International Journal of Information Security (2024) 23:119–140
https://doi.org/10.1007/s10207-023-00742-7

SURVEY

# A review on graph-based approaches for network security monitoring and botnet detection

Sofiane Lagraa[1] · Martin Husák[2] · Hamida Seba[3] · Satyanarayana Vuppala[4] · Radu State[5] · Moussa Ouedraogo[1]

Published online: 30 August 2023


## Abstract

This survey paper provides a comprehensive overview of recent research and development in network security that uses graphs and graph-based data representation and analytics. The paper focuses on the graph-based representation of network traffic records and the application of graph-based analytics in intrusion detection and botnet detection. The paper aims to answer several questions related to graph-based approaches in network security, including the types of graphs used to represent network security data, the approaches used to analyze such graphs, the metrics used for detection and monitoring, and the reproducibility of existing works. The paper presents a survey of graph models used to represent, store, and visualize network security data, a survey of the algorithms and approaches used to analyze such data, and an enumeration of the most important graph features used for network security analytics for monitoring and botnet detection. The paper also discusses the challenges and limitations of using graph-based approaches in network security and identifies potential future research directions. Overall, this survey paper provides a valuable resource for researchers and practitioners in the field of network security who are interested in using graph-based approaches for analyzing and detecting malicious activities in networks.

**Keywords** Graph theory · Machine learning · Network security · Botnet detection · Monitoring · Cybersecurity

## 1 Introduction

Cyberattacks are nowadays sophisticated, complex, and unpredictable, and detecting them is a real challenge due to the massive volume of heterogeneous data that typically needs to be processed to detect an attack. The enduring major threats are botnets and large-scale attacks performed by orchestrated bots; the attacks include network scanning, sending spam, and launching distributed denial-of-service (DDoS) attacks. In recent years, we have observed a steep rise in the number of ransomware attacks, which became a prevalent threat in today's networks. Both distributed botnet activities and ransomware infection hopping from one machine to another can be comprehensively visualized using graphs. A simple question is—if we can use graphs to visualize and understand such phenomena, can they also be used to detect them?

In cybersecurity, there are three globally accepted tasks for network security: prevention, detection, and investigation. The objective of prevention is to prevent and reduce the attack surface by discovering vulnerable nodes in the network [12, 40, 49, 50]. The objective of detection is to analyze network data or system logs to detect malicious activities

✉ Sofiane Lagraa
sofiane.lagraa@fujitsu.com

Martin Husák
husakm@ics.muni.cz

Hamida Seba
hamida.seba@univ-lyon1.fr

Satyanarayana Vuppala
satyanarayana.vuppala@citi.com

Radu State
radu.state@uni.lu

Moussa Ouedraogo
moussa.ouedraogo@fujitsu.com

1 Fujitsu Luxembourg, Capellen, Luxembourg

2 Institute of Computer Science, Masaryk University, Brno, Czech Republic

3 Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, 69622 Villeurbanne, France

4 Citibank, Dublin, Ireland

5 SnT, University of Luxembourg, Esch-sur-Alzette, Luxembourg

and anomalies and raise alerts using intrusion detection tools such as Snort [16] and Zeek (formerly Bro) [66, 86]. The objective of investigation is to discover the attack process and path and find compromised machines or users [3]. The common approach to these tasks in network security is network traffic monitoring and intrusion and anomaly detection based on network traffic analysis [79]. However, network monitoring produces enormous amounts of data which complicates its analysis. There is a need to use comprehensive approaches to filter and sample the data and find patterns in them; a promising approach is using graphs. The emerging field of graph-based data representation and analysis allows the automatic construction of large graphs from big data and their analysis via advanced graph-theoretical algorithms and techniques, which opens vast opportunities for network security and traffic analysis.

There are early papers from the 1980s and 1990s using graph models to reason about certain security properties [28, 34]. However, they did not have a lot of impact due to the lack of a real and complex environment with the advent of large data, complex attacks, and heterogeneity of systems. The concept of attack graph has been used for decades mostly to model cyberattacks and calculate their impact [46], predict the next step of an adversary [41], or host-based malware detection that rely on various graph types such as—call graphs, mainly directed acyclic graphs where the graphs are extracted from disassembled malware binaries [9] or interaction with system resources [84], and API call sequence graph where the graphs are constructed from a sequence of API events [57].

Such models are popular for their high comprehensibility, straightforward visualization, and extensibility up to recent times [54]. Another well-known use of graphs is for modeling the networks in order to achieve cyber-situational awareness [64]. The obtained graph-based models proved to be valuable to keep track of hosts, services, users, security events, and other entities. Such network-wide graphs allow assessing risks to the organization operating the network, optimizing the network defenses, or facilitating incident response. Recently, network security monitoring and botnet detection systems leveraged communication graph analysis using machine learning to deal more efficiently with the increasing volume of data related to security monitoring [40, 49].

Specifically, the approaches using graph-based data models turned out to be suitable for botnet detection [49, 80], attack visualization [11, 63], and alert correlation [40, 62]. The main advantages of graph-based modeling are its suitability to deal with large volumes of data, its extensibility, its straightforward visualization, and its comprehensibility. Graph-based modeling facilitates understanding complex events and attacks or determining their root cause.

### 1.1 Objectives and contributions

Our objective is to provide a structured and comprehensive overview of recent research and development in network security that uses graphs and graph-based data representation and analytics. We are especially interested in the graph-based representation of network traffic records and the application of graph-based analytics in intrusion detection. Namely, we aim at understanding:

1. **What types of graphs are used to represent network security data?** Are they small or large, densely or sparsely connected, labeled, or weighted?
2. **What kind of approaches are used to analyze such graphs to detect or analyze malicious network activities?** Are they used for human-friendly visualization or are they processed by machine or even with machine learning approaches?
3. **What metrics are used for the detection and monitoring?** We are especially interested in metrics intrinsic to graphs and their representation over the common metrics of intrusion detection.
4. **Are the existing works reproducible?** A research work is reproducible when others can reproduce the results of a scientific study given only the original data, code, and documentation [26].

To meet these objectives, we go thoroughly through existing works and present the following contributions:

1. We present a survey of graph models used to represent, store, and visualize network security data, correlating intrusion detection alerts and constructing attack scenarios.
2. We present a survey of the algorithms and approaches used to analyze such data. Most importantly, we illustrate the most common approaches based on similarity and clustering. Further, we review the emerging approaches based on graph mining. The literature review shows that the main application of such approaches is in botnet detection, among the detection of other malicious activities.
3. We enumerate the most important graph features used for network security analytics, discuss their semantics, and point to their use in related work. We dedicate special attention to the features used in graph mining and learning which turned out to be an emerging and highly promising issue of current research and development.

### 1.2 Literature search methodology and previous surveys

The main challenge of this study is that the theme is covered by several communities. Although the discussed problems are studied in the field of cybersecurity, the topics are often addressed in journals and conferences on computer networks and communications, database systems, formal methods in computer science, and data mining and knowledge discovery. There is no journal nor conference dedicated specifically to graph-based methods in cybersecurity but the topic frequently appears as a topic of special issues and conference workshops, such as GraSec[1] and CNASYS.[2]

Our study is distinguished by covering all these fields. It focuses on papers published in the last five years that discuss specifically the issues of network security monitoring and intrusion and botnet detection using graph models, algorithms, and tools.

There are several comprehensive surveys on graph-based approaches to network security analytics. The earliest survey by Akoglu et al. [1] from 2014 surveyed graph-based techniques to anomaly detection in diverse domains, including network traffic analysis. Later surveys mentioning graph-based approaches focused on particular issues such as attack graph construction [46], network-wide situational awareness [64], threat detection and investigation [56], classification and detection of botnets [2, 33], detecting and preventing insider threats [58], and predicting and projecting cyberattacks [41]. However, none of the previous surveys discusses the different representations of network traffic records into graphs for the needs of network security monitoring and intrusion detection. The exceptions are earlier surveys on botnet detection [2, 33] from 2014 and 2015. The emergence of graph-based data mining and machine learning calls for systemizing the knowledge and surveying these novel approaches. To the best of our knowledge, there is no recent survey covering the progress in the last years. An exception is a brief and thematically broad survey by Shevchenko et al. [75] written in Ukrainian language.

### 1.3 Paper overview and organization

The papers found in the literature search were grouped into three categories, each described in its own section. The first group contains works in which a graph is used as a data structure. Typically, such works use graph models and graph databases to represent and store the data for analysis and visualization. We illustrate various types of graphs used in cybersecurity, their properties, and construction. The second group contains works that use graph algorithms or graph mining to detect malicious activities. Typical work in this section uses graph similarity or graph clustering to detect malicious patterns or anomalies in the data. The third group focuses specifically on graph mining and graph-based features used in graph mining. The papers in this group do not discuss the detection methods or analysis but describe the graph features, their semantics, and significance for the security analysis of network traffic.

The paper is organized as follows. Section 2 lists the challenges of using graphs in network security and basic definitions useful for understanding the paper. Sections 3, 4, and 5 survey the literature in the three categories: graph-based data representation, graph analytics, and graph features. Section 6 summarizes and discusses the existing solutions. We conclude and provide future directions in Sect. 7.

## 2 Graphs in network security

Graphs have multiple uses in network security. Herein, we first provide the basic definitions and terms from graph theory and graph analytics. Subsequently, we highlight the major benefits and challenges of using graphs in network security monitoring. The section closes with an overview of graph-based technologies, including graph databases.

### 2.1 Basic definitions

In this section, we provide the basic definitions related to graphs, graph-based data representation, and the important graph algorithms used for botnet detection and network security monitoring. First, we define various types of graphs used in network security and botnet detection: undirected graphs, directed graphs, bipartite graphs, and weighted graphs.

**Definition 1** (*Graph*) A graph $G = (V, E)$ consists of a nonempty set $V$ of vertices (or nodes) and a set $E$ of edges. Each edge has two nodes associated with it. A graph is $undirected$ if the edges do not have a direction. Otherwise, the graph is $directed$.

**Definition 2** (*Bipartite graph*) A graph $G = (V, E)$ is called $bipartite$ if its node set can be partitioned into two disjoint subsets $V = V_1 \cup V_2$, such that every edge has the form $e = (u, v)$ where $u \in V_1$ and $v \in V_2$ and no nodes both in $V_1$ or both in $V_2$ are connected.

**Definition 3** (*Multigraph*) A graph $G = (V, E)$ is called $multigraph$ if $V$ is a set and $E$ is a multiset of 2-element subsets of $V$, i.e., pair of nodes joined by more than one edge, such edges are called $multiple$ or $paralleledges$.

**Definition 4** (*Weighted graph or property graph*) A graph $G = (V, E)$ is called $weighted$ if it is attributed by a function $w$ that assigns a weight $w(e)$ to each edge $e \in E$.

[1] https://grasec.uni.lu/.

[2] https://www.fvv.um.si/eicc2022/cnacys.html.

These kinds of graphs are also called *property graphs* in the database community.

Moreover, there are three terms used frequently throughout this survey and in related work, namely:

*Graph edit distance* Given $g_1$ and $g_2$, the edit distance between two graphs $g_1$, and $g_2$ is defined by the minimum set of edit operations that are necessary to transform $g_1$ into $g_2$ using edit operations such as insertion, deletion, or relabeling for both nodes or edges [15, 71].

*Clustering* A clustering algorithm measures the density of the partition of nodes of a graph into subgraphs called *modules or communities* by measuring the density of edges inside groups as compared to edges between groups [61]. The nodes in the same group are more close to each other than to those in other groups. An example of a clustering algorithm is the modularity algorithm [61].

*Shortest path* A path in a directed graph is a sequence of nodes where there is a directed edge pointing from each node in the sequence to its successor in the sequence. However, finding all possible paths is an NP-hard problem [45].

*Graph embedding* In machine learning, an auto-encoder learns a representation (encoding) from data, typically for dimensional reduction, and is considered a feature discovery or extraction method. In graph theory, the encoding method of graph data, called *graph embedding*, encodes both the structure of the graph (i.e., nodes and the edges) and the specific information (attributes) associated with them within a vector representation.

## 2.2 Benefits of using graphs

We highlight three main reasons that make graph-based approaches beneficial to prevention, detection, and investigation in network security compared to classical methods, which are mainly signature-based or machine learning-based. The main benefits are:

**Strong and robust representation** The representation and visualization of graphs are straightforward and intuitively comprehensive. The security analysts may have a global view of the entire communication network or a complex attack that can be used for prevention, detection, and investigation [64].

**Relational nature of network security data** The nature of network attacks could exhibit themselves as relational. For example, the propagation of botnet attacks and the communications between source and destination IP addresses can be modeled by graphs. Both of these situations point to the relational treatment of network attacks [62].

**Heterogeneity of security related data** The network data often exhibit linked dependencies that are related to each other. In addition, the graph is used to model homogeneous and heterogeneous data coming from multiple sources [63]. It represents relational data that necessitates being analyzed for finding anomalies [1].

## 2.3 Challenges of using graphs

The issues of graph modeling are twofold: how to represent the data by graphs and how complex the resulting graphs can be. Using graphs in network security is beneficial but also challenging. Herein, we describe the challenges related to using graph models with respect to domain-specific issues of network security.

**Lack of common approaches for security data modeling** There are several approaches to graph representation of security related data and no consensus on which representation is the best. This is an important issue as the analysis algorithms as well as the interpretation of the events depend on the representation [54].

**Complexity of graph algorithms** We need rapid algorithms to respond in real-time on dynamic graphs while most graph problems are hard problems. In fact, many graph problems are NP-hard [45].

**Visualization of large graphs** In security monitoring, visualizing the data is important. Even if a graph has an accessible visualization, visualizing large dense graphs is not simple and may be more complex than row data [11].

**Explaining the suspicious behavior or attacks** Explaining the suspicious behavior or attacks in the post-detection phase to security experts involves mainly explaining the root cause of an attack. Comprehensive representation and visualization based on graphs would be a welcome addition, but it remains a challenging problem due to the complexity of attacks, the heterogeneity of the data, and the combination between them [78].

## 2.4 Graph databases and tools

A graph database is a *NoSQL* database designed for structuring the data in the form of an attributed, directed, and labeled multigraph. The fundamental abstraction behind a database system is its database model. Popular graph databases include ArrangoDB [8], OrientDB [65], DGraph [25], Caley [17], and JanusGraph [43], and Neo4j [53, 59]. Neo4j [59] is the most widely used graph database in network security. It is a native disk-based storage manager that offers high performance and robustness. It also implements an object-oriented API and a framework for graph traversals. A comparison between Neo4j and the other SQL and NoSQL databases [51] highlighted its capabilities of executing complex queries in analyzing security events.

Graph databases use query languages that allow querying the graph-based data. A well-known example is Neo4j's declarative language called CQL (Cypher Query Language [60]). Another popular query language is Gremlin, a query

language co-development with Apache TinkerPop [5], a vendor-agnostic graph-computing framework. Using such a framework allows the user to approach the graph data stored in any supported graph database via a unified interface, be it an in-memory database or a distributed multi-head database. Another interesting graph-processing framework is GraphX [6], a component of Apache Spark engine [4] which is popular among Big Data analysts. While graph databases are suitable for persistent storage of data, GraphX aims at their real-time processing, often on a large scale.

## 3 Graph-based data representation

In this section, we present the graph-based models used to represent network security data such as network traffic records (PCAP or NetFlow), system logs, or alerts from IDS. We show how existing works model this data into a graph and what entities and relations they represent as nodes and edges. The graph models are categorized by the data they represent; each type of data is surveyed in a dedicated subsection.

### 3.1 Network traffic

The raw data in network security are the packet captures (PCAP), where the full packets are saved for analysis, or NetFlow data, in which the data from packet headers are aggregated to so-called flows. NetFlows are used namely in processing large volumes of data. A flow is a sequence of packets that share the same source and destination address and port and protocol. Each flow is accompanied with the number of packets and bytes, timestamps, and protocol-specific information, such as TCP flags. Various researchers leveraged on NetFlow specification when building graph models of network traffic.

Apruzzese et al. [7] proposed a temporal graph to represent NetFlow, where the nodes represent the hosts in the network, and the directed edges are bidirectional network flows with a timestamp as an attribute, see Fig. 1 for example. This is advantageous to represent time causality of network connections.

Leichtnam et al. [55] proposed Sec2graph, an approach to detect anomalies in the network based on graphs constructed over network events. The nodes are called *security*

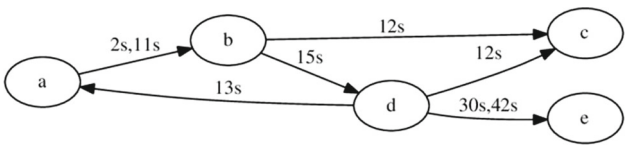



**Fig. 1** Temporal graph representing network flows between five hosts [7]

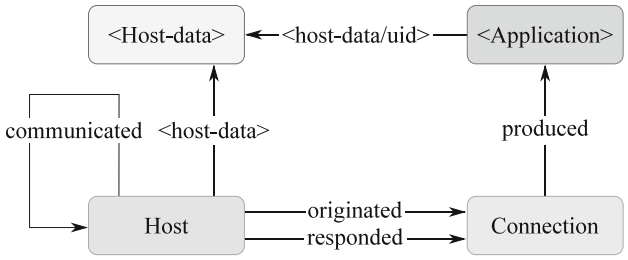



**Fig. 2** Database scheme of GRANEF [18]

*objects* (network connections, IP addresses, ports, protocols, and other entities). The edges are their semantic links. The entities are extracted from Zeek network security monitor tool [86].

A slightly simplified version of Sec2graph tuned to the needs of network forensics was proposed by Čermák and Šrámková [18] in the GRANEF toolkit. In GRANEF, the network connection between hosts in the network is stored in a Dgraph database and visualized in a web-based user interface. The paper focuses on conversion of data from logs to a graph and performance issues. Methods of data analysis are briefly outlined and left for future work (Fig. 2).

Berger et al. [10] proposed an approach to detect malicious websites by monitoring DNS traffic in access networks using graph analysis. They represented their graph as follow: the nodes are Fully Qualified Domain Names (FQDNs) and IP addresses, and edges indicate the existence of a suspicious mapping between them.

### 3.2 Alert correlation

Ben Fredj [29] proposed an approach of alert correlation based on graphs and absorbing Markov chains. They proposed the following weighted directed graph modeling: The nodes represent the alert ID. The edges represent relationships between alerts. Each edge has a weight that corresponds to the number of repetitions of the transition from an alert to another. The graph represents the behavior of alerts. It aggregates and correlates alerts.

Noel et al. [62] proposed a modeling and analytical framework for tracing cyber-attack vulnerability paths through networks, correlated with observed security events. The nodes represent exploit (i.e., attacks), machine, vulnerability, or domain. The edges represent a relation between exploits, machines, vulnerabilities, or domains. There are four relationship labels: IN, ON, LAUNCHES, AGAINST, VICTIM. The directed graph represents an attack graph between subnets, which contain machines with vulnerabilities. Figure 3 shows an attack graph represented as a property graph. The nodes represent the exploits, machines, vulnerabilities, and domains.

Husák and Čermák [40] proposed a graph-based representation to capture the relations between sensors and alerts for alert correlation in the SABU alert sharing platform [19]. The graph shows which sensor (e.g., IDS or honeypot) raises which types of alerts (e.g., scanning, brute-forcing). Further, it shows how often the sensors report the same events and how often the alerts of different types pinpoint to the same attacker. See Fig. 4 for an example. It helps understanding what is happening in a collaborative or distributed intrusion detection system, and, subsequently, design advanced detection methods. The nodes in the graph represent either a sensor or an alert type. Their properties are the numbers of reported alerts. The edges indicate that the sensor detects the type of alert, the two sensors detect attacks from the same source, or that the alerts of two types contained the same target.

Haas et al. [36, 38] proposed two graph representations for alert correlation for the detection of distributed multi-step attacks. The first graph represents a transformation of alerts into a weighted graph. The nodes represent the alerts and the edge represents the similarity between two alerts. The similarity function is based on the attribute of an alert: IP source/destination and port source/destination. The second graph represents the flow graph where the nodes represent the IP source and destination. Both graphs are used for the detection of multi-steps attacks. Figure 5 shows the graphs

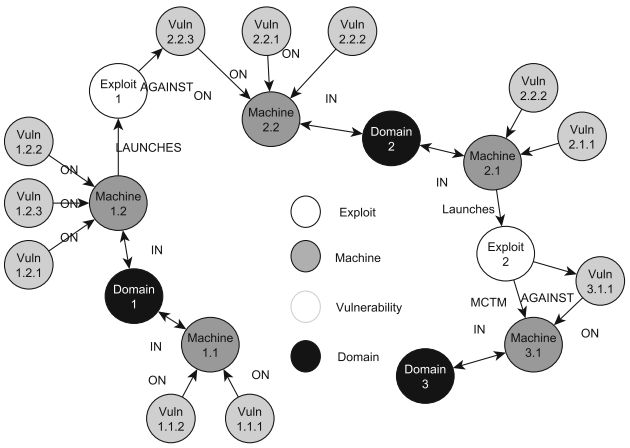



**Fig. 3** Attack graph represented as a property graph [62]

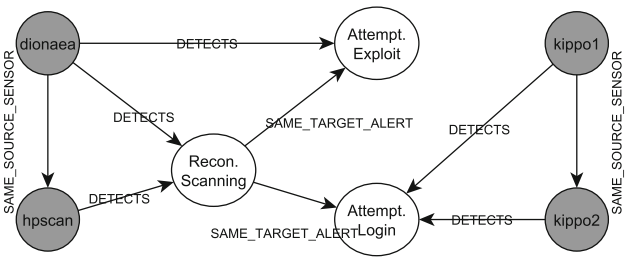
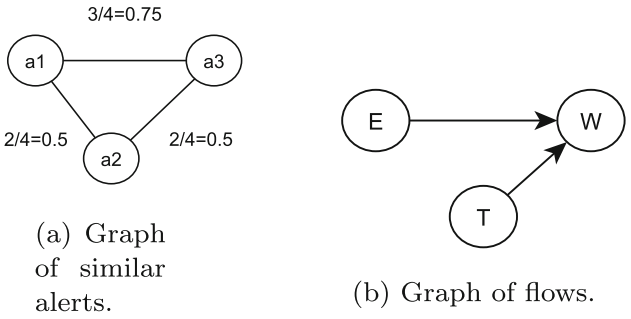



**Fig. 4** Graph representing relations between sensors and alert types in an alert sharing platform [40]




(a) Graph of similar alerts.

(b) Graph of flows.

**Fig. 5** Graph models from alerts set [36, 38]

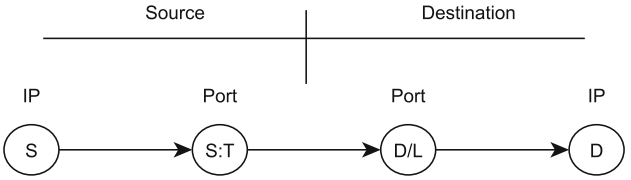



**Fig. 6** Graph communication from alerts set [37]

proposed by Haas et al. [36, 38]. The same authors proposed another graph in [37] for attack correlation and identification of attack scenarios based on network motifs. They build the graph from alerts, where some nodes represent the source and destination hosts and other nodes represent source and destination hosts with their ports (Fig. 6).

Böhm et al. [11] proposed a concept for interactive visual analytics of threat intelligence information. They used a graph database as a back-end for their visual interface supporting security experts in understanding and analyzing incident descriptions. They proposed the following graph representation: The nodes represent threat actors or threat actor group names, individual or organization names. Each node can have the following properties: the description of the threat or organization, the date of first/last seen, and objective of threat or organization. The edges represent relations between threats and individual/organization names. Each edge has a label or a name providing a semantic of the relation between two nodes. For example, Alice "uses" the server S1. "uses" is the edge name between the nodes Alice and S1.

### 3.3 Port scans

Lagraa et al. [50] and Evrard et al. [27] proposed a knowledge discovery approach from port scans. They proposed the following weighted directed graph modeling: The nodes represent targeted port numbers (destination port). The edges represent successive targeted ports in port sequences. The weights of an edge are then the number of dependency occurrences between two successive scanned ports. Figure 7 shows a graph of scanned ports. The graph represents a partial order of vertical scans by seeking the relationship of commonly

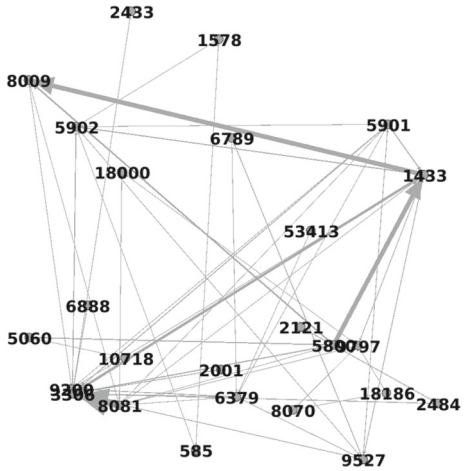



**Fig. 7** A graph of scanned ports [50]

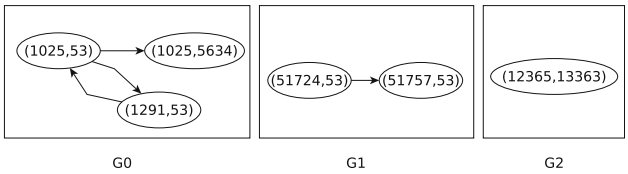



**Fig. 8** A graph dataset for botnet detection. Each graph represents an IP behavior [49]

scanned TCP ports. The authors use the constructed weighted directed graph for extracting clusters of ports scanned commonly.

Lagraa et al. [52] extended their works in [50] to analyze the horizontal scans with enriching clusters semantically. They propose the same weighted directed graph as [50] by replacing targeted port nodes by targeted IP nodes in order to analyze the common IP scans in horizontal scans.

### 3.4 Botnet activity

Lagraa et al. [49] proposed a graph mining approach to detect botnets in traffic flows. They proposed the following directed graph modeling: The nodes represent event attributes or a set of attributes. The edges represent successive events between the event at $t_i$ and the event at $t_{i+1}$. The graph represents the behavior of an entity. An entity could be a user, a source IP, or a pair of source and destination IP. Each entity is represented by a graph of successive event attributes. The entity and event attributes are represented by a key and a value, respectively. Then, the authors construct a set of graphs for behavior analysis for entities. After the graph construction, the authors performed an analysis of a set of graphs in order to detect botnets. An example is given in Fig. 8. This figure represents a graph dataset for botnet detection. Each graph represents an IP behavior [49].

Abou Daya et al. [23] proposed a graph-based machine learning approach for bot detection. They proposed the following weighted directed graph modeling. The nodes represent source or destination IP addresses in the NetFlow data. The edge is a directed edge from source to destination IPs and from destination to source IPs. The weights of the edges are the number of transferred bytes in NetFlow record.

Jaikumar et al. [42] proposed the following weighted, undirected graph-based modeling. The graph represents how the infected computers evolve with time: The nodes represent infected computers. The edges represent an interaction between bots. The edge weight means that two nodes are part of the same botnet. Edge weights are bounded between 0 and 1. A high probability means that an edge weight is close to 1 and the two nodes belong to the same botnet, while a low probability means that an edge weight is close to 0 and the two nodes belong to different botnets. The edge weights represent the temporal co-occurrences of malicious activities.

Wang et al. [83] proposed the following weighted directed graph modeling: The nodes represent source or destination IPs. The edges represent relationships between IPs. The edge weight represents the number of communications between source and destination IPs.

Chowdhury et al. [20], Sinha et al. [76], Shang et al. [73], Wang et al. [81, 82], and Venkatesh et al. [80], proposed the following directed/undirected/bipartite graph modeling: The nodes represent source or destination IPs. The edges represent a relationship between IPs.

Chowdhury et al. [20] represented connections between IP addresses by a directed graph. Sinha et al. [76] represented network communications over time (120 s window) for a set of malicious nodes from a P2P botnet by a directed graph. Wang et al. [73, 81, 82] represented the network communications by an undirected graph. Venkatesh et al. [80] represented P2P communications by an undirected graph for P2P bots detection.

Bou-Harb et al. [12] proposed the following graph for inferring darknet data: The nodes represent bots and the edges denote the probability of behavioral similarity computed by piece-wise comparisons between the feature vectors of each of the nodes.

### 3.5 Authentication events

Amrouche et al. [3] proposed a graph-based malicious login events investigation approach. They proposed the following directed graph modeling: The nodes represent authentication event attributes performed by a user achieving an attack. An event attribute contains all information except the time field: source/destination computer, authentication type, logon type, etc. The edges represent successive events between the event at $t_i$ and the event at $t_{i+1}$. The weights of an edge are then the number of occurrences between two successive events. The

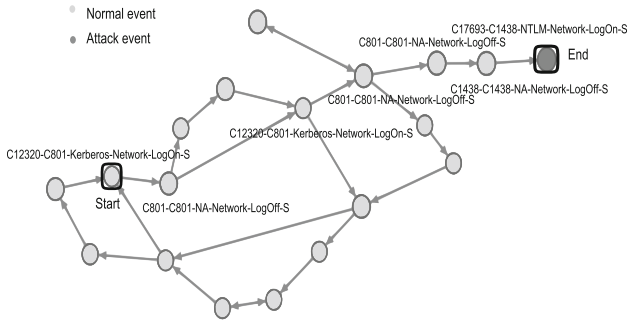



**Fig. 9** Graph of user U7394 in LANL dataset (attack in black) [3]

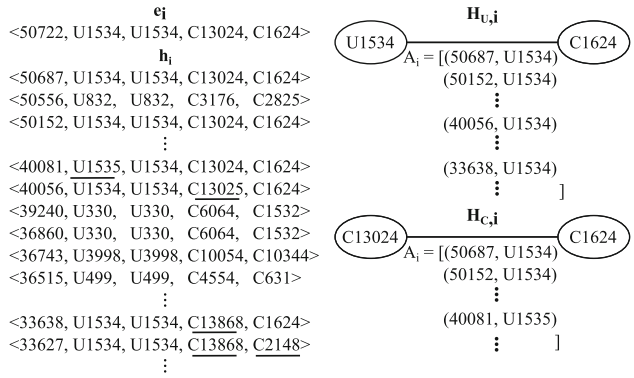



**Fig. 10** In the left, the set of authentication event logs. In the right, the two bipartite graphs extracted from the event logs: one for user–destination relations, $H_{U,i}$, and the second for computers relations, $H_{C,i}$ [44]

graph represents the behavior of a user. Then, each user is represented by a graph of successive events. After the graph construction, the authors performed a graph analysis in order to investigate the paths reaching an attack. Figure 9 represents a behavior of a user targeting a machine for an attack. The attack is represented by a red node.

Kaiafas et al. [44] proposed an approach for detecting malicious authentication events. They proposed two bipartite graphs-based modeling for detecting malicious authentication events. The first bipartite graph is represented as follows: The nodes represent source users and destination computers. The edges represent the relationships between source users and destination computers. An edge shows the relations between a user and the accessed destination computer by a user. They proposed a list of properties represented by tuples as edge properties. A tuple is composed of time and destination user. The second bipartite graph is represented as follows: The nodes represent source and destination computers. The edges represent the relationships between source and destination computers. An edge shows the relations between the used computer to target a destination computer. In the edge property, they used a tuple composed of time and source user. These graphs are constructed from sets of events, by different combinations of user and computer values. The computed bipartite graphs are used to extract features such as graph properties. Figure 10 represents bipartite graphs extracted from the event logs: one for user–destination relations, $H_{U,i}$ (in the top of the figure), and the second for computer relations, $H_{C,i}$ (in the bottom of the figure). It presents a simple example of these graphs. Each graph has 2 nodes and 1 edge.

Bowman et al. [13, 14] proposed to model the authentication events into a graph called *authentication graph* where the nodes represent IPs, users, and services and the edges represent the authentication of a user $u$ to a service $s$ using IP $ip$. The authentication graph is used for the detection of lateral movement of the attacker. Figure 11 shows an example of the authentication graph.

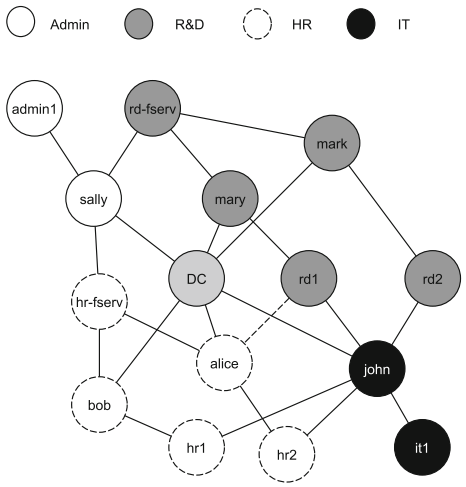



**Fig. 11** Authentication graph [13, 14]

## 3.6 Insider threats

Gamachchi and Boztas [30] proposed the use of attributed graph anomaly detection techniques for malicious activity detection. They proposed a model using a weighted directed graph where the nodes represent users, and the edges represent relationships between users. It is built based on organizational hierarchy or email communications between two users. The undirected graph represents the email communications to capture user relationships within the enterprise network. The relationship between users is captured by analyzing all addresses of emails within an enterprise domain. Then, an edge between the sender and the recipient is created.

Gamachchi et al. [31] proposed a graph-based framework for malicious threat detection. They proposed the following weighted, undirected, bipartite graph-based modeling: The nodes represent users or devices. The edges represent user's interaction with the devices. Edge weights correspond to the number of Log-off activities which appeared during

the whole time duration between an individual user and a device. The graph represents relationships between users and devices.

## 4 Graph-based analytics and mining approaches

Graph algorithms and analytics tools are used to mine network data and infer knowledge about attacks and attackers. Herein, we first comment on the papers discussing various use cases for graph-based analytics in network security. The detection of botnets and botnet-related activities turned out to be the most frequent application of graph analytics in network security and, thus, is discussed in its own subsection.

### 4.1 Security monitoring

#### 4.1.1 Intrusion detection

Sadreazami et al. [70] proposed a statistical-based intrusion detection approach for distributed sensor networks. First, they constructed a graph from both the sensor measurements and placements, resulting in the corresponding similarity and Laplacian matrices. Second, intrusion detection uses a Bayesian method. The authors evaluated their approach on simulated sensor data.

Apruzzese et al. [7] proposed an algorithm to detect pivoting activity, i.e., an activity in which the attacker uses one or more other machines to propagate commands from their machine to another to avoid detection or bypass security measures. Pivoting is considered as a path in the temporal communication graph in which each edge has a timestamp no bigger than the timestamp of the previous edge plus a predefined maximal value of propagation delay.

#### 4.1.2 Port scan detection

Lagraa et al. [50] proposed a solution to discover and detect patterns of port scans. They proposed a graph-based model to represent network packets into a graph. The graph contains the targeted ports by an attacker. It highlights semantic relationships between port numbers. They discovered and inferred the dependency between services using graph clustering in order to analyze the behavior patterns when the port scans are performed. They used methods utilized for clustering discovery in large graphs in order to identify clusters of common scanned services. They showed that there are particular relationships between sequences of scanned ports. They discovered important clusters of port nodes. The clusters are fully connected and contain nonconsecutive and non-randomly probes. It means that the attackers do not randomly target a ports of an organization but there is a semantic behind the probing. In fact, the authors highlight that all the database ports are jointly targeted. It is the same for medical tool ports where medical services are jointly targeted. The weakness of using port numbers with advanced methods is the lack of a proper metric to apprehend the similarity between the scanned ports. This weakness is tackled by the authors in [27, 52]. Lagraa et al. [52] provided an enrichment of the graph model proposed in [50] by meta-data related to services. This is helpful for the security analysts to analyze and evaluate the strategy of the attacker by understanding the types of jointly targeted applications or environments.

Evrard et al. [27] proposed a similarity measure between TCP port numbers which is able to catch the semantic of the port scans by taking into account semantic relations between port numbers. The semantic similarity is based on the shortest path between two ports.

#### 4.1.3 Attack investigation

Amrouche et al. [3] proposed an approach for investigating and tracking malicious activities with authentication events logs. They constructed a behavioral graph from the authentication dataset, where the nodes represent the successive events of an attacker. They profiled the behavior of authentications in order to understand the different steps of attacks using a shortest path algorithm. The shortest path algorithm is used for extracting previous events that occurred before a malicious event.

#### 4.1.4 Alert correlation

Ben Fredj et al. [29] proposed an alert correlation system based on graph modeling. The system deals with heterogeneous alerts in order to recognize multi-step attacks. They use Defcon's datasets. Defcon is the largest Internet security community in the world.

### 4.2 Botnet detection

Lagraa et al. [49] proposed BotGM, a tool to detect botnet behavior based on network traffic flows. It constructs a graph of behavior and uses graph-based mining techniques to detect the dependencies among flows. The advantage of their approach is to trace-back the root causes of an attack. They transformed NetFlow into a behavioral graphs dataset. Each graph represents the behavior of a source IP or pair of source and destination IPs. The nodes of a graph can be successive source/destination ports, etc. For detecting abnormal behavior, BotGM uses pairwise comparisons on a set of behavior graphs using graph edit distance measure. Based on the distances, BotGM uses a statistical method for outlier detection which is the inter-quartile method (boxplot). They applied BotGM on a CTU-13 [77] dataset, where it detects vari-

ous botnet behaviors with a high accuracy without any prior knowledge of them. Their results show that their approach works better in terms of accuracy compared with the techniques developed on the same dataset for three systems, namely BClus, CAMNEP, and BotHunter [32]. However, BotGM implies a high overhead and cannot scale well for large datasets. In fact, for every pair of unique IPs (source and destination IP), a graph is constructed in each time window. Every node in the graph represents a unique 2-tuple of source and destination ports.

Venkatesh et al. [80] proposed BotSpot for C2 channel detection which is an essential component of a botnet. BotSpot exploits the degree of a node, the edge density, and communities in a graph in order to identify dense subgraphs. In addition, BotSpot is based on the differences in the assortativity [3] and density properties of the structured P2P botnets. Based on a classification approach it differentiates between the structured P2P botnets and the legitimate structured P2P applications.

Wang and Paschalidis [81] detected botnets by analyzing the relationships of IPs, modeled as graphs. They proposed an anomaly detection in a graph using large deviations on the degree distribution, and community detection. They also proposed a refined modularity measure (community detection measure) adapted for botnet detection. The authors used the CAIDA dataset [22] for experiments. The results show that it has high detection accuracy. The same authors proposed in [82], a two-stage approach for botnet detection. The first stage applies a sliding window to network traffic and monitors anomalies in the network. While the second stage identifies the bots by analyzing these anomalies using a community detection algorithm. In each sliding window, the anomaly detection method constructs an interaction graph between IPs from packets and monitors the degree distribution in order to detect their deviations. They also detect bots by detecting the community in the graph that exhibits high interaction with highly interactive nodes. For their experiments, they use both CAIDA and CTU-13 datasets.

Haas et al. [36, 38] proposed $GAC$ a graph-based alert correlation approach that can be used for the detection of distributed attacks such as DDoS, port scans, and worm spreading. $GAC$ is composed of three blocks: alert clustering, context of attack scenarios, and attack interconnection. Each of the blocks use a specific graph representation. They detected clusters from a graph of alerts (block 1), then, they contextualize the clustering by specifying and tagging the type of attacks on each cluster (block 2), and finally, they interconnect the attacks based on the context of the clusters (block 3). For the experiments, they evaluated their approach on artificial data and how to identify distributed attack scenarios based on the node-degree among the hosts involved in malicious communication.

In [37], the same authors (Haas et al.) proposed a correlation approach that transforms clusters of alerts into a graph structure on which they computed signatures of network motifs to characterize these clusters. Network motifs are characteristic subgraphs and a motif signature summarizes the occurrence of different types of motifs in a graph of communication. The motifs are used as fingerprints for the attack detection. Their solution is based on a clustering algorithm on a similarity metric. For the experiments, they evaluate their approach on synthetic alerts as well as real-world alerts from DShield [72].

Bou-Harb et al. [12] proposed an approach that exploits darknet data for the following goals: inferring Internet-scale infected bots in a prompt manner, attributing the latter infections to a certain malware type or family, employing a set of behavioral analytics that model the infected machines in conjunction with several graph-theoretical notions.

Berger et al. [10] proposed an approach to detect malicious websites by monitoring DNS traffic in access networks using graph analysis features. Their approach is composed of two steps: the partition of the graph and finding the set of connected components, i.e., subgraphs or clusters which are not connected to each other. In the second step, they removed all clusters which contain only one FQDN and one IP address as such mappings do not represent any kind of agile activity. Agile groups are subject to filtering rules, which are based on a set of queries and statistical metrics such as: the number of FQDNs and IP per agile group. For the experiments, they used datasets from an Internet service provider.

## 5 Graph features

Machine learning and data mining have gained a lot of attention in network security, recently. The approaches based on graphs (colloquially referred to as graph learning and graph mining) are not exceptions. Thus, we decided to delve into its crucial aspect, the feature selection. In fact, in machine learning, features are variables or measurable properties that act as an input to the machine learning model. The model uses the features for different tasks: classification, clustering, prediction, etc. The construction of the features has a high impact on the quality of the model for the different machine learning tasks. In network security, constructing features from NetFlow data or logs is not trivial. The accuracy of the machine learning models depends on the quality of the features, their relationships, and the need of the knowledge of the expert which is important for the construction of features. The literature review shows a significant amount of papers using

[3] A network is said to be assortative when high degree nodes are, on average, connected to other nodes with high degree and low degree nodes are, on average, connected to other nodes with low degree [85].

graph features to detect botnets and several papers using them to detect malicious authentication.

### 5.1 Malicious authentication detection

Kaiafas et al. [44] developed a feature engineering process for detecting malicious authentication. The features are constructed from Windows-based authentication events. For instance, a feature is the number of connections used by a user for connecting to a remote machine during a time period, the number of machines used by a user, etc.

Bowman et al. [14] introduced the use of graph embedding and highlighted the advantages over traditional machine learning techniques. They showed how graph-learning can leverage the topology of the graph to produce improved unsupervised learning results. They applied a graph embedding algorithm by first building an authentication graph (relations between IPs, users, and services), and then applying node2vec [35] for embedding the authentication graph. They evaluated their approach on datasets from Los Alamos National Labs (LANL) [47]. The same authors (Bowman et al.) proposed a technique for detecting lateral movement of Advanced Persistent Threats inside enterprise level computer networks using unsupervised graph learning [13]. The approach consists of two phases: the construction of an authentication graph (similar to the one discussed previously) and an unsupervised graph-based machine learning pipeline. They used auto-encoders algorithms such as DeepWalk [67] and node2vec [35] for embedding the authentication graph. They applied their approach on two distinct datasets representing two contrasting computer networks: The first dataset is from a simulated environment they developed with only a few hosts and the second dataset is from LANL [47].

### 5.2 Graph features in botnet detection

Several graph-based models of network events have been developed for adding new contributions and perspectives to botnet detection and traffic classification. The graph models are proposed in order to use them for extracting features. Then, the features will be used in machine learning algorithms.

In [20, 76, 80, 81], the authors use graph-based features for botnet detection. They proposed a directed graph which represents connections between IP addresses. Chowdhury et al. [20] extracted the following features from the constructed graph: in-degree (weight), out-degree (weight), clustering coefficient, betweenness (measures the number of shortest paths that pass through a node), and eigenvector centrality. Then, the authors applied a clustering method to construct clusters of nodes in the network based on these features.

Sinha et al. [76] extracted the following features from the constructed graph in each time interval: in-degree, out-degree, in-neighbors, out-neighbors, PageRank, centrality, betweenness eigenvector centrality, authority and hub centralities, and local clustering coefficient (quantifies the neighborhood connectivity of a node). They extracted features in each time interval. It allows to track the temporal evolution of botnet communication structure and analyze network activity over time. Then, a supervised approach such as long short-term memory (LSTM) [39] based neural network architecture is used to detect malicious botnet hosts.

Abou Daya et al. [23] proposed an anomaly-based approach for bot detection, robust to zero-day attacks. Their approach is based on feature extraction from the constructed graph. The features are: in-degree, out-degree, in-degree weight, out-degree weight, betweenness centrality, local clustering coefficient, alpha centrality (measures the centrality of a node). The features are used for machine learning algorithms such as logistic regression support vector machine feed-forward neural network and decision trees. Their system detects the different types of bots in the CTU-13 dataset.

Shang et al. [73] proposed a hybrid analysis approach on flow-based and graph-based features of network traffic for botnet detection. The graph-based features are in-degree, out-degree, in-degree weight, out-degree weight, local clustering coefficient, betweenness and pageRank. The flow-based features are statistical metrics, excluding the source and destination IP and port. For instance, total number of transmitted packets, number of small packets less than 400 bytes. The authors applied anomaly detection models including k-means, K-nearest neighbor (k-NN), and one-class support vector machine (On-class SVM) on combined both features. For the experiments, they evaluated their approach on a simulated and a real computing environment.

We see that there are common metrics extracted from a graph in order to use them as features for machine learning algorithms. There metrics are: in-degree (weight), out-degree (weight), clustering coefficient, betweenness, and eigenvector centrality. We notice that the graph-based features approaches are very recent from 2017 to 2019. Most of them target the problem of botnet detection on NetFlow and particularly on CTU-13 dataset. Using the graph properties as features in order to apply machine learning algorithms for botnet detection is a good start. In fact, the graph concentrates the structure of the communications or connections that cannot be shown by the classical methods which extract features directly from NetFlow or event log data.

Recently, Leichtnam et al. [55] proposed an unsupervised learning approach based on auto-encoder algorithm for detecting network attacks. Leichtnam et al. [55] developed their own auto-encoder due to their specific graphs, i.e., multi-attributes and heterogeneous graphs. For the experiments, they applied their approach to the CICIDS2017 [74] dataset.

**Table 1** Summary of graph-based approaches for network security

| Paper | Nodes | Edges | Graph | Application | Problem | Graph solution |
|---|---|---|---|---|---|---|
| *Graph mining and analytics* | | | | | | |
| Lagraa et al. [50] | Ports | Successive ports | Directed | Monitoring | Port scanning similarities | Clustering |
| Lagraa et al. [52] | Ports | Successive ports | Directed | Monitoring | Port scanning similarities | Clustering/pattern mining |
| Evrard et al. [27] | Ports | Successive ports | Directed | Monitoring | Semantic port scanning similarities | Shortest paths |
| Amrouche et al. [3] | Events | Successive events | Directed | Investigation | Knowledge extraction from attacks | Shortest paths |
| Lagraa et al. [49] | Ports/events | Successive ports/events | Directed | Detection | Botnet detection | Graph edit distance |
| Venkatesh et al. [80] | IP | Communications | Undirected | Detection | P2P bots detection | Clustering |
| Wang et al. [81] | IP | Communications | Undirected | Detection | Botnet detection | Clustering |
| Wang et al. [82] | IP | Communications | Undirected | Detection | Botnet detection | Anomaly detection |
| Ben Fredj et al. [29] | Alert | Relationship | Weighted Directed | Prevention | Alert correlation | Classification |
| Haas et al. [36, 38] | Alert, IP | Relationship | (Un) Weighted (Un) Directed | Detection | Alert correlation | Clustering |
| Haas et al. [37] | IP, IP:port | Communication | Weighted Directed | Detection | Alert correlation | Clustering |
| Husák et al. [40] | Sensors and Alerts | Relationship | Undirected | Monitoring/visualization | Alert correlation | Querying |
| Böhm et al. [11] | Threat/organization | Relationship | Undirected | Visualization | Visualization | Querying |
| Apruzzese et al. [7] | Hosts | Communication | Directed | Detection | Pivoting detection | Finding paths |
| Sadreazami et al. [70] | IP | Relationship | Weighted graph | Detection | Intrusion detection in sensors | Statistical |
| Čermák and Šrámková [18] | Hosts and connections | Actions | Directed | Investigation/visualization | Network forensics | Querying |
| Bou-Harb et al. [12] | IP | Relationship | Weighted graph | Monitoring | Behavior analytics | Graph theory |
| Berger et al. [10] | IP, FQDNs | Relationship | Undirected graph | Monitoring | Cybercrime detection | Graph clustering |
| *Graph-based features* | | | | | | |
| Kaiafas et al. [44] | User/computer | Connections | Undirected | Detection | Malicious authentication events | Supervised learning |
| Chowdhury et al. [20] | IP | Communications | Directed | Detection | Botnet detection | Clustering |
| Sinha et al. [76] | IP | Communications | Directed | Detection | Botnet detection | LSTM |
| Daya et al. [23] | IP | Communications | Weighted Directed | Detection | Botnet detection | Machine learning |
| Daya et al. [24] | IP | Communications | Weighted Directed | Detection | Botnet detection | Neural network |
| Wang et al. [83] | IP | Communication | Weighted Directed | Detection | Botnet detection | Machine learning |
| Shang et al. [73] | IP | Communication | Undirected | Detection | Botnet detection | Clustering |
| Leichtnam et al. [55] | Security objects | Relationship | Multi-attributes and heterogeneous | Detection | Attack detection | Auto-encoder, machine learning |
| Bowman et al. [13, 14] | IP, user, service | Authentication | Undirected | Detection | Lateral movement | Auto-encoder, machine learning |

**Table 1** continued

| Paper | No. of graphs | Data used | Big data | Large graph | Heterogeneous | Time complexity | Runtime | Scalability | Available code |
|---|---|---|---|---|---|---|---|---|---|
| *Graph mining and analytics* | | | | | | | | | |
| Lagraa et al. [50] | 1 | Darknet | Yes | No | No | No | No | No | No |
| Lagraa et al. [52] | 1 | Darknet | Yes | No | No | No | No | No | No |
| Evrard et al.[27] | 1 | Darknet | Yes | No | No | No | No | No | No |
| Amrouche et al. [3] | * | LANL | Yes | No | No | No | No | No | No |
| Lagraa et al. [49] | * | CTU-13 | No | No | No | No | No | No | No |
| Venkatesh et al. [80] | 1 | CAIDA | Yes | Yes | No | $\mathcal{O}(log\|V\|)$ | No | No | No |
| Wang et al. [81] | 1 | CAIDA | No | No | No | No | No | No | No |
| Wang et al. [82] | * | CAIDA, CTU-13 | Yes | No | No | No | No | No | No |
| Ben Fredj et al. [29] | * | Defcon | Yes | No | No | Linear | No | No | No |
| Haas et al. [36, 38] | 2 | Artificial | No | No | No | No | No | No | No |
| Haas et al. [37] | 2 | Artificial, DShield | Yes | No | No | No | No | No | No |
| Husák et al. [40] | 1 | SABU | No | No | Yes | No | No | No | No |
| Böhm et al. [11] | 1 | STIX | No | No | Yes | No | No | No | Yes |
| Apruzzese et al. [7] | 1 | NetFlow | Yes | No | No | $\mathcal{O}(m^{L_{max}} \cdot log_2(m) \cdot \tau)$ | Yes | Yes | No |
| Sadreazami et al. [70] | 1 | Artificial | No | No | No | No | Yes | No | No |
| Čermák and Šrámková [18] | 1 | Zeek | No | No | Yes | No | No | No | No |
| Bou-Harb et al. [12] | 1 | Darknet | Yes | No | No | No | Yes | No | No |
| Berger et al. [10] | 1 | Internet service provider | Yes | Yes | Yes | $\mathcal{O}(V+E)$ | Yes | No | Yes |
| *Graph-based features* | | | | | | | | | |
| Kaiafas et al. [44] | 1 | LANL | Yes | No | Yes | No | No | No | No |
| Chowdhury et al. [20] | 1 | CTU-13 | No | No | No | $\simeq \mathcal{O}(S^2)$ [69] | No | No | No |
| Sinha et al. [76] | * | CTU-13 | No | No | No | No | No | No | Yes |
| Daya et al. [23] | 1 | CTU-13 | No | No | No | No | No | No | No |
| Daya et al. [24] | 1 | CTU-13 | No | No | No | No | No | No | No |
| Wang et al. [83] | 1 | No | Yes | No | No | No | No | No | No |
| Shang et al. [73] | 1 | Artificial | Yes | No | No | No | No | No | No |
| Leichtnam et al. [55] | 1 | CICIDS2017 | No | No | Yes | No | No | No | No |
| Bowman et al. [13, 14] | 1 | Simulated, LANL | No | No | Yes | No | No | No | No |

$S$ represents the size of the sample. $m$ is the number of network flows within a time window, $L_{max}$ is the maximum of searched path length, and $\tau$ is the maximum number of flows between any time interval. $V$ is the set of vertices. The symbol $*$ stands for many graphs

## 6 Summary and discussion

In this section, we first provide a summary of all the surveyed works. Subsequently, we discuss the findings of the survey, starting with the answers to questions stated in the introduction and followed by the discussion of the metrics. Finally, we summarize the resolved problems and formulate open research challenges.

### 6.1 Summary of related work

Table 1 summarizes all the approaches within the three categories presented in previous sections. The table summarizes these approaches according to the following facets:

- **Graph** The type of graph, e.g., directed or weighted.
- **Nodes/Edges** What do the nodes and edges represent.
- **Application** The targeted application, such as intrusion detection or network forensics.
- **Problem** The problem targeted in the related work.
- **Solution** The solution used to solve the problem. The solution could be based on graph representation, analysis, mining or learning, or specific graph features.
- **Number of graphs** The number of graphs used for resolving a problem. It means the number of constructed graphs for analysis. The symbol * stands for many graphs.
- **Data used** The dataset used for the experiments.
- **Big data** It is a Boolean metric measuring how much the IP traffic is big? A data is big when the size of the IP traffic is greater than 35.5 gigabytes per month. This number is estimated from the report published by CISCO in [21].
- **Large graph** We say that a graph is large if the number of vertices and edges are greater than million.
- **Heterogeneous** A graph is heterogeneous if it contains different types of nodes and edges.
- **Time complexity** It is a notion which is often addressed in algorithmic classes, but not in machine learning algorithms. It is harder to evaluate the complexity of a machine learning algorithm, especially as it may be implementation dependent, input parameters passed to the algorithm, properties of the data (categorical, numerical) may lead to other algorithms. In our comparison, we put the exact time complexity of graph theory algorithms and approximate the machine learning algorithms. The approximation is noted by the symbol $\simeq$.
- **Runtime** It is a Boolean variable showing if the authors compute the runtime of their algorithm.
- **Scalability** It is a Boolean variable showing if the authors measure the scalability of their algorithm.
- **Available code** It is a Boolean variable showing if the source code of the proposed tool is public or not.

**Table 2** Graph dataset characteristics in each research paper

| Paper | Avg of vertices | Avg of edges | Type |
|---|---|---|---|
| *Graph mining and analytics* | | | |
| Lagraa et al. [50] | 1169 | 290,359 | Sparse |
| Lagraa et al. [52] | 1169 | 290,359 | Sparse |
| Evrard et al. [27] | N/D | N/D | N/D |
| Amrouche et al. [3] | 657 | 189,871 | Sparse |
| Lagraa et al. [49] | N/D | N/D | N/D |
| Venkatesh et al. [80] | 1,997,513 | 9,488,076 | Dense |
| Wang et al. [81] | 396 | N/D | N/D |
| Wang et al. [82] | 396 | N/D | N/D |
| Ben Fredj et al. [29] | ∼17 | ∼53 | Dense |
| Haas et al. [36, 38] | N/D | N/D | Dense |
| Haas et al. [37] | N/D | N/D | Sparse |
| Husák et al. [40] | N/D | N/D | N/D |
| Böhm et al. [11] | N/D | N/D | N/D |
| Apruzzese et al. [7] | N/D | N/D | N/D |
| Sadreazami et al. [70] | N/D | N/D | N/D |
| Čermák and Šrámková [18] | 718,475 | 397,632 | Sparse |
| Bou-Harb et al. [12] | 87 | N/D | N/D |
| Berger et al. [10] | 14.6M | N/D | N/D |
| *Graph-based features* | | | |
| Kaiafas et al. [44] | 403 | N/D | N/D |
| Chowdhury et al. [20] | 227,949 | N/D | N/D |
| Sinha et al. [76] | N/D | N/D | N/D |
| Daya et al. [23] | 250,359 | N/D | N/D |
| Daya et al. [24] | 250,359 | N/D | N/D |
| Wang et al. [83] | N/D | N/D | N/D |
| Shang et al. [73] | N/D | N/D | N/D |
| Leichtnam et al. [55] | N/D | N/D | N/D |
| Bowman et al. [13, 14] | N/D | N/D | N/D |

Avg of vertices: average number of vertices. Avg of Edges: average number of edges. Type: type of a graph (sparse/dense). N/D is not defined

The important findings of this survey are listed in the following subsection as either resolved problems or open challenges.

Throughout this survey, we can see that various solutions are proposed for graph modeling, analysis, and mining for network security purposes. Most of the works construct and analyze one graph. The prevalent use case, outstanding among other network security use cases, is botnet detection.

#### 6.1.1 Graph-based data representation

Our survey shows that there are a plethora of models designed to approach various goals; there is no unifying or common model used by a significant number of researchers. The existing works typically create their own model and choose custom semantics to nodes, edges, and their proper-

ties. Different types of graphs have been used: (un)directed, (un)weighted graphs for a problem due to different manners of representing data into a graph and the difficulty to find the best representation. Many works represent the IP addresses as nodes and the communication between them as edges. Nevertheless, the graph representation should include more information. The vital pieces of information are the timestamps, port numbers, numbers of transferred bytes. Most of the graphs are static and do not take into consideration the time, which would make them dynamic.

The most common network data to model as a graph are network traffic records in PCAP or NetFlow format, including the data generated by Zeek [66, 86]. CTU-13, a publicly available dataset containing 13 separate scenarios and different botnet families [32], is the most widely used in the surveyed works. It might be worth recommending using the dataset in future work to allow for comparison to previous work.

Table 2 shows graph dataset characteristics in each research paper. We highlight the average number of vertices and edges as well as the type of graph: sparse or dense. A dense graph is a graph in which the number of edges is close to the maximal number of edges. A sparse graph is a graph in which the number of edges is much less than the possible number of edges. We notice that the majority of works do not describe the constructed or used graph. The description of the graph is important for comparisons and measuring the performance of the proposed detection tool. It allows to have an overview of the graph. We see that in the defined graph characteristics, the graphs are not large and in some cases they do not reflect the real-world cases.

#### 6.1.2 Graph analysis

Several works have been proposed to tackle the modeling of data into a graph for both security monitoring and botnet detection problems. The graphs are modeled for each problem and objective, and the application of mining algorithms depend on the targeted problem and objective.

For the graph solutions, most of the works use classical graph theory algorithms such as shortest paths or clustering, but recently the use of neural network solutions provides an interesting perceptive by increasing the detection of bots.

Table 3 shows the advantage and disadvantage of the proposed approaches. The algorithm column focuses on the main contribution which is the use of a graph algorithm. In the case, when the graph algorithm in the feature discovery process, we put the type of the machine learning approach.

#### 6.1.3 Graph features

Regarding graph-based machine learning, the authors apply different unsupervised and supervised machine learning models depending on the problem and objectives; there is no prevalent approach. On the contrary, the authors extract quite similar sets of graph features such as the degree of nodes, centrality of the graph, communities. The graph metrics are used as features for machine learning algorithms.

### 6.2 Evaluation metrics

Table 5 shows a comparison of computed metrics for the network security. These metrics are used for measuring the performance of a detection tool. In fact, there are various ways to evaluate a model. We enumerate some of the most popular metrics used for the attack and threat detection.

**Confusion Matrix** Confusion matrix is not a metric, but it is a key concept in classification performance of machine learning models. It is a tabular visualization of the model predictions versus the ground-truth labels. Each row of confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class. For example, let us consider we are building a binary classification to classify attack events from non-attack events. Let us assume our test set has 1100 events (non-attack events, and 100 attack events), with the confusion matrix in Table 4.

Out of 100 attack events the model has predicted 90 of them correctly and has misclassified 10 of them. If we refer to the attack events class as positive and the non-attack events class as negative class, then 90 samples predicted as attack events are considered as **true-positive (TP)**, and the 10 samples predicted as non-attack events are **false negative (FN)**. Out of 1000 non-attack events, the model has classified 940 of them correctly, and misclassified 60 of them. The 940 correctly classified samples are referred as **true-negative (TN)**, and those 60 are referred as **false-positive (FP)**.

In Table 4, the diagonal elements of this matrix denote the correct prediction for different classes, while the off-diagonal elements denote the misclassified events.

**Classification accuracy** Classification accuracy (*accuracy*) is defined as the number of correct predictions divided by the total number of predictions. For example, in Table 4, out of 1100 events 1030 are predicted correctly, resulting in a classification accuracy of accuracy $= (90 + 940)/(1000 + 100) = 1030/1100 = 93.6\%$.

**Precision** Classification accuracy is not a good indicator of a machine learning model performance in many cases. One of these cases in botnet detection is when a class distribution is imbalanced. Imbalanced data is one class is more frequent than others (attacks versus non-attacks). In this case, if the prediction of all samples as the most frequent class, then the model gets a high accuracy rate, which not accurate because the model is not learning anything, and is predicting everything as the top class. For example, in Table 4, if the model predicts all samples as non-attack events, it would result in a $1000/1100 = 90.9\%$.

**Table 3** Advantage and disadvantage of each approach

| Paper | Algorithm | Advantage | Disadvantage |
|---|---|---|---|
| *Graph mining and analytics* | | | |
| Lagraa et al. [50] | Modularity clustering | Extracts clusters with quite-small computational cost | Fails to detect communities smaller than a scale |
| Lagraa et al. [52] | Modularity clustering | Extracts clusters with quite-small computational cost | Fails to detect communities smaller than a scale |
| Evrard et al. [27] | Shortest paths | Is enough efficient to use for relatively large problems | A blind search by consuming a lot of time and resources, if not guided |
| Amrouche et al. [3] | Shortest paths | Is enough efficient to use for relatively large problems | A blind search by consuming a lot of time and resources, if not guided |
| Lagraa et al. [49] | Graph Edit Distance | Measuring the similarity between pairwise graphs | Computational complexity which is exponential in the number of nodes of the involved graphs |
| Venkatesh et al. [80] | Community detection | Highlights the botnet as a community | Some groups of the botnet may be misclassified |
| Wang et al. [81] | Community detection | Highlights the botnet as a community | Some groups of the botnet may be misclassified |
| Wang et al. [82] | Community detection | Highlights the botnet as a community | Some groups of the botnet may be misclassified |
| Ben Fredj et al. [29] | Classification | No prior knowledge and no training required | Without prior knowledge sometime is not good |
| Haas et al. [36, 38] | Clustering | Discovering similar alerts by reducing the false-positive alerts | Focusing on alerts for the detection may be risky |
| Haas et al. [37] | Querying | A motif representation of attack characteristics is like a signature-based detection | May lose some unknown attacks (patterns) |
| Husák et al. [40] | Querying | allows to focus on specific graph patterns | Difficult and time-consuming to develop the Querying patterns |
| Böhm et al. [11] | Visualization | Offers a global and local overview | Difficult to visualize when the graph is huge |
| Apruzzese et al. [7] | Finding path | Provides an interpreting perspective to the analysts regarding the root cause of an attack | May lose some paths and increases false negatives |
| Sadreazami et al. [70] | Bhattacharyya distance | Appropriates for stochastic model updating where the distributions of the features cannot be exactly determined | Measures the similarity of two probability distributions |
| Čermák and Šrámková [18] | Querying | The connection of exploratory analysis of network traffic with results visualization allowing analysts to easily go through the acquired knowledge and visually identify interesting network traffic | Difficult to developed graph queries and visualize a large graph |
| Bou-Harb et al. [12] | Graph inference | Allows to define complex botnet | needs a little bit time-consuming |
| Berger et al. [10] | Clustering | Discovering repeating patterns | Focusing on a specific pattern can lose others |
| *Graph-based features* | | | |
| Kaiafas et al. [44] | Ensemble learning | Can make better predictions and achieve better performance than any single model | Less interpretable and the output of the ensembled model is hard to predict and explain |
| Chowdhury et al. [20] | Self-organizing map | Very simple/easy to understand and use | Difficult to determine what input weights to use |

**Table 3** continued

| Paper | Algorithm | Advantage | Disadvantage |
|---|---|---|---|
| Sinha et al. [76] | Long Short-Term Memory (LSTM) | Uses previous time events for training/prediction | Requires a lot of resources and time to get trained |
| Daya et al. [23] | Unsupervised + Supervised | Combines them together for better results | Difficult to measure uncertainties of the results from each individual model |
| Daya et al. [24] | Unsupervised + Supervised | Combines them together for better results | Difficult to measure uncertainties of the results from each individual model |
| Wang et al. [83] | Hybrid analysis | The use of different techniques allows to increase the results | The detection process is more likely to take more time and effort |
| Shang et al. [73] | Hybrid analysis | The use of different techniques allows to increase the results | The detection process is more likely to take more time and effort |
| Leichtnam et al. [55] | Novelty Detection | The ability to adapt to non-stationary data | Assumes that the positive class is very well sampled, while the other class(es) is/are severely under-sampled |
| Bowman et al. [13, 14] | Anomaly detection | Can help to detect unknown attacks | May not be accurate |

Thus, the *precision* metric is suitable for measuring at class specific performance, which is defined as: $Precision = TP/(TP + FP)$.

The precision of attack events and non-attack events class in Table 4 can be calculated as:

- Precision_attack_events = number of samples correctly predicted attack events/number samples predicted as attack events = 90/(90+60) = 60%.
- Precision_non_attack_events= 940/950= 98.9%.

**Recall** Recall is defined as the fraction of samples from a class which are correctly predicted by the model. Formally is defined as follows: $Recall = TP/(TP + FN)$. Thus, the recall rate of attack events and non-attack events classes can be found as:

- $Recall\_attack\_events = 90/100 = 90\%$.
- $Recall\_non\_attack\_events = 940/1000 = 94\%$.

**F1-Score** F1-Score is a combination of the precision and recall into a single metric, which is the harmonic mean of precision and recall defined as: $F1{-}score = 2{*}Precision{*}Recall/(Precision + Recall)$. Thus, for our classification example in Table 4, the F1-score is calculated as:

$$F1\_attack\_events = 2 * 0.6 * 0.9/(0.6 + 0.9) = 72\%.$$

**ROC Curve** The receiver operating characteristic (ROS) curve is a plot which shows the performance of a binary classifier as a function of its cut-off threshold. It essentially shows the true-positive rate (TPR) against the false-positive rate (FPR) for various threshold values.

**AUC** The area under the curve (AUC) is an aggregated measure of performance of a binary classifier on all possible threshold values (and therefore it is threshold invariant). AUC calculates the area under the ROC curve, and therefore, it is between 0 and 1. The interpretation of AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

**Table 4** Example of a confusion matrix

| | Actual class | |
|---|---|---|
| | Attack events | Non-attack events |
| *Predicted class* | | |
| Attack events | 90 | 60 |
| Non-attack events | 10 | 940 |

In Table 5, most of the works compute the true positive, false positive, and accuracy. Few works go further in the measurement of the performance models by computing the precision, recall, F1-score, ROC, and AUC. These latter are very important when the classes are imbalanced involving two classes: a negative case with the majority of examples (normal flows) and a positive case with a minority of examples (abnormal flows). They are used for diagnostic and in the interpretation of binary classification models. The future works need to compute these metrics in order to diagnostic better their model.

## 6.3 Answers to questions

In the introduction, we asked questions in order to get answers related to the graph-based representation of network traffic records and the application of graph-based analytics in network security problems such as the intrusion detection and monitoring.

**Table 5** Comparison of computed metrics

| Paper | TP | FP | Accuracy | Precision | Recall | F1-score | ROC | AUC |
|---|---|---|---|---|---|---|---|---|
| *Graph mining and analytics* | | | | | | | | |
| Lagraa et al. [50] | N/D | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Lagraa et al. [52] | N/D | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Evrard et al. [27] | N/D | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Amrouche et al. [3] | N/D | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Lagraa et al. [49] | • | • | • | N/D | N/D | N/D | N/D | N/D |
| Venkatesh et al. [80] | N/D | N/D | • | • | • | • | N/D | N/D |
| Wang et al. [81] | • | • | • | N/D | N/D | N/D | • | N/D |
| Wang et al. [82] | • | • | • | • | • | • | • | N/D |
| Ben Fredj et al. [29] | N/D | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Haas et al. [36, 38] | • | • | • | • | N/D | N/D | N/D | N/D |
| Haas et al. [37] | • | • | • | N/D | N/D | N/D | N/D | N/D |
| Husák et al. [40] | N/D | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Böhm et al. [11] | N/D | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Apruzzese et al. [7] | N/D | N/D | • | N/D | N/D | N/D | N/D | N/D |
| Sadreazami et al. [70] | • | • | N/D | N/D | N/D | N/D | • | N/D |
| Čermák and Šrámková [18] | N/D | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Bou-Harb et al. [12] | • | • | • | N/D | N/D | N/D | N/D | N/D |
| Berger et al. [10] | • | • | N/D | • | N/D | N/D | N/D | N/D |
| *Graph-based features* | | | | | | | | |
| Kaiafas et al. [44] | • | • | • | • | • | • | N/D | N/D |
| Chowdhury et al. [20] | • | N/D | N/D | N/D | N/D | N/D | N/D | N/D |
| Sinha et al. [76] | • | • | N/D | N/D | N/D | N/D | • | • |
| Daya et al. [23] | • | • | • | • | N/D | N/D | N/D | N/D |
| Daya et al. [24] | • | • | • | • | N/D | N/D | N/D | N/D |
| Wang et al. [83] | • | • | • | • | • | • | • | N/D |
| Shang et al. [73] | • | • | • | • | • | • | • | N/D |
| Leichtnam et al. [55] | • | • | • | • | • | • | N/D | N/D |
| Bowman et al. [13, 14] | • | • | N/D | N/D | N/D | N/D | N/D | N/D |

N/D is not defined

1. **Question** What types of graphs are used to represent network security data? **Answer** The trivial representation of network security data is the directed graph. However, a weighted directed graph is used for botnet detection or network monitoring. Weighted undirected graph is also used for monitoring and measuring the similarity between entities (e.g., IP addresses, domains, users).
2. **Question** What approaches are used to analyze such graphs to detect or analyze malicious network activities? **Answer** The frequent used approaches are unsupervised learning approaches where there is no need of labels for training and detecting. The network security problems is translated either to outlier detection, clustering, or querying problems. In outlier detection, the outliers are considered as anomalies, threats, or attacks. In clustering, the data is grouped into clusters to be analyzed. In querying, the graph is queered for finding specific patterns in a graph.
3. **Question** What are the metrics used for the detection and monitoring? **Answer** In machine learning domain, the metrics are very important scores for measuring the performance of a classifier tool. However, for the network security problems, not all metrics are used, most of them used the accuracy, true-positive, and false-negative rates. These three metrics are not sufficient for measuring the strength of a model. Thus, all metrics can be used for measuring the strong and weak points of a model.
4. **Question** Are the existing works reproducible? **Answer** Very few research papers share their data and code, and if so, it is not always well documented. This is an issue for the progress of the network security research. Comparing with other domains, in which similar approaches are used, such as natural language processing or image

processing, the code, data, and documentation are often publicly available. Thus, the research community of network security should progress in the reproducibility of research papers. However, we are aware that the insufficient number of usable datasets in cybersecurity is given by their rapid obsolescence caused by constantly changing threat landscape and the rapid evolution of attackers and protected systems.

## 6.4 Open challenges and future prospects

Despite several research works in the past decades, there are still several aspects to be explored in the intersection of network security and graphs. In fact, the arrival of big data, the complexity of attacks, and the heterogeneity of data, there is a need for techniques and algorithms adapted to these characteristics.

### 6.4.1 Graph-based data representation

The input data typically do not form a graph; it is up to the researcher or security analysts to construct it. There are many existing models and different types of graphs. In fact, the authors model the data into a graph for each problem and objective. Thus, there is no unique graph model for all problems. Without expert knowledge, it is hard to select an existing model or to design an optimal model for a specific problem or objective, especially when graph mining or learning is considered.

Although the use of graph databases is on the rise, there are few works that use graph databases, namely for botnet detection. If the researchers use a graph database, they most often use Neo4j and Cypher query language. In future, we may expect wider use of GraphQL [68], a graph query language for API that allows for integration of graph databases with other tools and integrating them to security services.

When the graph database is used, there are typically no proofs of their efficiency for a resolved problem. The efficiency could be in terms of speedup, horizontal and vertical scalability, or memory consumption. Only a few works discuss the differences between graph database and alternative options [51]. Instead of using graph databases, many researchers load and save the graph models in a file. Loading the graph in the memory each time when the user wants to process or query the data can be a constraint, especially when the graph is large.

### 6.4.2 Graph analysis

The analysis of graph is lacking computational streaming models [48]. Streaming models address updating graph analysis results given a starting result and snapshot views of the changing graph. Streaming models are suitable for dynamic graphs. Another aspect not taken into consideration is the scalability of graph-processing computations. In addition, the analysis of large graphs has not received considerable attention. The proposed solutions would often not be suitable for processing big graphs.

The reproducibility of research and experiments is a challenge in many fields; it is especially challenging in network security. In fact, most of the research papers are difficult to reproduce due to the insufficient description of the approach or the data and source codes that are not published. We already mentioned that many works on botnet detection use the CTU-13 dataset. Nevertheless, the datasets in network security become obsolete extremely fast due to the continuously evolving threats, attacks, and network traffic patterns. The situation is slowly changing due to the adoption of Open Science practices. Still, it might be problematic to compare novel approaches suited to detect current threats to the previous work suited to detect past attacks. Moreover, we face problems comparing the existing approaches to network security (e.g., botnet detection) based on graphs to other approaches. There will be the need to set up a set of metrics to compare graph-based and non-graph-based approaches and quantify the benefits of such approaches.

### 6.4.3 Graph features

The issue of explainability of approaches based on machine learning was not discussed in the literature in the cybersecurity context and is an open challenge. The development of new graph mining algorithms and explainable embedding solutions could be one of the solutions. In fact, in the existing solutions, we can find two types of features: features extracted from the graph (e.g., in-degree, out-degree) or learned graph (e.g., using neural networks). The approaches based on the first type of features are very easy to understand, and the model based on these features could be explainable when they are combined with explainable machine learning models (e.g., decision tree, k-means). However, they may suffer from low accuracy. On the contrary, the approaches based on the second type of features are not easily explainable, but they may achieve higher accuracy. Experimental comparisons of the two types of features should be performed, and a combination of them should be proposed in order to find a balance between explainability and accuracy.

### 6.4.4 Graph neural network for network security

Machine learning, especially deep representation learning, on graphs is an emerging field with a wide range of applications. Within this field, graph neural networks (GNNs) have been recently proposed to model and learn over graph-based data representation by generating graph embedding (Sect. 2.1). Due to their unique ability to generalize over

graph data, GNNs are a central technique to apply artificial intelligence techniques to networking security as well as networking applications. A combination between GNNs and machine learning algorithms may provide better results than machine learning algorithms alone or statistical tools.

## 7 Conclusions

In this survey, our aim has been to provide a comprehensive overview of graph-based approaches to network security problems. We surveyed qualitative and quantitative graph-based approaches with special attention to network traffic analysis and botnet detection. The surveyed works were categorized into three groups:

(i) graph-based data models, in which we observed a prevalence of models of network traffic,
(ii) graph-based analysis, in which we observed the emerging topic of graph mining mostly applied to botnet detection, and
(iii) graph features, in which we delved into the features used for botnet detection via machine learning on graphs.

The important message we aimed to highlight is the strength of graphs in capturing network security data, including NetFlow, intrusion detection alerts, and authentication event logs. Graphs are a powerful mechanism for prevention, detection, and investigation in network security. In fact, we highlight that

(i) data are often linked and inter-dependent between heterogeneous sources,
(ii) there are numerous graph models for resolving various problems, and
(iii) the graphs are robust for understanding complex data by capturing interactions and structures.

The goal of this paper was to convey the advantages of graphs and their applications in network security by providing a comprehensive list of available techniques and algorithms that use graphs. Nevertheless, there are open challenges for research and development in the field. Namely, it is up to the security analysts to select the most suitable graph models and algorithms, which might be complicated without expert knowledge. Further, the graph databases and big graph-processing systems are not used at their full potential yet.

**Author Contributions** All authors contributed to the study conception and design. The first draft of the manuscript was written by SL, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript. Here are the details. SL and MH, as experts in network security and machine learning at Fujitsu and Masaryk University, respectively, wrote the main manuscript text and figures. HS, as an expert in graph theory, contributed to and wrote a machine learning and graph theory part with a machine learning point of view. SV, as a cyber security expert at Citibank, provided a security overview by reviewing each step of the writing process. RS, as an expert in network and cybersecurity, reviewed the manuscript text, by providing a cybersecurity and machine learning point of view. MO as an expert and head of cybersecurity at Fujitsu, reviewed the manuscript text by providing a cybersecurity point of view. All authors reviewed the manuscript.

**Funding** For the research leading to these results, Hamida Seba received funding from Agence National de la Recherche (ANR) under Grant Agreement No. ANR-20-CE39-0008, Radu State received funding from Fonds National de la Recherche (FNR) for CAFFE project. Martin Husák was supported by ERDF "CyberSecurity, CyberCrime, and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

**Research data policy and data availability** Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

## Declarations

**Conflict of interest** All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

**Ethical approval** All authors declare that they adhere to the ethical principles of the journal.

## References

1. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Min. Knowl. Disc. **29**(3), 626–688 (2014)
2. Amini, P., Araghizadeh, M.A., Azmi, R.: A survey on botnet: classification, detection and defense. In: International Electronics Symposium (IES), pp. 233–238 (2015)
3. Amrouche, F., Lagraa, S., Kaiafas, G., State, R.: Graph-based malicious login events investigation. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 63–66 (2019)
4. Apache Software Foundation: Apache Spark. https://spark.apache.org/. Accessed 1 Nov 2021
5. Apache Software Foundation: Apache TinkerPop. https://tinkerpop.apache.org/. Accessed 1 Nov 2021
6. Apache Software Foundation: GraphX. https://spark.apache.org/graphx/. Accessed 1 Nov 2021
7. Apruzzese, G., Pierazzi, F., Colajanni, M., Marchetti, M.: Detection and threat prioritization of pivoting attacks in large networks. IEEE Trans. Emerg. Top. Comput. **8**(2), 404–415 (2020)
8. ArrangoDB. https://www.arangodb.com. Accessed 1 Nov 2021
9. Bai, J., Shi, Q., Mu, S.: A malware and variant detection method using function call graph isomorphism. Secur. Commun. Netw. **2019**, 1043,794:1-1043,794:12 (2019)
10. Berger, A., D'Alconzo, A., Gansterer, W.N., Pescapé, A.: Mining agile DNS traffic using graph analysis for cybercrime detection. Comput. Netw. **100**, 28–44 (2016)

11. Böhm, F., Menges, F., Pernul, G.: Graph-based visual analytics for cyber threat intelligence. Cybersecurity **1**(1), 16 (2018)
12. Bou-Harb, E., Debbabi, M., Assi, C.: Big data behavioral analytics meet graph theory: on effective botnet takedowns. IEEE Netw. **31**(1), 18–26 (2017)
13. Bowman, B., Laprade, C., Ji, Y., Huang, H.H.: Detecting lateral movement in enterprise computer networks with unsupervised graph AI. In: 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020), pp. 257–268 (2020)
14. Bowman, B., Huang, H.H.: Towards next-generation cybersecurity with graph AI. SIGOPS Oper. Syst. Rev. **55**(1), 61–67 (2021)
15. Bunke, H., Allerman, G.: Inexact graph matching for structural pattern recognition. Pattern Recognit. Lett. **1**(4), 245–253 (1983)
16. Caswell, B., Foster, J.C., Russell, R., Beale, J., Posluns, J.: Snort 2.0 Intrusion Detection. Syngress Publishing, Oxford (2003)
17. Cayley. https://cayley.io. Accessed 1 Nov 2021
18. Čermák, M., Šrámková, D.: GRANEF: utilization of a graph database for network forensics. In: Proceedings of the 18th International Conference on Security and Cryptography, pp. 785–790. SCITEPRESS (2021)
19. CESNET and Masaryk University: SABU. https://sabu.cesnet.cz/en/start. Accessed 1 Nov 2021
20. Chowdhury, S., Khanzadeh, M., Akula, R., Zhang, F., Zhang, S., Medal, H., Marufuzzaman, M., Bian, L.: Botnet detection using graph-based feature clustering. J. Big Data **4**(1), 14 (2017)
21. CISCO: global—2021 forecast highlights. https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf (2021)
22. Data Collection, C., Sharing. https://www.caida.org/data/. Accessed 1 Nov 2021
23. Daya, A.A., Salahuddin, M.A., Limam, N., Boutaba, R.: A graph-based machine learning approach for bot detection. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 144–152 (2019)
24. Daya, A.A., Salahuddin, M.A., Limam, N., Boutaba, R.: BotChase: graph-based bot detection using machine learning. IEEE Trans. Netw. Serv. Manag. **17**(1), 15–29 (2020)
25. DGraph. https://dgraph.io. Accessed 1 Nov 2021
26. Essawy, B.T., Goodall, J.L., Voce, D., Morsy, M.M., Sadler, J.M., Choi, Y.D., Tarboton, D.G., Malik, T.: A taxonomy for reproducible and replicable research in environmental modelling. Environ. Model. Softw. **134**, 104,753 (2020)
27. Evrard, L., François, J., Colin, J.: Attacker behavior-based metric for security monitoring applied to darknet analysis. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 89–97 (2019)
28. Fitch, J.A., III., Hoffman, L.J.: A shortest path network security model. Comput. Secur. **12**(2), 169–189 (1993). https://doi.org/10.1016/0167-4048(93)90100-J
29. Fredj, O.B.: A realistic graph-based alert correlation system. SEC Commun. Netw. **8**(15), 2477–2493 (2015)
30. Gamachchi, A., Boztas, S.: Insider threat detection through attributed graph clustering. In: IEEE Trustcom/BigDataSE/ICESS, pp. 112–119 (2017)
31. Gamachchi, A., Sun, L., Boztas, S.: Graph based framework for malicious insider threat detection. In: 50th Hawaii International Conference on System Sciences, HICSS, pp. 1–10 (2017)
32. García, S., Grill, M., Stiborek, J., Zunino, A.: An empirical comparison of botnet detection methods. Comput. Secur. **45**, 100–123 (2014)
33. García, S., Zunino, A., Campo, M.: Survey on network-based botnet detection methods. Secur. Commun. Netw. **7**(5), 878–903 (2014)
34. Gligor, V.D.: A note on denial-of-service in operating systems. IEEE Trans. Softw. Eng. **SE–10**(3), 320–324 (1984). https://doi.org/10.1109/TSE.1984.5010241
35. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, pp. 855–864 (2016)
36. Haas, S., Fischer, M.: GAC: graph-based alert correlation for the detection of distributed multi-step attacks. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC '18, pp. 979–988. Association for Computing Machinery (2018)
37. Haas, S., Wilkens, F., Fischer, M.: Efficient attack correlation and identification of attack scenarios based on network-motifs. In: 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC) (2019). https://doi.org/10.1109/IPCCC47392.2019.8958734
38. Haas, S., Fischer, M.: On the alert correlation process for the detection of multi-step attacks and a graph-based realization. SIGAPP Appl. Comput. Rev. **19**(1), 5–19 (2019)
39. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
40. Husák, M., Čermák, M.: A graph-based representation of relations in network security alert sharing platforms. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 891–892 (2017)
41. Husák, M., Komárková, J., Bou-Harb, E., Celeda, P.: Survey of attack projection, prediction, and forecasting in cyber security. IEEE Commun. Surv. Tutor. **21**(1), 640–660 (2019)
42. Jaikumar, P., Kak, A.C.: A graph-theoretic framework for isolating botnets in a network. Secur. Commun. Netw. **8**(16), 2605–2623 (2015)
43. JanusGraph. http://janusgraph.org. Accessed 1 Nov 2021
44. Kaiafas, G., Varisteas, G., Lagraa, S., State, R., Nguyen, C.D., Ries, T., Ourdane, M.: Detecting malicious authentication events trustfully. In: 2018 IEEE/IFIP Network Operations and Management Symposium (NOMS) (2018)
45. Kao, M.Y.: Encyclopedia of Algorithms. Springer, New York (2007)
46. Kaynar, K.: A taxonomy for attack graph generation and usage in network security. J. Inf. Secur. Appl. **29**, 27–56 (2016)
47. Kent, A.D.: Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory (2015). https://doi.org/10.17021/1179829
48. Kiouche, A.E., Lagraa, S., Amrouche, K., Seba, H.: A simple graph embedding for anomaly detection in a stream of heterogeneous labeled graphs. Pattern Recognit. **112**, 107,746 (2021)
49. Lagraa, S., François, J., Lahmadi, A., Minier, M., Hammerschmidt, C.A., State, R.: BotGM: unsupervised graph mining to detect botnets in traffic flows. In: Cyber Security in Networking Conference, CSNet (2017)
50. Lagraa, S., François, J.: Knowledge discovery of port scans from darknet. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 935–940 (2017)
51. Lagraa, S., State, R.: What database do you choose for heterogeneous security log events analysis? In: 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 812–817. IEEE (2021)
52. Lagraa, S., Chen, Y., François, J.: Deep mining port scans from darknet. Int. J. Netw. Manag. **29**(3), e2065 (2019)
53. Lal, M.: Neo4J Graph Data Modeling. Packt Publishing, Birmingham (2015)
54. Lallie, H.S., Debattista, K., Bal, J.: A review of attack graph and attack tree visual syntax in cyber security. Comput. Sci. Rev. **35**, 100,219 (2020)
55. Leichtnam, L., Totel, E., Prigent, N., Mé, L.: Sec2graph: network attack detection based on novelty detection on graph structured

data. In: Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 238–258. Springer (2020)
56. Li, Z., Chen, Q.A., Yang, R., Chen, Y., Ruan, W.: Threat detection and investigation with system-level provenance graphs: a survey. Comput. Secur. **106**, 102,282 (2021)
57. Li, S., Zhou, Q., Zhou, R., Lv, Q.: Intelligent malware detection based on graph convolutional network. J. Supercomput. **78**(3), 4182–4198 (2022)
58. Liu, L., De Vel, O., Han, Q., Zhang, J., Xiang, Y.: Detecting and preventing cyber insider threats: a survey. IEEE Commun. Surv. Tutor. **20**(2), 1397–1417 (2018)
59. Neo4j. https://neo4j.com/. Accessed 1 Nov 2021
60. Neo4j: cypher query language. https://neo4j.com/developer/cypher/. Accessed 1 Nov 2021
61. Newman, M.E.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. USA **103**, 8577–8582 (2006)
62. Noel, S., Harley, E., Tam, K.H., Gyor, G.: Big-Data Architecture for Cyber Attack Graphs Representing Security Relationships in NoSQL Graph Databases (2015)
63. Noel, S., Harley, E., Tam, K.H., Limiero, M., Share, M.: CyGraph: graph-based analytics and visualization for cybersecurity. In: Handbook of Statistics, vol. 35, pp. 117–167. Elsevier (2016)
64. Noel, S.: A Review of Graph Approaches to Network Security Analytics, pp. 300–323. Springer, New York (2018)
65. OrientDB. https://orientdb.org. Accessed 1 Nov 2021
66. Paxson, V.: Bro: a system for detecting network intruders in real-time. Comput. Netw. **31**(23–24), 2435–2463 (1999)
67. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: Online Learning of Social Representations, pp. 701–710. ACM (2014)
68. Quiña Mera, A., Fernandez, P., García, J.M., Ruiz-Cortés, A.: GraphQL: a systematic mapping study. ACM Comput. Surv. **55**(10), 25 (2023). https://doi.org/10.1145/3561818
69. Roussinov, D.G., Chen, H.: A scalable self-organizing map algorithm for textual classification: a neural network approach to thesaurus generation (1998)
70. Sadreazami, H., Mohammadi, A., Asif, A., Plataniotis, K.N.: Distributed-graph-based statistical approach for intrusion detection in cyber-physical systems. IEEE Trans. Signal Inf. Process. Netw. **4**(1), 137–147 (2018)
71. Sanfeliu, A., Fu, K.: A distance measure between attributed relational graphs for pattern recognition. IEEE Trans. Syst. Man Cybern. B **13**(3), 353–363 (1983)
72. SANS Internet Storm Center: DShield. https://secure.dshield.org/. Accessed 1 Nov 2021
73. Shang, Y., Yang, S., Wang, W.: Botnet detection with hybrid analysis on flow based and graph based features of network traffic. In: Cloud Computing and Security, pp. 612–621. Springer (2018)
74. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), pp. 108–116 (2018)
75. Shevchenko, S., Zhdanova, Y., Skladannyi, P., Spasiteleva, S.: Mathematical methods in cybersecurity: graphs and their application in information and cybersecurity. Cybersecur. Educ. Sci. Tech. **1**, 25 (2021). https://doi.org/10.28925/2663-4023.2021.13.133144
76. Sinha, K., Viswanathan, A., Bunn, J.: Tracking temporal evolution of network activity for botnet detection (2019). https://doi.org/10.48550/ARXIV.1908.03443. arXiv:1908.03443
77. Stratosphere Lab: The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background traffic. https://www.stratosphereips.org/datasets-ctu13. Accessed 1 Nov 2021
78. Tiddi, I., Schlobach, S.: Knowledge graphs as tools for explainable machine learning: a survey. Artif. Intell. 103627 (2021)
79. Umer, M.F., Sher, M., Bi, Y.: Flow-based intrusion detection: techniques and challenges. Comput. Secur. **70**, 238–254 (2017)
80. Venkatesh, B., Choudhury, S.H., Nagaraja, S., Balakrishnan, N.: BotSpot: fast graph based identification of structured P2P bots. J. Comput. Virol. Hack. Tech. **11**(4), 247–261 (2015)
81. Wang, J., Paschalidis, I.C.: Botnet detection using social graph analysis. In: 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 393–400 (2014)
82. Wang, J., Paschalidis, I.C.: Botnet detection based on anomaly and community detection. IEEE Trans. Control Netw. Syst. **4**(2), 392–404 (2017)
83. Wang, W., Shang, Y., He, Y., Li, Y., Liu, J.: BotMark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. Inf. Sci. **511**, 284–296 (2020)
84. Wüchner, T., Ochoa, M., Pretschner, A.: Malware detection with quantitative data flow graphs. In: 9th ACM Symposium on Information, Computer and Communications Security, pp. 271–282. ACM (2014)
85. Yang, R.: Adjusting assortativity in complex networks. In: Proceedings of the 2014 ACM Southeast Regional Conference, Kennesaw, GA, USA, pp. 2:1–2:5 (2014)
86. Zeek: Zeek Network Security Monitor tool. https://zeek.org/. Accessed 1 Nov 2021