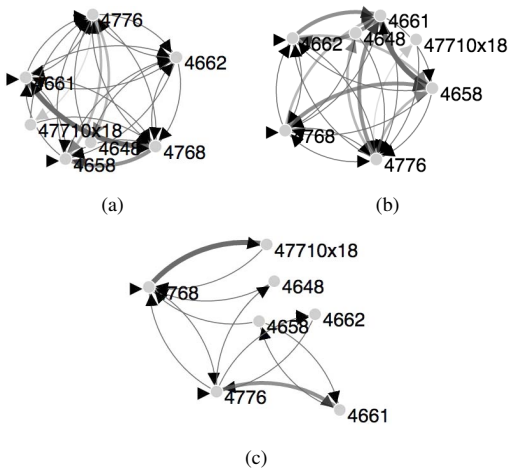


Fig. 3. Patterns of user A_{case1} who ever participated in "Multiple accounts logon from single IP in short time" event. Figure 3(a), 3(b), and 3(c) represent Markov chains summarizing user's historical benign (training), normal testing, and abnormal testing phases, respectively.

V. Conclusion

To complement the blind spot of relying upon human forensics or traditional techniques, this research firstly proposes a cyber threat detection framework - *ChainSpot*, focusing on both application-layer AD and proxy sequential logs analysis. Novelty of *ChainSpot* mainly comes from building graphical patterns by summarizing user's application-layer sequential behaviors as Markov chains, and from monitoring deviations against one's normal behaviors. All practical issues with

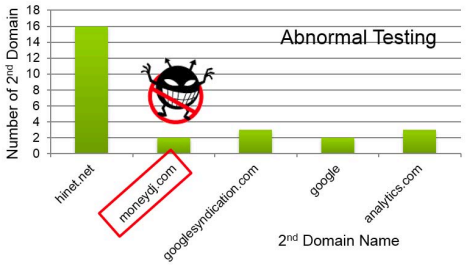
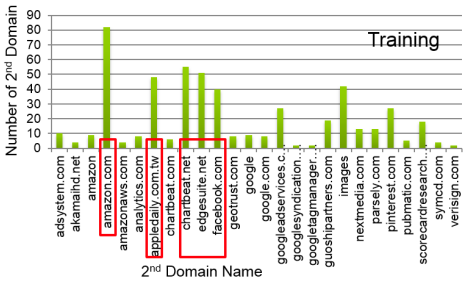
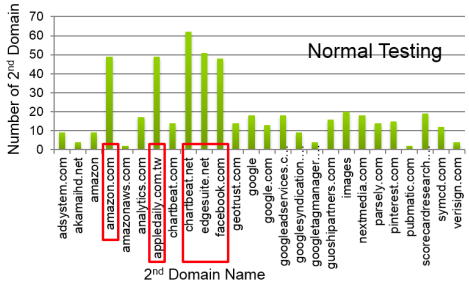


Fig. 4. Access records extracted from abnormal-phase Markov chain of account A_{case2} who participated in "Host connects to malicious domain" event. Note that domain "moneydj.com", which has been proven as malicious website, never occurs in A_{case2}'s profile.



(a)



(b)

Fig. 5. Access records extracted from Markov chains of account A_{case2} who participated in "Host connects to malicious domain" event. Figure 5(a) and 5(b) list accumulated counts accessing to domains where A_{case2} ever visited during both historical benign (training) and normal testing phases, respectively.

respect to implementing *ChainSpot*, such as trade-off between choosing enough features and also keeping computation simplicity, were carefully addressed with appropriate domain knowledge. Experiment results show that *ChainSpot* is able to alleviate lots of human efforts on monitoring or forensics. Besides, case demonstrations show that the differences between benign and suspicious patterns could be organized as threat intelligence used for describing attacking scenarios. The next-step future work for this research is try to combine and align heterogeneous logs, such as active directory, proxy, firewall, etc., into a super timeline, and try to model the benign/malicious patterns.

ACKNOWLEDGMENT

At the first, the authors would like to especially thank to prof. Christos Faloutsos of Dept. of Computer Science, Carnegie Mellon University for his kindly long-term consultant and advice to this research work. Further, this research is supported in part by the Ministry of Science and Technology of Taiwan under grants number MOST 102-2218-E-011-011-MY3.

REFERENCES

[1] "M-Trends 2015: A VIEW FROM THE FRONT LINES," 2015.
[2] "Trend micro white paper on advanced persistent threat(apt)," Trend Micro Inc., Tech. Rep., 2013.
[3] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013, pp. 199–208.
[4] T.-F. Yen, V. Heorhiadi, A. Oprea, M. K. Reiter, and A. Juels, "An epidemiological study of malware encounters in a large enterprise," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 1117–1130.
[5] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 2015, pp. 45–56.
[6] L. Invernizzi, S. Miskovic, R. Torres, C. Kruegel, S. Saha, G. Vigna, S.-J. Lee, and M. Mellia, "Nazca: Detecting malware distribution in large-scale networks." in *NDSS*, vol. 14, 2014, pp. 23–26.
[7] 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.
[8] R. C. Holte and C. Drummond, "Cost-sensitive classifier evaluation using cost curves," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2008, pp. 26–29.
[9] "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
[10] 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.
[11] "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
[12] "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
[13] M. Shapiro, "Structure and encapsulation in distributed systems: the proxy principle," in *icdcs*, 1986, pp. 198–204.
[14] J. R. Norris, *Markov chains*. Cambridge university press, 1998, no. 2.
[15] 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.
[16] 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.
[17] 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.
[18] "Windows security log events," 2016, [Online; accessed: 17-April-2016]. [Online]. Available: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/Default.aspx
[19] "Event code 4771: Kerberos pre-authentication failed," 2016, [Online; accessed: 17-April-2016]. [Online]. Available: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4771