Graph-based malicious login events investigation

Faouzi Amrouche, Sofiane Lagraa, Georgios Kaiafas, Radu State
SnT, University of Luxembourg
firstname.lastname@uni.lu

***Abstract*—A large body of research has been accomplished on detecting malicious events, attacks, threats, or botnets. Different techniques and approaches have been proposed to detect them such as machine-learning-based, or rule-based. However, there is a lack of sophisticated techniques for investigating malicious events and understanding the root cause of attacks. In this paper, we propose a knowledge discovery approach for investigating and visualizing malicious authentication events. The approach is based on data mining techniques on attacks in order to extract the behavior of malicious authentication. We also propose a novel graph-based representation method that helps highlight attack scenarios. The evaluation is performed on a publicly available large dataset, where we analyze behavior of malicious authentication events. The results are useful for security experts in order to improve the existing solutions by making them robust.**

I. Introduction

In cybersecurity, there are three concepts to strengthen security in a domain environment: prevention, detection and identification, and investigation. In prevention, the goal is to reduce attack surface by discovering vulnerable nodes in the network. In detection and identification, the goal is to study logon patterns, detect anomalies, threats and reconnaissance. In investigation, the goal is to discover the attack path and find additional compromised machines or users.

Most of the existing works focus only on the prevention, detection and identification. The works can be based on rule-based (or signature based) detection [1], [4] or machine learning-based techniques [5], [11]. However, the techniques and methods for investigating and understanding attacks are not well developed due to the following : first, there is a lack of new data in terms of quantity and quality. Several benchmarks used in research are issued from simulations and thus lack variety of attacks. The quantity of data available does not reflect the reality compared to real life, where gigabytes of logs are produced each day [12]. Second, there is a lack of redteam data highlighting attacks. Without attacks, the researcher cannot provide tools for investigation to discover to the source of an attack.

Investigation is a challenging task, because it's about understanding and interpreting attacks which are not well developed.

The investigation allows to upskill the security experts, implement more sophisticated defense tools based on rules or artificial intelligence, and extract insights for improving the existent tools.

The objective of this paper is to propose a method to investigate attacks by discovering their root cause and extract knowledge from them.

978-3-903176-15-7

Problem Statement. Given a set of authentication event logs, including attacks, the problem is to analysis each event of an attack scenario and extract a behavioral model for it.

Our solution is based on empirical studies and graph-based approach. The general idea is to use malicious events logs from the redteam dataset. We recursively project the source user and computer used for attacks into the global authentication log related to source user and source computer. This process partitions the event logs to be analyzed, and confines each partition being conducted to the corresponding smaller projected dataset. In each projected event logs of attacks, the events related to a source user of malicious events are analyzed and modeled into a *behavior graph*. Behavior graph highlights the different steps leading to malicious events. Graph-based techniques and visualization are applied on a behavior graph for extracting knowledge behind an attack. For experiments, we use authentication dataset of Los Alamos National Laboratory (LANL). LANL published multiple log datasets which is also known as Sandia Dataset [8]. This dataset includes separate files for authentication, process logs on various computers, DNS, netflow logs along with validated anomalies detected by their red Team [7].

The results show that the behavior graphs of each user help the security experts for investigation by going back to the source of an attack and its history. The graph-based model reduces the redundant events and makes the user's behavior visualization easier in a compact way.

In this paper, our contributions are as follows:

- We profile authentication and malicious dataset by extracting features.
- We develop a graph-based model of user's behavior of authentications.
- We analyze and visualize behavior graphs using graph techniques for investigations and root cause of attacks.

II. Dataset description

This work uses a public comprehensive dataset provided by the Los Alamos National Laboratory [8], [7]. Its content was collected over a period of 58 consecutive days and is comprised of 1.05 billion authentication events (total uncompressed size of 70GB) from multiple sources, such as individual computers, servers, and Active Directory servers running the Microsoft Windows operating system. It is publicly available at *https://csr.lanl.gov/data/cyber1/*. In the rest of this paper we will refer to an *authentication event* as an *event*. We formally define them as:

Definition 1 (Authentication event): An *authentication event* e is defined as a vector $e = < t, su, du, sc, dc, at, lt, ao, sf >$, where it represents the time, source user, destination user, source computer, destination computer, authentication type, logon type, authentication orientation, success/failure, respectively.

The *authentication event logs* $AUTH = \{e_1, ..., e_n\}$ is the ordered set of authentication events. Table VI shows an example of authentication event logs.

t	SU	DU	SC	DC	AT	LT	AO	SF
145015	U1723@C1759	U1723@C1759	C17693	C1759	NTLM	Network	LogOn	Fail
150885	U620@DOM1	U620@DOM1	C17693	C1003	NTLM	Network	LogOn	Success

TABLE I: Example of authentication event logs

Definition 2 (Malicious event): An *Malicious event e_malicious* is defined as a vector $e\_malicious = < t', su', sc', dc' >$, where it represents the time, source user, source computer, destination computer.
The *malicious event logs* $MAL = \{e\_malicious_1, ..., e\_malicious_m\}$ is the ordered set of authentication events. Table II shows an example of malicious event logs.

t	SU	SC	DC
150885	U620@DOM1	C17693	C1003
830548	U1653@DOM1	C22409	C754

TABLE II: Example of malicious event logs

Definition 3 (Malicious User): A user $u_i \in U$ is called a *Malicious User* if there is at least 1 Malicious event $e\_malicious_i$ in which $SrcUser_{e\_malicious} = u_i$.

III. Attack mining approach

Our goal is to track and profile users involved in malicious events. In addition, we want to establish causal relationships among authentication events to build behavior users activities. We propose a graph-based modelling approach for tracking the behavior of a user through time. Different steps are needed for transforming events into graph describing the states of a user's session in the system and their relationships during the authentications. User behavior has been widely used in the detection of malicious website visitors [13], P2P IPTV services [2], security analysis [3]. In our context, we are interested in the tracking of authentication process to successive events. Relying on a graph modeling will allow to reduce duplicated events and thus to model global authentication process.

The investigation process is detailed as follows.

1) Finding a set of event logs in $AUTH$ containing only malicious events. The only event logs containing malicious source users are selected for analysis, by scanning authentication event logs once. Thus, we select a set of event logs for each source user.
2) For each source user, we construct a sequence of events based on the timestamp to model the full behavior of a user.
3) For each source user, we construct a graph model. This step allows us to construct a behavior graph from each sequence. It allows to reduce the behavioral representation from a long sequence to a graph that aggregates redundant events while still keeping the order of events. In addition, the graph-based modelling facilitates the visualization, investigation, analysis, and scalability.
4) Behavior graph metric is applied in order to find the root cause of a malicious event. We extract all paths of previous events before a malicious event.

A. Filtering authentication events

Given authentication event logs $AUTH$ and malicious event logs MAL. The goal of this step is to filter the $AUTH$ according to users in MAL. Having malicious event logs, the authentication event logs can be filtered to focus on events having malicious source users used for attacks. The output of this filtering step is a sequence of authentication event logs related to each source user in malicious event logs. We define $Trace_{SU_i}$ as set of authentication events highlighting the traces of a source user SU_i: $Trace_{SU_i} = \{e | e \in AUTH \wedge e\_malicious \in MAL \wedge e.SU = e\_malicious.SU\}$. $Trace_{SU_i}$ is sorted on increasing time in order of events, as well as in $AUTH$. $Trace_{SU_i}$ describing the history of events related to SU_i.

B. Authentication events transformation and representation

After the filtering of events, dependencies among events are built based on users and time to identify successive events. For transforming the set of authentication events into sequences of events, the events are grouped based on users.

Definition 4 (Authentication events sequence): Let $Trace_{SU_i}$ be the set of all events of SU_i. We denote $S_{SU_i}(Trace_{SU_i}, T_{start}, T_{end})$, the sequence of SU_i between the starting time T_{start} and the ending time T_{end} where $T_{start} < T_{end}$. S_{SU_i} is thus a list of authentication events ordered by time: $S_{SU_i} =< (e_1, e_1.t), \ldots, (e_n.t), \ldots >$, where $e_i \in Trace_{SU_i}$, $T_{start} \geq e_i.t \leq T_{end}$ such that $e_{i+1}.t > e_i.t$.

Figure 1 shows an example of an authentication events sequence. All events with same source user are grouped into a single sequence.

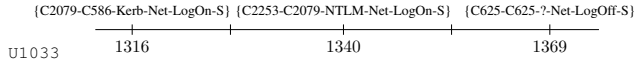


Fig. 1: Authentication events sequence

This sequence representation allows to characterize the total order of events of a specific user. $S_{SU} = \{S_{SU_1}, S_{SU_2}, ..., S_{SU_l}\}$, where l is the number of source users used in MAL.

C. Constructing a behavior graph

In practice, the length of sequences are very long and can reach up to a million of events [10], [9]. In our case, the length of authentication event sequences can be very long due to the redundancy in a sequence in most of the time. To characterize causal relations between two successive events in a authentication events sequence, we introduce the notion of the *behavior graph* model as a graph representation for

successive events in a user sequence. A behavior graph is the aggregation of successive events from a single sequence (identified by a unique user) into a single representation. It is a directed graph that represents successive relationships between events of an event sequence. Thus, a cycle (or a loop) in the graph model indicates successive repetitive events.

Formally, a behavior graph is defined by the following definition.

Definition 5 (Behavior graph): Behavior graph of a user SU_i is a labelled directed graph $G_{SU} = (V, E, \beta)$:

- $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices, where $v_i = (du, sc, dc, at, lt, ao, sf)$. Specifically, each vertex v_i represents event attributes except the time and source user. A vertex is identified by the concatenation of event attributes.
- E is a set of edges in G_{SU_i}. Let u and v be two vertices in V. There is an edge $(u, v) \in E$ if and only if there exists a dependency $u \xrightarrow{f_{u,v}} v$ in S_{SU_i}. Each edge (u, v) indicates that the event v occurs after the event u.
- β is a function that assigns for each edge (u, v) the number of dependency occurrence $f_{u,v}$.

Figure 3 shows a behavior graph constructed from the events sequence in Figure 2.

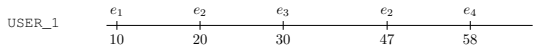


Fig. 2: Events sequence

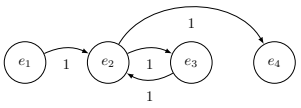


Fig. 3: Behavior graph

The advantage of such an approach is to reduce the number of events into graph by avoiding redundant events. The represented relations and dependencies between all events represent the signature features of each user, i.e. the behavior of user during the authentication. Behavior graph helps the security experts to track and profile a user and discover knowledge and paths leading to an attack.

D. Behavior graph analysis

In order to ease the analysis of behavior graph by discovering the root cause, and what happened before a malicious event, the goal is to extract all paths in the behavior graph from any sources to target i.e. malicious event. Given a behavior graph, a source vertex $source$ and a destination vertex $target$. The goal is to discover all paths from given $source$ to $target$. A path in a directed graph is a sequence of vertices in which there is a directed edge pointing from each vertex in the sequence to its successor in the sequence. Finding all possible paths is a NP-Hard problem [6], since there are exponential number of simple paths. Thus, we focus only on simple paths referring to paths which contain no repeated vertices and only paths having a fixed and maximum length $length$.

For example, given a graph in Figure 3, and considering that the event e_4 is a malicious event. The list of paths from any sources to e_4 having $length = 3$ are: • $e_1 \rightarrow e_2 \rightarrow e_4$ • $e_2 \rightarrow e_4$ • $e_3 \rightarrow e_2 \rightarrow e_4$

IV. Experimental results

In this section, we present the experimental results of the conducted investigation. First, we present the characteristics of a behavior graph. Then, we do an in-depth analysis.

A. Behavior graph characteristics

The graph representation that we introduced presented important space optimizations. Both in terms of visualization and disk storage. This is illustrated through Table III and Table IV. Table III shows behavior graph characteristics in terms of number of vertices and edges for the malicious community. We notice the existence of graphs with only 2 vertices. The largest graphs can achieve million of edges and thousands of vertices. Through Table IV, we represent the 10 users with

$avg\|V\|$	$avg\|E\|$	$max\|E\|$	$min\|E\|$	$max\|V\|$	$min\|V\|$
657	189871	11186191	2	16535	2

TABLE III: Behavior graph characteristics.

largest graphs in terms of events. Events : Number of events related to each user. $|V|$: Number of vertices, $|E|$: Number of edges, Size : Size of the user's file on disk, GSize : Size of a behavior graph on disk.

Graph	Events	$\|V\|$	$\|E\|$	Size (MB)	GSize (MB)
U66@DOM1	11186192	1124	11186191	688.98	605.79
U13@DOM1	1503716	374	1503715	89.87	78.64
U24@DOM1	1047494	1203	1047493	61,58	53.82
U78@DOM1	1047494	763	1047493	40.69	35.53
U12@DOM1	466915	432	466914	27.54	24.06
U1289@DOM1	321917	375	321916	21.39	18.95
U293@DOM1	243473	1510	243472	14.91	13.09
U679@DOM1	204922	177	204921	12.85	11.31
U453@DOM1	170061	934	170060	10.59	9.36
U86@DOM1	158946	1053	158945	9.46	8.27

TABLE IV: Top-10 of users having a large events.

The experiments resulted in important reduction rates. For the disk space, we get approximately 12.5% decrease in size, which can be much higher in some cases. For for the vertices/events ratio, the rates are impressively high. Some users had reductions by up to 10000. This is mainly due to the redundant characteristic of the attack events (repeated actions), which appears clearly when we ignore the time dimension.

B. Behavior graph analysis

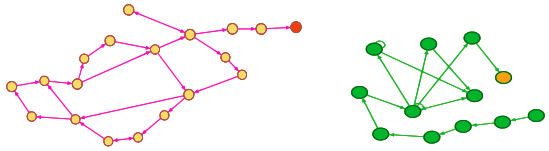
In this part, we will present some behavior analysis that we have done on users graphs. For this purpose, we picked two random users from the malicious dataset and analyzed their graphs. For each user, we extracted two different paths from the graph that lead to an attack event and walked through each one. We fixed the length of paths to 7 to have a right amount of relevant events to analyze, while ignoring the events which are not related to the attack.

Event	Path 1	Path 2
1	C3352-C743-NA-NA-TGS-S	C3352-C1877-NA-NA-TGS-S
2	C2106-C2106-NA-Network-LogOff-S	C3352-C528-NA-NA-TGS-S
3	C3352-C2106-Kerberos-Network-LogOn-S	C3352-C743-NA-NA-TGS-S
4	C2106-C2106-NA-Network-LogOff-S	C2106-C2106-NA-Network-LogOff-S
5	C1618-C1617-NA-NA-TGS-S	C1618-C1617-NA-NA-TGS-S
6	C1618-C1618-NA-NA-TGT-S	C1618-C1618-NA-NA-TGT-S
Attack	C17693-C492-NTLM-Network-LogOn-S	C17693-C492-NTLM-Network-LogOn-S

TABLE V: Paths for user U7394

User U7394 :

Path 1: The user starts by successfully getting a service ticket (TGS) from C743 through the machine C3352. We notice then a cycle (Apparent in graph): LogOff from C2106, LogOn and then LogOff again from the same machine. Then, U7394 changes machines (C1618) to request a Ticket Granting Ticket, and finally achieves the attack through NTLM.



(a) Graph of user U7394 (Attack in red)

(b) Graph of user U5254 (Attack in orange)

Fig. 4: Graph of users

Path 2: Unlike the other path, in this one, the user starts by doing successive SGS Authentications on different machines (from C3352). The user logs Off from C2106, requests TGS and realizes the attack on C492. We notice that there is a similarity of events between path 1 and path 2 differing only in terms of source or destination computer, a feature that can be explored in future works.

Event	Path 1	Path 2
1	C12320-C801-Kerberos-Network-LogOn-S	C1521-C12320-Negotiate-Unlock-LogOn-S
2	C801-C801-NA-Network-LogOff-S	C12320-C12320-NA-NA-AuthMap-S
3	C12320-C801-Kerberos-Network-LogOn-S	C12320-C12320-NA-Unlock-LogOff-S
4	C801-C801-NA-Network-LogOff-S	C801-C801-NA-Network-LogOff-S
5	C801-C801-NA-Network-LogOff-S	C801-C801-NA-Network-LogOff-S
6	C1438-C1438-NA-Network-LogOff-S	C1438-C1438-NA-Network-LogOff-S
Attack	C17693-C1438-NTLM-Network-LogOn-S	C17693-C1438-NTLM-Network-LogOn-S

TABLE VI: Paths for user U5254

User U5254 :

Path 1: We notice that U5254 Logs On and Off repeatedly from the same machine (C12320) to the same machine (C801). That appears on the graph as a cycle, in addition to a loop on the instantly repeated events. The LogOns are done through Kerberos, and all events are of type Network. In addition, the attack occurs through NTLM. We see that the event of attack is rare. It means that the number of occurrences of the event is infrequent (less than 10 occurrences). In addition, the attack is performed from unusual source computer.

Path 2: This path is quite different from the other one. We notice that the user starts by logging-On on C12320 with a Negociate Auth type, different from the previously used (Kerberos). The user does an Authmap authentication on the same machine, and logs off from the machine he logged on. After that, he/she logs off from C801 (the one from path 1), logs off from C1438, and the attack happens.

V. Conclusion and Future Work

In this paper, we propose a new approach for investigating and tracking malicious activities with authentication events monitoring. It relies on a behavioral graph modeling of authentication events. The proposed approach is based on graph modeling and analysis to profile the behavior of malicious authentications. To the best of our knowledge, the approach introduces a novel solution for attack scenarios modeling and investigation. It allows an easy analysis of the attack sequence and facilitates the study of its root causes. In addition, it presents important optimizations in terms of required space and simplified visualization. Our future plan consists on mining behavior graphs by exploiting characteristics like similarities, common behavior of attacks and their frequent or infrequent root cause. In addition, we plan to integrate the time dimension to extract time-related patterns like connection duration and login frequencies.

References

[1] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg. Rule-based anomaly detection on ip flows. In *IEEE INFOCOM 2009*, pages 424–432, 2009.

[2] M. Elhoseny, A. Shehab, and L. Osman. An empirical analysis of user behavior for p2p iptv workloads. In A. E. Hassanien, M. F. Tolba, M. Elhoseny, and M. Mostafa, editors, *The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018)*, pages 252–263, 2018.

[3] M. S. Farash. Security analysis and enhancements of an improved authentication for session initiation protocol with provable security. *Peer-to-Peer Networking and Applications*, 9(1):82–91, Jan 2016.

[4] N. Hubballi and V. Suryanarayanan. Review: False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Comput. Commun.*, 49:1–17, Aug. 2014.

[5] G. Kaiafas, G. Varisteas, S. Lagraa, R. State, C. D. Nguyen, T. Ries, and M. Ourdane. Detecting malicious authentication events trustfully. In *2018 IEEE/IFIP Network Operations and Management Symposium, NOMS*, pages 1–6, 2018.

[6] M. Y. Kao. *Encyclopedia of Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2007.

[7] A. D. Kent. Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory, 2015.

[8] A. D. Kent. Cybersecurity data sources for dynamic network research. In *Dynamic Networks in Cybersecurity*. Imperial College Press, 2015.

[9] S. Lagraa and J. François. Knowledge discovery of port scans from darknet. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 935–940, 2017.

[10] S. Lagraa, J. François, A. Lahmadi, M. Miner, C. A. Hammerschmidt, and R. State. Botgm: Unsupervised graph mining to detect botnets in traffic flows. In *1st Cyber Security in Networking Conference, CSNet 2017*, pages 1–8, 2017.

[11] E. Lopze and K. Sartipi. feature engineering in big data for detection of information system misuse. IBM / ACM, 2018.

[12] J. Navarro, V. Legrand, S. Lagraa, J. François, A. Lahmadi, G. D. Santis, O. Festor, N. Lammari, F. Hamdi, A. Deruyver, Q. Goux, M. Allard, and P. Parrend. Huma: A multi-layer framework for threat analysis in a heterogeneous log environment. In *Foundations and Practice of Security - 10th International Symposium, FPS*, pages 144–159, 2017.

[13] D. Stevanovic, N. Vlajic, and A. An. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Applied Soft Computing*, 13(1):698 – 708, 2013.