MalSpot: Multi^2 Malicious Network Behavior Patterns Analysis

Ching-Hao Mao[1], Chung-Jung Wu[1], Evangelos E. Papalexakis[2], Christos Faloutsos[2], and Kuo-Chen Lee[1]

[1] Institute for Information Industry, Taipei, Taiwan
{chmao,cklonger,kclee}@iii.org.tw
[2] Cargegie Mellon University, Pittsburgh, PA, USA
{epapalex,christos}@cs.cmu.edu

Abstract. What are the patterns that typical network attackers exhibit? For a given malicious network behaviors, are its attacks spread uniformly over time? In this work, we develop MalSpot, multi-resolution and multi-linear (Multi^2) network analysis system in order to discover such malicious patterns, so that we can use them later for attack detection, when attacks are concurrent with legitimate traffic. We designed and deployed MalSpot, which employs multi-linear analysis with different time resolutions, running on top of MapReduce (Hadoop), and we identify patterns across attackers, attacked institutions and variation of time scales. We collect over a terabyte of proven malicious traces (along with benign ones), from the Taiwanese government security operation center (G-SOC) , during the entire year of 2012. We showcase the effectiveness of MalSpot, by discovering interesting patterns and anomalies on this enormous dataset. We observed static and time-evolving patterns, that a vast majority of the known malicious behavior seem to follow.

Keywords: multi-resolution, tensor, anmoaly detection, multi-linear, uncorrelated levels

1 Introduction

In today's wide interconnected world, malicious network attacks have a long incubation period, and as a result, existing state-of-the-art information security/data analysis mechanisms find it very challenging to compete against those attacks in a timely manner. Information security monitoring enterprises have limited information and, hence, fail to see the big picture of the attacks that are being orchestrated. However, due to it's immense scale, the Internet provides us with a large variety of data, both structured (e.g. logs), as well as unstructured, that can be used in aid of information security analysis. Thus, today's Internet's scale calls for big data analysis techniques. The main focus of the present work is

to investigate large-scale and stealthy malware behaviour. Our analysis is based on considerable number of logs from real security information event management systems.

Detection of stealthy attacks is particularly challenging, since, statistically, they are hard to distinguish from normal connections. Furthermore, obtaining attack data (e.g. through a network sniffer or a honeynet) poses challenges in its own right. Dainotti et al.[1] employ a horizontal scan of the entire IPv4 address space, in order to detect attacks created by the *Sality* botnet. Chen et al.[8] propose a scalable network forensic mechanism for stealthy, self-propagating attack detection; However, both these state-of-the-art methods are specialized in terms of the attacks they target. Therefore, there is still need for a tool that is able to identify attacks without specific assumptions of their characteristics or their behaviour/propagation pattern.

Key to the discovery of malicious patterns is the summarization of the network behaviour, characteristics, and propagation of a connection. Therefore, we need a systematic and scalable approach which is able to effectively summarize large heterogeneous pieces of data that represent different aspects of the network. Such tools can be drawn from time evolving graph mining and tensor analysis literature. In particular tensors or multi-dimensional arrays have appeared in numerous interesting application including clustering, trend & anomaly detection [6], and network forensics [3]. In [13] the authors propose GigaTensor, a scalable, distributed algorithm for large scale tensor decomposition. In this work, we leverage GigaTensor to the end of stealthy malware detection, without assuming prior knowledge on the malware's behaviour.

A more formal definition of the problem at hand is as follows

Problem 1. Attack Patterns Discovery in MalSpot

- **Given**: (1) intrusion detection system (IDS) event logs, recording ⟨ event_name, timestamp, target_ip ⟩ (2) Honeynet firewall logs, recording ⟨ source_ip, target_ip, timestamp ⟩
- **Find**: (1) the suspicious and common patterns in all three modes/aspects of the data, (2) provide an intuitive visualization of the above patterns, and (3) scale up in millions of nodes in our network.

Guided by the format of the data at hand, we propose MalSpot which choses to formulate the problem as multi-linear solution as well as tensor analysis. Additionally, we propose to experiment with the granularity of the time window in our data; hence, we propose a *multi-resolution* approach, which will be able to identify different types of anomalies, in *uncorrelated* levels of temporal granularity.

Definition 1 (Uncorrelated Levels) *Two different levels of temporal granularity are called* uncorrelated*, if the network behavior in those levels, for a particular node, or set of nodes, is significantly different. For instance, a set of nodes may experience different patterns in network traffic in an hourly scale, as opposed to a daily scale.*

By leveraging multi-level, multi-linear analysis of the aforementioned data, we are able to conduct scalable and efficient anomaly detection. Our main contributions are the following:

- **Design** of the MalSpot system: leveraging `hadoop`, our proposed system can scale up to arbitrarily big datasets, and it is scales near linearly with the network trace size (number of non-zero entries).
- **Discoveries:** we report that attacks come in different flavours: there are, for example, attacks that are particularly short in time and stop showing after a period of time, while other attacks are more persistent, or focus only on a specific port.

The benefit from using MalSpot as opposed to standard techniques is the fact that by doing so, we are able to detect correlations of entities participating in a heterogeneous network for a very long term, and additionally detect multi-aspect correlations of entities (e.g. using the port number as a dimension), the comparison is as shown in Table 1.

Table 1. Qualitative analysis of commercial SIEM event analysis packages, as compared to our proposed method.

	Time Granularity	Anomaly Detection	Pattern Discovery
MalSpot	*Second to Year*	✓	✓
Splunk	Second to Day	**x**	**x**
ArcSight (HP)	Second to Min	**x**	**x**

The rest of this paper is organized as follows: related work is outlined in Section 2. We first elaborate on MalSpot, in Section 3, we then provide experimental studies in Section 4. Finally, Section 5 concludes the discussion and highlights future directions.

2 Related Work

For handling huge collections of time-evolving events, [9] proposes a multi-resolution clustering scheme for time series data using k-means clustering and progressively renes the clusters. In order to discover the streaming pattern in multiple time-series, [11] propose SPIRIT (Streaming Pattern dIscoveRy in multIple Time-series) which can incrementally find correlations and hidden variables by means of using principal components analysis (PCA) and singular value decomposition (SVD) to summarize the key trends in the entire stream collection. In [10] the authors propose TriMine which consider multiple features to provide hidden topics modeling and forecast future events. In [4] the authors apply SVD

to compress sensor data sequences by decomposing them into local patterns and weight variables.

Tensors and tensor decompositions have been extensively used in a variety of fields, including but not limited to Data Mining, Chemometrics, Signal Processing and Psychology. A concise review of tensor decompositions in the literature can be found in [5]. In this particular work, we focus on the so called CP/PARAFAC decomposition, however, [5] provides an overview of the entire variety of decompositions that have been introduced.

In the immediate field of interest, anomaly detection, there has been a fair amount of tensor applications. In particular, [2] develop a decomposition model that is suitable for stream data mining and anomaly detection. The authors of [6] introduce a scalable anomaly detection framework using tensors. In [12] the authors perform anomaly detection in a (source IP, destination IP, port number) dataset, and in [7] the authors operate on (source IP, destination IP, port number, timestamp) dataset. Finally, in [3] the authors propose a framework for anomaly detection in multi-aspect data that is based in tensor decompositions.

In terms of scalable tensor decompositions, [12] proposes a fast, sparse and approximate method that scales very well mostly in multicore environments. In [13], the authors propose a MapReduce, scalable and distributed algorithm for CP/PARAFAC; this suits better our purpose, since our data resides in a distributed file system.

3 MalSpot

In this section, we describe the MalSpot, an approach for finding the pattern in huge data from scalable design. The MalSpot has two modes, i.e., single-resolution mode and multi-resolution mode based on multi-linear analysis process. The notations are shown in Table 2.

Initially, given the data description, provided a few lines above, we are able to form three mode tensors, whose non-zero entries correspond to the non-zero entries of the network logs. Since the logs record counts of events, and due to high data skew, it is often the case that a few set of connections will outnumber the rest of the connections, in terms of counts. To that end, we have two choices, in order to alleviate this issue: We may, either, make our data binary, where the tensor, we may take the logarithm of the counts, so that we compress very big values.

Tensor formulation of our problem

In order to form a tensor out of the data that we posses, we create a tensor entry for each (i, j, k) triple of, say (source IP, target IP, timestamp) that exists in our data log. The choice for the value for each (i, j, k) varies: we can have the raw counts of connections, we can compress that value (by taking its logarithm), or we can simply indicate that such a triplet exists in our log, by setting that value to 1.

Tensor decomposition leverages multi-linear algebra in order to analyze such high-order data. The canonical polyadic (CP) or PARAFAC decomposition we

Table 2. Table of Symbols

Notations	Definitions and Descriptions
D	raw data from different types of information security logs with three different kinds of features
x_1,x_2,x_3	the three features defined in data D
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	tensor factor matrices, associated with x_1, x_2 and x_3
$\underline{\mathbf{X}}$	the 3-mode tensor
$\mathbf{C}$	covariance matrix for measuring the prioritizing of investigation from clusters
R	the rank for tensor decomposition
k	the number of clusters given for the malicious patterns clustering
λ	the threshold of the top-n selected result
k-means($\mathbf{M}$,k)	scalable k-means algorithms given matrix $\mathbf{M}$
GigaTensor(χ)	scalable tensor decomposition [13]
$\oplus$	string concatenation
$Cov(G^+, G^*)$	covariance measures between clustering groups G^+ and G^*

employ can be seen as a generalization of the Singular Value Decomposition (SVD) for matrices. CP/PARAFAC decomposes a tensor to the weighted sum of outer products of mode-specific vectors for a 3-order tensor. Formally, for an M-mode tensor $\underline{\mathbf{X}}$ of size $\{I_1 \times I_2 \times \cdots \times I_M\}$, its CP/PARAFAC decomposition of rank R yields $\underline{\mathbf{X}} \approx \sum_{r=1}^{R} \lambda(\mathbf{a}_r^{(1)} \circ ... \circ \mathbf{a}_r^{(M)}) = \sum_{r=1}^{R} \prod_{m=1}^{M} \mathbf{a}_r^{(m)}$ where $\circ$ denotes the outer product, and $\prod$ is in the sense of vector outer product multiplication (and not in the traditional multiplication operation).

3.1 Network Malicious Behavior Decomposition

Given the data description, provided a few lines above, we are able to form three mode tensors, whose non-zero entries correspond to the non-zero entries of the network logs. Since the logs record counts of events, and due to high data skew, it is often the case that a few set of connections will outnumber the rest of the connections, in terms of counts. To that end, we have two choices, in order to alleviate this issue: We may, either, make our data binary, where the tensor, instead of counts, stores 1 or 0, depending on whether a specific triplet exists in the logs, or, we may take the logarithm of the counts, so that we compress very big values.

Introduction to tensors

We start by introducing a few definitions. A tensor is essentially a multi-dimensional extension of a matrix; more precisely, a n-mode tensor is a structure that is indexed by n indices. For instance, a 1-mode tensor is a vector, a 2-mode tensor is a matrix, and a 3-mode tensor is a cubic structure. In this work, we focus on three-mode tensors, however, given the appropriate data, we can readily extend our techniques to higher modes.

Tensor formulation of our problem

In order to form a tensor out of the data that we posses, we create a tensor entry for each (i, j, k) triple of, say (source IP, target IP, timestamp) that exists in our data log. The choice for the value for each (i, j, k) varies: we can have the raw counts of connections, we can compress that value (by taking its logarithm), or we can simply indicate that such a triplet exists in our log, by setting that value to 1.

The $\underline{\mathbf{X}}$ is no longer approximate if R is equal to rank($\underline{\mathbf{X}}$), however, we often want to decompose $\underline{\mathbf{X}}$ to $R \ll$ rank($\underline{\mathbf{X}}$), so that we force similar patterns to map to the same low rank basis. Forcing R to be small is *key* to our application, because in this way, we force connections behaving similarly to map to the same low rank subspace.

3.2 Single Resolution Mode in MalSpot

We obtain suspicious patterns via scalable tensor decomposition, as mentioned in previously. In order to find out the groups of similar patterns, we use a scalable k-means implementation, in MapReduce, so that we cluster different malicious patterns, produced by the tensor decomposition. For the cluster, we choose to use the cosine similarity (or rather its inverse) as a distance measure. The cosine distance we used in this study is shown as $similarity(\mathbf{p}, \mathbf{q}) = cos(\theta) = \frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{p}\| \|\mathbf{q}\|}$ where $\mathbf{p}$ and $\mathbf{q}$ are pairs of columns of the factor matrices $\mathbf{A}$, $\mathbf{B}$ or $\mathbf{C}$, produced by the decomposition.

Algorithm 1: MalSpot algorithm (single-resolution mode)

Input: Dataset D with x_1,x_2 and x_3, with size of l, m and n, and a decomposition rank R, clustering number k

Output: Prioritized malicious patterns groups G

`/* step 1: Tensor Construction */`

MapReduce:

key $\leftarrow x_1 \oplus x_2 \oplus x_3$

Map $\langle$ *key* , 1 $\rangle \leftarrow D$

Tensor $\underline{\mathbf{X}} \leftarrow$ Reduce $\langle$ *key* , *count* $\rangle$, i=1 to $|D|$

`/* step 2: Decomposition */`

$\langle \mathbf{A}_{[l \times R]}, \mathbf{B}_{[m \times R]}, \mathbf{C}_{[n \times R]} \rangle \leftarrow GigaTensor(\underline{\mathbf{X}})$

`/* step 3: Clustering and Ranking */`

$\mathbf{a}_i \in$ columns of $\mathbf{A}, i = 1...R \leftarrow$ k-means$(\mathbf{A}, k)$

$\mathbf{b}_i \in$ columns of $\mathbf{B}, i = 1...R \leftarrow$ k-means$(\mathbf{B}, k)$

$\mathbf{c}_i \in$ columns of $\mathbf{C}, i = 1...R \leftarrow$ k-means$(\mathbf{C}, k)$

$\{G_A \mid \{G_A^1,...,G_A^k\}\} \leftarrow$ PrioritizeCov($\mathbf{a}$,λ)

$\{G_B \mid \{G_B^1,...,G_B^k\}\} \leftarrow$ PrioritizeCov($\mathbf{b}$,λ)

$\{G_C \mid \{G_C^1,...,G_C^k\}\} \leftarrow$ PrioritizeCov($\mathbf{c}$,λ)

After clustering, we obtain different groups of connections, as summarized by decomposing $\underline{\mathbf{X}}$. Prioritization of each group is helpful for recommending groups of anomalous connections to domain experts, for further inspection. The covariance distance between two clusters G^+ and G^* as $Cov(G^+, G^*) = \frac{\sum (G_i^+ - \overline{G^+})(G_i^* - \overline{G^*})}{n-1}$.We use the covariance matrix $\mathbf{C}_{[k \times k]}$ to store all-pairs of clusters ; we then rank the groups according to $\sum_{i=1}^{k} \mathbf{C}(:, i)$ and choose the top-k^* to show to a domain expert, for further inspection.

3.3 Multi Resolution Mode in Malspot

As opposed to the single-resolution mode MalSpot, the multi-resolution version takes significant changes among different temporal granularities into account, in order to identify pieces of the data that bear the *uncorrelated levels* characteristic. In order to introduce the multi-resolution mode of MalSpot, let $T_1, T_2, \ldots, T_n$ be the different time granularities (i.e., hourly, daily, weekly and so on). For each T_i, there are k clusters denoted by G_i^j ($j = 1$ to k). In order to identify uncorrelated levels among these different temporal levels of resolution, we use the adjacency matreix of G_i, which we denote by A_i. A detailed outline of the procedure we follow is shown in Algorithm 2.

Algorithm 2: MalSpot algorithm - (multi-resolution mode)

Input: For each time scale T_i we have a set of k clusters $G_i = \{G_i^j \mid j = 1 \text{ to } k\}$ where $\bigcup_{j=1}^{k} G_i^j = D$ and $G_i^m \bigcap G_i^n = \emptyset \, \forall m \neq n$

Output: data $\mathbf{N}_{i,i+1}$ that change clusters from T_i to T_{i+1} and $I_{i,i+1}(a)$ the degree of similarity between $F_i(a)$ and $F_{i+1}(a)$.

`/* step 1: Generating Adjacency matrices` $\mathbf{A}_i = \{a_{mn}\}$ `of` $\mathbf{G}_i$ `*/`

1 Calculate a_{mn} according to (1)

`/* step 2: Generating reduced adjacency matrices` $\mathbf{A}_i^{\text{reduce}} = \{a'_{mn}\}$ `*/`

2 Calculate a'_{mn} according to (2)

`/* step 3: Find data` $\mathbf{N}_{i,i+1}$ `that change clusters from` T_i `to` T_{i+1} `*/`

3 $\mathbf{A}_i^{\text{diff}} = \mathbf{A}_{i+1}^{\text{reduce}} - \mathbf{A}_i^{\text{reduce}}$

4 Find the minimum number of lines $\mathbf{L} = \{(p, \text{l})\}$ where l represents row number or column number that cross the nonzero rows or columns of $\mathbf{A}_i^{\text{diff}}$, p represents nonzero row or column entry of I.

5 $\mathbf{N}_{i,i+1} \leftarrow \{p | (p, \text{l}) \in \mathbf{L} \text{ for some l}\}$

`/* step 4: Calculate the probability of migration of each datum */`

6 Calculate $I_{i,i+1}(a) \, \forall i$ and $\forall a$ according to (4)

In Algorithm 2, we have a set of k clusters $G_i = \{G_i^j \mid j = 1 \text{ to } k\}$ where $\bigcup_{j=1}^{k} G_i^j = D$, which are generated iteratively from different temporal resolutions. In the first step, we use the adjacency matrices $\mathbf{A}_i = \{a_{mn}\}$ in order to record whether a member of a cluster *changes* its cluster assignment between different resolutions, as shown in Eq. 1

$$a_{mn} \leftarrow \begin{cases} 1, & \begin{cases} \text{a.} & \text{if } m < n \text{ and data } m \text{ and } n \text{ are in the same cluster} \\ \text{b.} & \text{if } m = n \text{ and data } m \text{ itself forms a cluster} \end{cases} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In order to reduce the computational complexity, we summarize the the reduced adjacency matrices as $\mathbf{A}_i^{\text{reduce}} = \{a'_{mn}\}$, by employing the characteristics of transitivity among clusters as shown in Eq. 2.

$$a'_{mn} \leftarrow \begin{cases} 1, & \text{if } a_{mn} = 1 \text{ and } a_{pn} = 0 \quad \forall p > m \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Let F_i be a function that maps the data in time scale T_i to its cluster. That is, $F_i : D \mapsto G_i$ s.t. $F_i(a) = G_i^j$ iff $a \in G_i^j$. For any $a \in D$, denote $C_{i \to i+1}(a) = F_i(a) \bigcap N_{i,i+1}$ (C denotes *change*), $C_{i \leftarrow i+1}(a) = F_{i+1}(a) \bigcap N_{i,i+1}$, $R_{i \to i+1}(a) = F_i(a) \backslash N_{i,i+1} = F_i(a) - O_{i \to i+1}(a)$ (R denotes that the assignment remain the same), $R_{i \leftarrow i+1}(a) = F_{i+1}(a) \backslash N_{i,i+1}$, the conditions considering change or no change are shown in Eq. 3.

$$S_{i,i+1}(a) \leftarrow \begin{cases} C_{i \to i+1}(a) \bigcap C_{i \leftarrow i+1}(a), & \text{if } a \in N_{i,i+1} \\ R_{i \to i+1}(a) \bigcap R_{i \leftarrow i+1}(a), & \text{if } a \notin N_{i,i+1} \end{cases} \tag{3}$$

$I_{i,i+1}(a)$ is the degree of similarity between $F_i(a)$ and $F_{i+1}(a)$ is as shown in Equation (4).

$$I_{i,i+1}(a) = \frac{\sum_{a \in S_{i,i+1}(a)} p_{i,i+1}(a)}{\sum_{a \in F_{i+1}(a)} p_{i,i+1}(a)} \tag{4}$$

where the $p_{i,i+1}(a)$ is given as in Eq. 5.

$$p_{i,i+1}(a) \leftarrow |S_{i,i+1}(a)| - 1 \tag{5}$$

4 Experiments

In this section we show the effectiveness of MalSpot in detecting anomalous behavior in diverse settings of information security logs, i.e., Security Operations Center (SOC) event logs and Honeynet firewall logs. We design all the experiments in order to answer the following questions:

- **Q1: Malicious pattern discovery:** how can MalSpot effectively identify malicious events in a variety of sites or for a long term. Especially, how effective is MalSpot in detecting random scanning, and hit-list scan behavior?
- **Q2: Providing insights to domain experts:** what is an intuitive and concise way of presenting the detected anomalies to domain experts and network administrators, so that they can, in turn, validate our methodology, as well as further investigate a set of attacks.

4.1 Dataset and Environment

As we mention in the problem definition, we analyze data coming from three different sources. We use three different types and sources of datasets in our experiments to demonstrate MalSpot performances. These two datasets are 1) a Honeynet dataset, 2) an intrusion detection system (IDS) events dataset from Taiwan government; we summarize the datasets in Table 3. The Honeynet dataset was collected especially for the purposes of this work, using the Honeynet project system from 10 distributed sites in Taiwan[3]. The IDS events dataset is collected from the Security Operation Center of the Taiwanese government, for the entire year of 2012. This dataset includes the IDS triggered alerts from 61 government institutes with 4,081 types of events. These types of events can be categorized into 39 classes, Of which 33 classes of attack is defined by Snort IDS, the other 6 classes are custom by us (including: blacklist, high threat malware behavior and so on). In total, the SOC dataset contains 828,069,066 events, the dataset details could be shown as Table 3. .

Table 3. Datasets harvested & analyzed

Dataset	Description	Dimension	Nonzeros
Honeynet	Gathered from 10 distributed Honeynet sensors in Jan. 2013	368K x 64K x 31	3243K
G-SOC (Type)	Taiwan official institutes IDS events in 2012	[8187, 361, 52, 12] x 4081 x 61	800M+
G-SOC(Cat)	Taiwan official institutes IDS events in 2012	[8187, 361, 52, 12] x 39 x 61	1742K

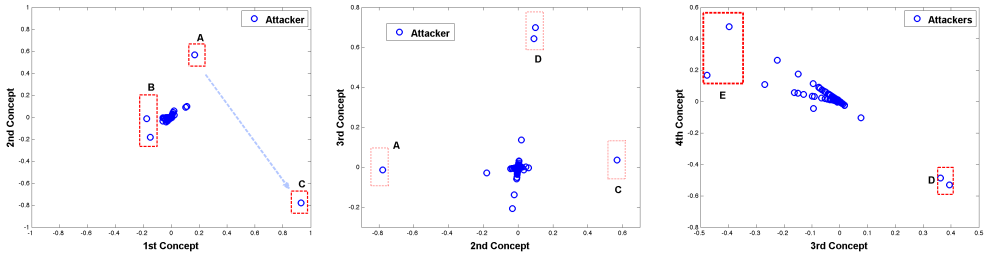
We leveraged the scalability of GigaTensor[13] in 16 nodes of a Hadoop cluster where each machine has 2 quad-core Intel 2.83 GHz CPU, 8 GB RAM, and 2 Terabytes disk. The Apache Mahout(Scalable Machine Learning and Data Mining)[4] version 0.7 is used for supporting clustering algorithms.

4.2 Analysis of the Honeynet data

For this dataset, we set $R = 5$ as the low rank of the tensor decomposition; after decomposing the tensor, we obtain three factor matrices each representing one of the three modes of our data: source IP, target IP and timestamp respectively. Each column of those factor matrices corresponds to one out of the R latent groups, in our low rank representation of the data. Based on this low rank embedding of the data, we compare pairs of columns for each factor matrix, in order to detect outliers. For example, given the factor matrix $\mathbf{A}$ that corresponds to the source IP, if we plot, say, columns 1 and 2 against each other, we will see

[3] Honeynet project, http://www.Honeynet.org/
[4] Apache Mahout, http://mahout.apache.org/

a scatterplot that contains one point for each source IP; given this scatterplot, we are able to detect the outliers. We henceforth refer to the 'score' for each source IP (or any other entity associated to a particular mode of a tensor), as expressed by the values of the columns of the corresponding factor matrix, as TensorScore. We show our most outstanding results in Figure 1.



(a) (b) (c)

Fig. 1. Scatter plot in Honeynet result from both attackers and victims views. (a) In 1^{st}-2^{nd} concept of source IP view, we observe three different cases, case A is POP3 (port 110) brute force attacks, and case C is port scanning in port 25. Case B contains a lot of instances but cannot be separated in this plot ; (b) In $2^{nd}3^{rd}$ concept sof source IP view, case C and D appear in this plot which is medium scale of scanning behavior, using ports 22, 23, 135 and 445; (c) In 4^{th}-5^{th} concepts of source IP view, case E appears which represents another scanning behavior.

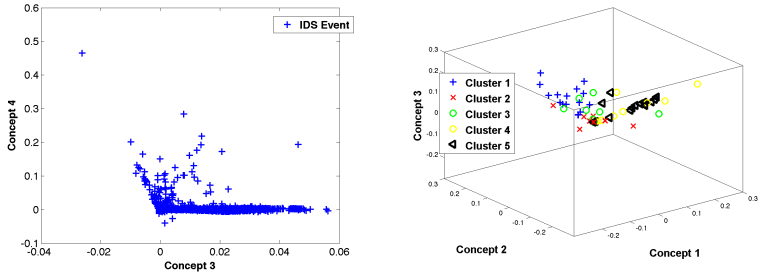
According to the scatterplots obtained from tensor analysis, in Figure 1(a)-(c), we may observe the relative attackers' relations according to different directions of the TensorScore. In Figure 1(a), we found two outliers (denoted as A and C) out of three clusters. After further inspection of the participating of the attacks, we found out that the attackers in group A are focused on port port 110 and perform POP3 probing. The attackers in group C attempted to use ports 50 79 aiming to perform a large scale port scan. Both attack groups (A and C) appear only on a single day (January 10th and January 25th respectively). In Figure 1(b) and (c), we are able to discover a new set of anomalies, as we choose a different couple of latent factor vectors to obtain the TensorScore from. In attack group D, we were able to identify an attacker who attempted to trigger 14,652 connections to 236 target Honeynet system IP addresses, with a duration of 8 days. We present the attacks that belong to group E; those attacks focus on a particular Microsoft Network security vulnerability that is associated with ports 139 and 445.

4.3 IDS Event Result

The IDS events consist of an entire year's worth of data, collected by the G-SOC of Taiwan in 2012. The single resolution mode of MalSpot use the day scale

granularity to analyze these logs. In Fig. 2(a), we illustrate the two groups that MalSpot was able to spot in the IDS event logs. The first group is associated with the Web and native IDS event rules, whereas the second group is related to the blacklist-based event rules.

We proceed to the second step of our analysis, by setting $k = 5$ and cluster the tensor decomposition latent groups as shown in Fig. 2(b). This post-analysis, forces hosts with similar characteristics to end up in the same cluster. For instance, cluster 1 contains a vast number of hosts that are related to a large scale government institute. In clusters 4 and 5, we mostly observe service-oriented information systems tend.



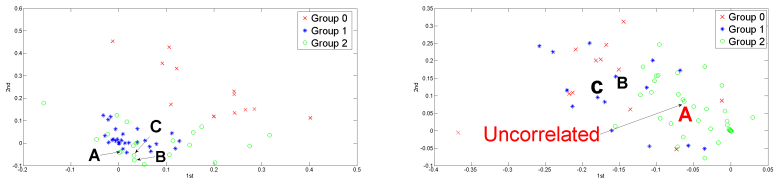
(a) (b)

Fig. 2. IDS alert events scatterplot: (a) In 3rd-4th concept of event view (IDS alerts), we observe three different cases, part of alerts are triggered more often and part of them triggered rarely; (b) from the organization's view, we can see 5 groups are clustered together.

In order to evaluate the multi-resolution nature of MalSpot, we select 4 different time resolutions, i.e., hour, day, week and month, and seek to identify the uncorrelated levels. The result is shown as Fig. 4. In fact, hosts A, B and C are grouped as the same cluster in both hourly and daily levels. MalSpot is able to select the uncorrelated levels for B and C in weekly and monthly granularities, respectively.

In Fig. 3, we use event classes with 4 different temporal resolutions to identify uncorrelated levels of activity, for various institutes. We identify that institute A has an uncorrelated level of activity between weekly and daily granularity, as opposed to institutes B and C. From further investigation, institute A has suffered from a "system-call detect" event class during the uncorrelated time period (e.g., B and C have a uniform distribution of activity during the entire year, but A is skewed towards early 2012). Additionally, MalSpot offers huge savings in computational time in order to detect the aforementioned attacks, when compared to competing methods.

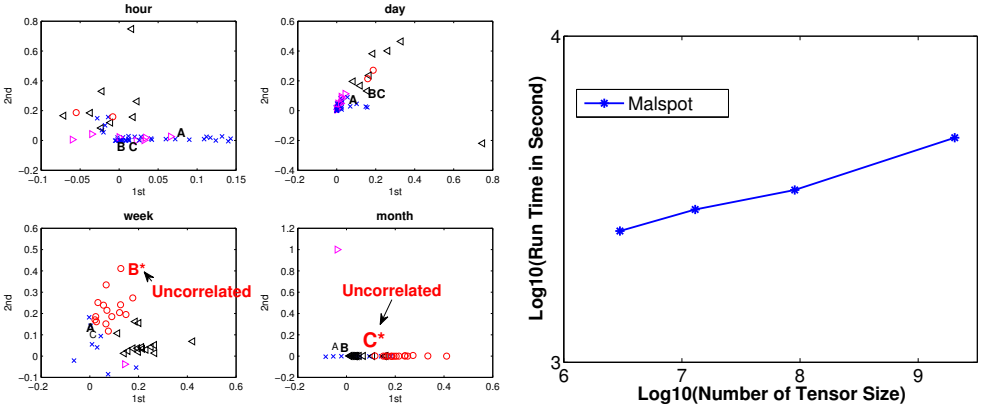
Based on the Fig. 4 (a), we plot the scatter-plots from the hour and week (X-axis) versus the triggered event types (Y-axis). We observed a critical difference



(a) (b)

Fig. 3. Scatter plot of different time resolutions of G-SOC(cat) dataset. Each point in the scatter-plot denotes an institution. (a) day resolution and (b) week resolution.

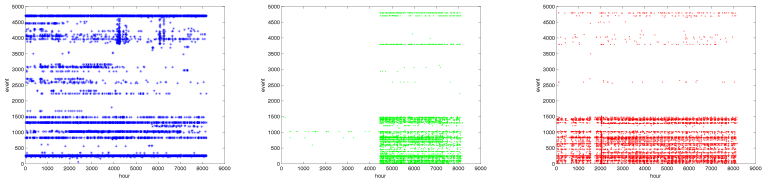
between institute A and institutes B, C with respect to attack periodicity. Institutes B and C suffered so-called "WEB-MISC TOP10.dll access" attacks [5] while A did not. Fig. 5 shows detail time-event scatter plots. Therefore, MalSpot identifies potential malicious behavior between hosts A and B, C employing the notion of uncorrelated levels. Host A suffered periodic attacks targeted on the Windows OS (Windows ANI File Parsing Buffer Overflow (MS05-002)), whereas the periodic attacks of host B and C were concentrated on a malicious relay station.



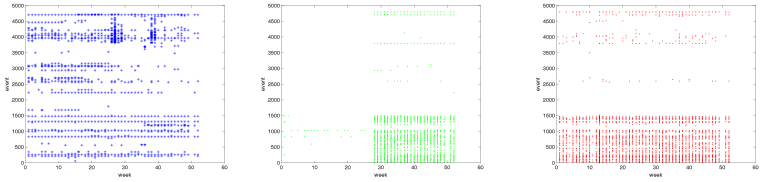
(a) (b)

Fig. 4. (a) Scatter plot of different time resolutions. Each point in scatterplot denotes an institution. (b) The scalability of MalSpot, as the input size grows.

[5] This event is generated when an attempt is made to exploit a buffer overflow in the Trend Micro InterScan eManager.



(a) Host A (hour) Host B (hour) Host C (hour)



(a) Host A (week) Host B (week) Host C (week)

Fig. 5. Scatter plot of different time resolutions. (X-axis is time scale and Y-axis is event types). We can see the difference in the distribution of scatterplots.

5 Conclusion

In this work, we develop a big data analytics system in order to discover malicious patterns in a variety of network/malware propagation settings, so that we can further use them for attack detection and prevention, when attacks are concurrent with legitimate traffic. By conducting large-scale information security data analysis, our proposed method, MalSpot, can easily identify the patterns in massive IDS logs, spamming delivery logs, and Honeynet firewall logs, pertaining to long-term and stealthy attack behavior.

The contributions of this work are the following:

- **Design** of the MalSpot system: based on `hadoop`, it can scale up to arbitrary-size datasets, and it is nearly linear as the log trace size grows.
- **Discoveries:** We report very interesting attack patterns, and positively identified attacks, as detected by MalSpot.
- **Scalability:** regardless of data scale or data source variety, MalSpot is able to detect attacks efficiently and effectively.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grants No. IIS-1247489 and CNS-1314632 Research was sponsored by the Defense Threat Reduction Agency and was accomplished under contract No. HDTRA1-10-1-0120. Also, sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. This work is also partially supported by a Google Focused Research Award. Any opinions, findings, and conclusions or recommendations expressed in this

material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

1. Dainotti, A.: Analysis of a "/0" stealth scan from a botnet. In: IMC'12. (2012)
2. J. Sun, S.P., Yu, P.S.: Window-based tensor analysis on high-dimensional and multi-aspect streams. In: In ICDM. (2006) 1076–1080
3. K. Maruhashi, F.G., Faloutsos, C.: Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In: ASONAM'11. (2011)
4. Kishino, Y., Sakurai, Y., Yanagisawa, Y., Suyama, T., Naya, F.: Svd-based hierarchical data gathering for environmental monitoring svd-based hierarchical data gathering for environmental monitoring. In: Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication. UbiComp '13 Adjunct, New York, NY, USA, ACM (2013) 9–12
5. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review **51**(3) (2009) 455–500
6. Kolda, T., Sun, J.: Scalable tensor decompositions for multi-aspect data mining. In: ICDM. (2008)
7. Koutra, D., Papalexakis, E.E., Faloutsos, C.: Tensorsplat: Spotting latent anomalies in time. In: Informatics (PCI), 2012 16th Panhellenic Conference on, IEEE (2012) 144–149
8. Li-Ming Chen, Meng-Chang Chen, W.L., Sun, Y.S.: A scalable network forensics mechanism for stealthy self-propagating attacks. Computer Communications (2013)
9. Lin, J., Vlachos, M., Keogh, E., Gunopulos, D.: Iterative incremental clustering of time series. In: IN EDBT. (2004) 106–122
10. Matsubara, Y., Sakurai, Y., Faloutsos, C., Iwata, T., Yoshikawa, M.: Fast mining and forecasting of complex time-stamped events. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '12, New York, NY, USA, ACM (Aug 2012) 271–279
11. Papadimitriou, S., Yu, P.: Optimal multi-scale patterns in time series streams. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, ACM (2006) 647–658
12. Papalexakis, E.E., Faloutsos, C., Sidiropoulos, N.D.: Parcube: sparse parallelizable tensor decompositions. In: Machine Learning and Knowledge Discovery in Databases. Springer (2012) 521–536
13. U. Kang, E. Papalexakis, A.H., Faloutsos, C.: Gigatensor: scaling tensor analysis up by 100 times - algorithms and discoveries. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '12, New York, NY, USA, ACM (2012) 316–324