APIN: Automatic Attack Path Identification in Computer Networks

Eric Ficke and Shouhuai Xu
Department of Computer Science, University of Texas at San Antonio

***Abstract*—Identifying the scope of a network attack can be difficult with limited information about the nature of the attack. Even more difficult is the automation of this process. Because of this, it is important to investigate new methods for mapping and quantifying the threat posed by an attack, in order to prioritize actions during incident response. To this end we propose a framework for automatic attack path identification in computer networks (APIN) by leveraging observable malicious behaviors to quantify the threat score of a set of attacks. Using two academic datasets, experimental results show that APIN is able to quickly reconstruct paths that offer meaningful insight into the nature of multi-step threats on the network, given only reasonable restrictions on network size and structure. These insights would not be possible with only existing tools, such as IDSs, and human analysts would require significant time and expertise to obtain the same findings without APIN's guidance.**

***Index Terms*—Attack Path, Threat Score, Alert Prioritization, Intrusion Detection, Cyber Attack, Incident Response**

I. Introduction

In cyber incident response, the defender needs to identify the location and extent of damage that an attacker has been able to inflict. Since most, if not all, cyber attacks are conducted via multiple steps [1], [2], [3], the *attack paths* that have been exploited by attackers must be identified quickly and with high certainty. Additionally, it is important that the information provided to human defenders has clear cyber security meaning and, if possible, suggests a mitigation approach [4], [5]. For these reasons, this paper aims to identify all potential attack paths with respect to a known or suspected target and rank the most significant attacks based on the magnitude of threat to the network and the causal relationship between attacks.

A. Our Contributions

In this paper we make the following contributions. First, we formulate the problem of cyber attack incident response via the identification of attack paths, which facilitates the *automation* of incident response. To the best of our knowledge, we are the first to formulate the problem from this perspective. Second, in order to rank the significance of attack paths, we introduce a framework which quantifies the threat score of an attack path. Named for its purpose in automatic Attack Path Identification in computer Networks, APIN also introduces the work of quantifying the notion of a *threat* score, as distinguished from a *vulnerability* score, although the two terms have been used interchangeably in some of the literature. The framework utilizes output from existing network monitors such as intrusion detection systems (IDSs) and incorporates existing database software for efficiency. Our approach achieves high *explainability*, *resistance* to some attacks, and can be *automated*. Third, in order to demonstrate the effectiveness of the framework, we conduct case studies on two widely-accepted datasets. Experimental results demonstrate our model's ability to identify paths quickly and accurately, given reasonable restrictions on network size and architecture.

B. Related Works

The problem of identifying attack paths is closely related to the formulation of attack narratives [6] with respect to some cyber attack models (e.g., cyber kill chains [2], [3]) and the formulation of attack stories [7]. Contrasting these studies, we aim to achieve *automated* formulation and reconstruction of attacks that are described in diverse collections of data.

There have been studies on reconstructing Distributed Denial-of-Service (DDoS) attacks (via probabilistic packet marking) [8], malware infections on hosts and command-and-control [9], [10], [11], and network attack paths (via similarities) [12]. Contrasting these studies, we seek to identify and *rank* attack paths, since real networks naturally have many discrete paths of various significance.

Some other works have taken a vulnerability-based approach to analyzing attacks [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]. However, vulnerability information is often incomplete or out of date, since new vulnerabilities are found and published daily [26]. Hoping to avoid this limitation, we instead use a *threat-based* approach, which may be better able to identify previously-unseen threats [27]. The identification of attack paths falls into the broader context of cybersecurity data analytics [28], [29], [30], [31], [32], [33] of the Cybersecurity Dynamics framework [34], [35], [36], [37], [38], [39], [40].

II. Framework

We start with a description of the terminology used in the paper, given in Table I.

A. Problem Statement

We model a computer network as a graph $G = (V, E)$, where V is the set of vertices or nodes (representing IP addresses) E is the set of arcs or directed edges (representing suspicious communications) between nodes. Each edge $e \in E$ consists of a unique tuple $(src, dst, time, SID)$, which identifies a suspicious communication between a source IP address (src) and a destination IP address (dst) at a certain time ($time$) and indicates the type of suspicious activity

Term	Meaning / Usage
Attacker	A computer which behaves maliciously. This does not imply ownership of the system, merely the ability to make it perform some action.
Victim	A computer which has been targeted by an attack. This does not imply compromise of the computer.
Attack Path	Series of computers on a network which begin at some attacker (the origin) and terminate at some victim (the target).
Path Link	A tuple of computers in a network (i.e., an attacker and a victim) which belong to some attack path and which are connected by at least one edge directed to the victim.
ITS	Independent Threat Score corresponding to a specific computer within the network.
CTS	Composite Threat Score corresponding to an attack path and denotes the combined threat against every system in the path.
Origin	Node from which an attacker began executing its strategy (or where this was first observed)
Target	Node which is presumed to be the objective of an attacker's mission. If specified, this node is always at the end of an attack path.

TABLE I: Terms used throughout the paper.

via a signature identifier (SID). SIDs may be provided by intrusion detection systems or other defense tools.

Given a set of security events (e.g., IDS alerts), where an *event* is represented as an edge, these security events can formulate a graph $G = (V, E)$ as follows: vertices are extracted from each event such that both the src and dst represent vertices in the graph. The edge's SID is added to the src vertex's list annotation of $SIDs_{out}$ and the corresponding dst vertex's list annotation of $SIDs_{in}$. The dst address is added to the src vertex's list annotation of $neighbors_{out}$, which contains the set of neighbors against which it has initiated attacks. Likewise, the src vertex is added to the dst vertex's corresponding list annotation of $neighbors_{in}$, which contains the set of neighbors that have initiated attacks against it. Edges are naturally converted directly from alerts because each alert indicates an attack waged from src againt dst.

The *research problem* is to extract and rank the observed attack paths within a network based on the perceived threat of each path. The ranking of these paths should be *generalizeable* so that it can be used in any network without manual tagging and training, and *robust* so that attackers cannot easily force paths to be ranked in the wrong order (i.e., higher ranked attack paths are attacked more heavily).

B. Solution Framework

Figure 1 highlights the proposed framework, known as APIN, per the title of this work. The framework has 3 stages. The first stage uses the alerts provided to construct the graph and quantify the independent threat score (ITS) of each node. If no suspected target is provided, this stage will also select some potential targets or origins based on the ITS. The second stage is to identify possible attack paths with respect to a given node. This stage uses the network communication data from the aforementioned $G = (V, E)$. The third stage aggregates the ITS scores in each path to obtain the composite threat score (CTS). These paths are ranked according to CTS, so the paths with the most significant threat can be handled first.

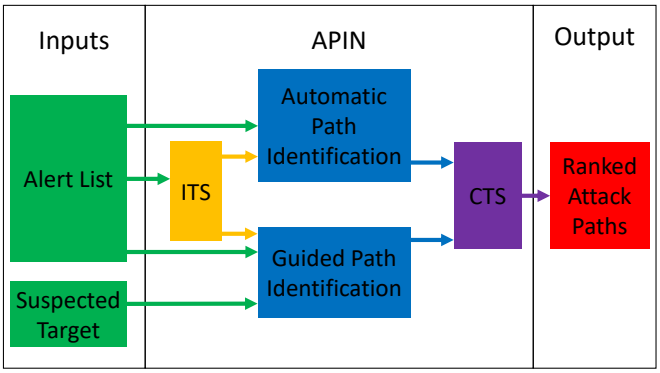

Fig. 1: An overview of the APIN framework. ITS refers to the calculation of the independent threat score. CTS refers to the calculation of composite threat score.

1) Independent Threat Score: In this stage, APIN parses the security events and builds a database each for nodes and edges. Nodes are defined by the IP addresses named as the src or dst of the events. They are annotated with the alert type (i.e., SID) and quantity with which they share an edge, either outbound or inbound, for the src and dst, respectively. Within the database, nodes are indexed according to address, ITS, and number of neighbors in and out. Edges are defined by the tuple described in Section II-A, which is a representation of the corresponding security event. Within the database, edges are indexed according to src, dst and $time$, for each combination of the 3 terms. The combination of all three is indexed twice, to facilitate both a chronological and a reverse-chronological query, as necessary. The 'nodes' and 'edges' databases comprise $G = (V, E)$.

ITS is the first building block we use to analyze attack paths. Intuitively, it measures the scale and cost of attacks against a node. In this case, we interpret cost as the difficulty of generating and launching diverse attacks. This value must be calculated for each computer in an attack path. The parameters used in the ITS formula are chosen as follows:

- Inbound alert diversity (D_{in}). This captures the idea that adaptive attacks by skilled attackers will result in various alarms being triggered. It is calculated as the number of inbound alert types, or 1, if there are no inbound alerts.
- Outbound alert diversity (D_{out}). As with inbound alert diversity, this captures the idea of an adaptive attacker, but is distinguished in the case of server-side attacks, data exfiltration and similar alerts. It is calculated as the number of outbound alert types, or 1, if there are no outbound alerts.
- Inbound alert scale by type (S_{in}). This provides a generic and intuitive measurement of the threat against a node. It is calculated as the geometric mean of the number of inbound alerts of each type present, or 1, if there are no inbound alerts. Alert type is defined by its signature identifier (SID).

Our initial attempt to formulate ITS also used the metric of outbound alert scale, but this was found to produce heavily skewed results with the highest values held exclusively by nodes which had conducted network scans (since these produce an inordinate amount of alerts relative to other attack types). This not only puts the focus on nodes with relatively routine activity (even though scans are indeed suspicious), but enables attackers to easily push nodes in their control to the top of the ITS ranking. Because of these observations, we removed outbound alert scale from the formula, making it resistant to the "noisy network scan diversion." For the same reason, we choose to define S_{in} using the *geometric mean* of the number of inbound alerts of each type present, rather than the *arithmetic mean*. This is because some attacks are cheaper than others (in terms of configuration complexity, time of execution, etc.), so attackers could more easily manipulate the arithmetic mean – inflating the threat score – by producing more cheap attacks (such as scans). Given this discussion, we define ITS as follows:

Definition 1 (ITS): The ITS of a node is the weighted geometric mean of the node's inbound alert diversity, outbound alert diversity, and inbound alert scale by type. Specifically,

$$ITS(x) = \sqrt[W]{D_{in}^{w_1} \cdot D_{out}^{w_2} \cdot S_{in}^{w_3}}, \quad (1)$$

where $W = w_1 + w_2 + w_3$ and each weight is configurable.

For our case studies, we used the default values of 1 for each weight. If no reference node (i.e., suspected target) is provided, the 10 nodes with the highest ITS are selected and used in turn as reference nodes, as both target and origin (i.e., path identification is run 20 times). This means that the use of a reference node can significantly improve the runtime of APIN, but that the reliability of the corresponding results are wholly dependent on the reliability of the reference node.

2) Identifying Attack Paths: In principle, attack paths can be reconstructed with respect to a *time-based* or *node-based* approach. In the time-based approach, the idea is to parse each arc in reverse-chronological order and add newly identified nodes to the DAG as appropriate. This process is repeated for each known or suspected node in V. In the node-based approach, arcs are indexed by their source and destination, then each victim node's adjacency list is parsed significantly faster than in the time-based approach. Indexing needs only to occur once for each arc.

APIN also includes a configurable blacklist, in case some nodes (e.g., honeypots) need not be examined. This is also useful for nodes which have a high cardinality of neighbors (such as routers and broadcast addresses), which can cause an exponential increase in the runtime of the algorithm. This effect imitates that of attempting to identify all possible attack paths in a fully-connected graph, as discussed below. Nodes excluded for this reason should be manually inspected.

Because of its apparent runtime advantage over the time-based approach, we only provide pseudocode for the node-based approach; because the origin-centered algorithm follows the same logical flow in reverse, we only give the target-centered algorithm, namely Algorithm 1. In the pseudocode, we construct a tree of all the nodes which connect to the target, noting the time which they do so. This preserves the temporal dependency between two attacks (i.e., a secure node cannot be used to conduct an attack before that node itself has been attacked). Once the tree has been constructed, the paths extending down to each leaf are the possible paths the attacker could have taken, with the leaves being the corresponding origin.

Algorithm 1 Target-Centered APIN with Node Indexing

Input: Target_Address, $G = (V, E, Z)$, Blacklist
Output: Attack_Paths $= ([V])$

root $\leftarrow$ NewTree
root.$time \leftarrow TIME\_MAX$
New_Leaves $\leftarrow \{root\}$
while New_Leaves has nodes **do**
  Candidates $\leftarrow$ New_Leaves
  New_Leaves $\leftarrow \emptyset$
  for candidate $\in$ Candidates **do**
    Query edges with $dst =$ candidate and $time <$ candidate.$time$
    Sort Query_Result in reverse-chronological order
    for edge $\in$ Query_Result **do**
      if edge.$src \notin$ Blacklist $\cup$ candidate.$ancestors \cup$ candidate.$children$ **then**
        New_Leaf $\leftarrow \{src =$ edge.$src, time =$ edge.$time\}$
        Add New_Leaf to candidate.$children$
        Add New_Leaf to New_Leaves
      end if
    end for
  end for
end while
Paths $\leftarrow \emptyset$
for leaf $\in$ root.$leaves$ **do**
  path $\leftarrow$ leaf.$ancestors$
  Add path to Paths
end for
return Paths

The time complexity of the algorithm is heavily dependent on the connectedness of the graph. The worst-case complexity is for a graph that is fully-connected such that each node has $|V|$ connections to *each* other node, with connections interleaved specially to ensure that, at each branch of the tree, every node has an edge to every other node. In this case, the complexity is $\mathcal{O}(|V|^3)$. Alternatively, the worst case complexity as defined by the number of edges is $\mathcal{O}(|E|^2)$. That is, the worst-case complexity is $\min\{\mathcal{O}(|V|^3), \mathcal{O}(|E|^2)\}$, depending on the density of graph $G = (V, E)$. For sparsely connected graphs, it is likely that many nodes will never enter a given tree, resulting in a significantly better expected runtime. For a reasonably secure network, we expect attacks to be sparse, or at least concentrated around certain attackers or victims.

3) Composite Threat Score: We introduce the concept of CTS to quantify the threat posed against a given path as a whole. The purpose of this definition is to gain insight into the attacker's intent and/or objective. In part, this can be inferred based on the amount of resources directed at individual computers in a path (given by the ITS). Additionally, the length of a path may be a partial indicator of how well-defined the attacker's objective may be (e.g., if their first target is their only target, perhaps the attacker has some inside knowledge about the location of data they seek to compromise). Based on these principles, we define CTS as follows:

Definition 2 (CTS): The CTS of an attack path is the sum of the ITS of all nodes in the given attack path.

Threat score is conceptualized independently of path identification in Figure 2. Specifically, this figure highlights the fact that even though ITS may be calculated for each node individually, CTS requires both the path structure and the ITS of each node in that path.

The CTS calculation is designed to rank the paths containing those computers most targeted by an attacker, even if not all attacks incident to nodes in that path follow the path precisely. For example, an attacker may conduct some attacks from a variety of external nodes (e.g., using spoofing or a botnet). In this case, it is more important to model the threat against the target than to precisely determine which attacker realized the compromise of that node. Because of this, the measurement of ITS for nodes in a path may be impacted by attacks from/to computers not in the path.

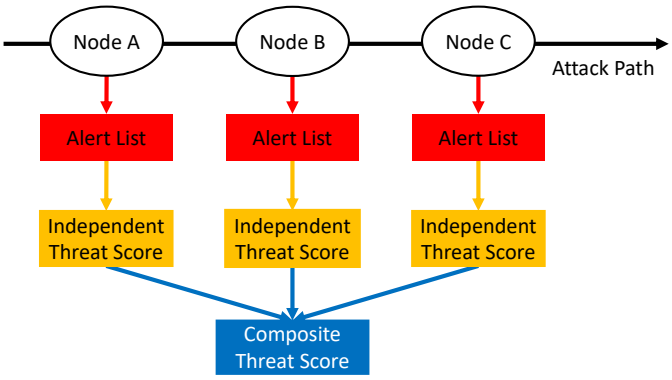

Fig. 2: Threat Score Calculation

III. Evaluation with Real Data

We evaluate our framework using two research-oriented network traffic datasets, DARPA99 [41] and CSECICIDS2018 [42]. The former, despite major criticism and age, remains one of the most-referenced datasets for IDS evaluation today, while the latter offers a significantly improved attack landscape, volume of data, and more robust data description.

A. Configuration

Because the attack paths are derived from the output of an existing IDS (in this case, Suricata 4.0 [43]), it is important that the IDS is properly configured for the network in question. For our experiments, we keep the configuration changes minimal, keeping in mind that our hope is that APIN will be useful to analysts even without advanced expertise and that many enterprise networks already incorporate their own custom configurations. Specifically, we ensure that the $home\_net$ variable is specified according to the architecture given in the datasets, and we download the most updated version of the well-reputed Emerging Threats ruleset [44], with the exception of the policy-based rules, which we exclude because the datasets do not specify what software use policies were implemented during the data collection.

The experiments were processed using an Intel Xeon X5650 (2.67GHz) CPU running ESXi 6.5.0, with two allocated VMS, one for the APIN driver and one for the database. The driver was allocated 2 cores, 32 GB RAM, and 48 GB HDD and ran Ubuntu 18.04.3 LTS Desktop. The database was allocated 4 cores, 16 GB RAM, and 100GB HDD and ran Ubuntu 16.04.2 LTS Server. They were attached to the same virtual subnet.

B. DARPA99 results

The 16,616 alerts from Suricata's output for DARPA99 were converted from text to JSON format in 1.06s. The graph of 431 nodes was constructed and indexed in 8.70s. APIN completed in 0.67s. We consider this runtime to be satisfactory.

Because of the architecture used to collect the DARPA99 data, the results produced by APIN were severely limited. First, the simplicity of many of the attacks limits our ability to identify multi-step attacks because those used in the experiment seem to consist exclusively of a single link, where the attacker stopped the attack once the target was compromised and began a new attack, rather than pivoting between multiple internal nodes. Specifically, all of the paths identified by APIN contained one node in the "172.16.112.0/20" subnet and one node from another subnet or network. In this case, APIN is not useful. Nevertheless, understanding this limitation allows us to draw the following insight.

Insight 1: Research datasets designed to model network attacks should include multi-step attacks.

C. CSECICIDS2018 results

According to the data description provided by its authors, the CSECICIDS2018 dataset contains a network of 450 benign nodes and a separate network of 50 attacker nodes [42]. During processing, we observed an unusually high connectedness for a network of this size. Specifically, we identified 406 nodes that had over 1000 inbound neighbors (i.e., those which produced alerts when processed by Suricata) over the span of data collection, with the highest reaching 5992. This phenomenon seems to be the result of either mass spoofing by the attacker network (which is not clearly detailed in the data description) or of some third-party interference during the experiment. In this particular case, we found that 11 of the 14 documented victims were in the top 406 highly-connected nodes. This complexity prevents the real-time parsing of the graph, which grows exponentially. Nevertheless, we draw some insight from this limitation:

Insight 2: High-granularity network segmentation is important for the efficient analysis of attack paths.

Following this observation, we chose to blacklist the nodes with over 1000 inbound neighbors, preferring a partial result over a prohibitive runtime.

The 3,323,426 alerts from Suricata's output for CSECICIDS2018 were converted from text to JSON format in 3m10s. The graph of 97,873 nodes was constructed and indexed in 29m16s. APIN completed in 42.08s. Given that alert processing and graph construction can proceed during data collection (which spanned a week), we consider this runtime to be satisfactory for the size of the dataset.

Another phenomenon in the APIN output is the frequent occurrence of attempts to probe or exploit the vulnerability known as EternalBlue (MS17-010). This observation was noted for 4 of the top 5 paths and many besides, and was triggered by several different Emerging Threats signatures: [1:2025649:2], [1:2025992:1], [1:2025650:2]. Because the publishers of the dataset do not describe any attacks using EternalBlue or otherwise targeting SMB (the protocol which EternalBlue targets), this supports our previous suspicion that some of the traffic in the dataset was produced by external sources. If this is the case, it may have implications for the validity of ground truth for other experiments (such as training machine-learning based IDSs). From this, we draw the following insight:

Insight 3: Datasets collected from networks with internet access should have strict controls over gateway traffic, on par with those used in production networks.

The highest-ranked path (with a CTS of 34.31) contains five nodes (A-E) across four links. Its detail are below. Alert descriptions have been modified to improved readability, and SIDs have been included for reference. External IP Addresses have been truncated to preserve anonymity.

1) 103.aaa.aaa.aaa (A) to 172.31.67.46 (B)
   - (1:2102465:9) SMB share access
   - (1:2102466:9) SMB unicode share access
   - (1:2025649:2) ETERNALBLUE Probe MS17-010 (MSF style)
   - (1:2025992:1) ETERNALBLUE Probe MS17-010 (Generic Flags)
2) 172.31.67.46 (B) to 103.ccc.ccc.ccc (C)
   - (1:2025650:2) ETERNALBLUE Probe Vulnerable System Response MS17-010
3) 103.ccc.ccc.ccc (C) to 172.31.66.112 (D)
   - (1:2102466:9) SMB unicode share access
   - (1:2102465:9) SMB share access
   - (1:2025649:2) ETERNALBLUE Probe MS17-010 (MSF style)
   - (1:2025992:1) ETERNALBLUE Probe MS17-010 (Generic Flags)
4) 172.31.66.112 (D) to 54.eee.eee.eee (E)
   - (1:2016149:2) Session Traversal Utilities for NAT (STUN Binding Request)

We can clearly see that the attacker probes node B in link 1, which is verified as vulnerable in link 2. This sequence may be indicative of a reflected attack (in which the response calls back to an IP distinct from the one used to initiate the attack), of a two-part attack (in which the attacker probes from node A and launches the full exploit from node C), or of two probes from distinct attackers. The precise meaning of this interaction is not clearly discernible by APIN because of the nature of the model. Specifically, because the model is directional and acyclical, APIN only describes one side of each connection. In any case, this can be manually verified now that APIN has ranked its threat score appropriately. Note that the output given above does not show the temporal relationship between alerts, except in that at least one alert of a given link must precede at least one alert of all following links.

IV. Discussion

The present study has several limitations, which should be addressed in future studies. First, the path identification algorithms are limited by the ability of existing IDSs to identify malicious and anomalous traffic. False-negatives from these devices may prevent APIN from identifying certain links, resulting in paths that are too short. However, because path links require only a single edge to be included, but are ranked according to threat score, False negatives from IDSs may cause negligible harm in the APIN model if there are other correctly-identified attacks along the same link as the missed ones.

Second, false-negatives and false-positives from the IDS(s) used may reduce accuracy in the calculation of ITS for affected nodes (and therefore CTS for affected paths). However, poor accuracy in the source data can be mitigated in part by careful configuration of IDS parameters. During the design and testing of APIN, we found it particularly important to validate policy-based rules, which prior to reconfiguration imposed a false-positive rate of 82.2% in the CSECICIDS2018 dataset.

Third, ITS and CTS are objective standards, but they are subject to certain parameters, such as the timespan of data collection, which must be consistent for the metrics to preserve their meaning between various samples or datasets. A simple solution to this could be to define a specific amount of time for which to consider, but at present such a definition would be arbitrary and unprincipled. We leave this to future work.

Fourth, the above also makes it difficult to share precise intelligence between different organizations. However, since the sharing of this sort of intelligence between different networks is often subject to privacy concerns anyway, and since the attack paths can be abstracted with relative ease (e.g., replacing an IP address with "a DNS server"), we do not feel that this limitation is prohibitive.

V. Conclusion

In this paper, we introduced an empirical approach for modeling multi-step attacks. Our model is based on the concept of threat score, which we quantified in terms of individual threat score (against a single node) and composite threat score (against an attack path). Our model includes algorithms to

identify potential attack paths, including the option to specify a known or suspected target or origin, and alternatively, can automatically identify high-threat nodes and identify paths targeting or originating from them. We have described some reasonable parameters which must be met in the source network, in order for the model to be useful and efficient, and we have demonstrated that given those parameters, the model is capable of identifying significant attacks, including several that had not been identified in the given dataset. The *metrics* we defined in order to quantify threat score are *generalizeable* to many datasets (although not universal to each simultaneously), *resistant* to some manipulation by attackers (although possibly not fully robust), and *explainable* in plain language. The *algorithms* for identification and ranking of attack paths are also *explainable* and can be *automated*.

ACKNOWLEDGEMENTS

We would like to thank Matthew Herrada for his involvement with the formulation of the models used in this paper. This work is supported in part by NSF Grant #1736209 and the NSA OnRamp II program.

REFERENCES

[1] J. Navarro, A. Deruyver, and P. Parrend, "A systematic survey on multi-step attack detection," *Computers & Security*, vol. 76, pp. 214–249, 2018.

[2] Lockheed Martin, "Cyber kill chain." http://cyber.lockheedmartin.com/solutions/cyber-kill-chain, (Accessed July 08, 2016).

[3] Mandiant, "Apt1 report." https://www.fireeye.com/content/dam/fireeyewww/services/pdfs/mandiant-apt1-report.pdf, February 16, 2013 (Accessed July 08, 2016).

[4] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*, pp. 305–316, IEEE, 2010.

[5] E. Ficke, K. M. Schweitzer, R. M. Bateman, and S. Xu, "Analyzing root causes of intrusion detection false-negatives: Methodology and case study," in *Proc. IEEE MILCOM'2019*, 2019.

[6] J. Mireles, J. Cho, and S. Xu, "Extracting attack narratives from traffic datasets," in *Proc. CyCon U.S. 2016*, pp. 118–123, 2016.

[7] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu, "Hercule: Attack story reconstruction via community discovery on correlated log graph," in *Proc. ACSAC'2016*, p. 583–595, 2016.

[8] S. Saurabh and A. S. Sairam, "A more accurate completion condition for attack-graph reconstruction in probabilistic packet marking algorithm," in *Proc. 2013 National Conference on Communications*, pp. 1–5, 2013.

[9] J. Morales, M. Main, W. Luo, S. Xu, and R. Sandhu, "Building malware infection trees," in *Proc. MALWARE*, pp. 50–57, 2011.

[10] A. F. Shosha, J. I. James, and P. Gladyshev, "A novel methodology for malware intrusion attack path reconstruction," in *Lecture Notes in Social Informatics and Telecommunications Engineering*, pp. 131–140, Springer Berlin Heidelberg, 2012.

[11] J. A. Morales, A. Al-Bataineh, S. Xu, and R. S. Sandhu, "Analyzing and exploiting network behaviors of malware," in *SecureComm*, pp. 20–34, 2010.

[12] J. Tian, X. Li, Z. Tian, and W. Qi, "Network attack path reconstruction based on similarity computation," in *Proc. ICNC-FSKD*, pp. 2457–2461, 2017.

[13] P. Mell, K. Scarfone, and S. Romanosky, *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*. NIST and Carnegie Mellon University, 1 ed., June 2007.

[14] S. Xu, X. Li, T. Parker, and X. Wang, "Exploiting trust-based social networks for distributed protection of sensitive data," *IEEE T-IFS*, vol. 6, no. 1, pp. 39–52, 2011.

[15] X. Li, P. Parker, and S. Xu, "A stochastic model for quantitative security analyses of networked systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 1, pp. 28–43, 2011.

[16] S. Xu, W. Lu, and L. Xu, "Push- and pull-based epidemic spreading in networks: Thresholds and deeper insights," *ACM TAAS*, vol. 7, no. 3, 2012.

[17] S. Xu, W. Lu, and . Zhan, "A stochastic model of multivirus dynamics," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 30–45, 2012.

[18] M. Xu and S. Xu, "An extended stochastic model for quantitative security analysis of networked systems," *Internet Mathematics*, vol. 8, no. 3, pp. 288–320, 2012.

[19] W. Lu, S. Xu, and X. Yi, "Optimizing active cyber defense dynamics," in *Proc. GameSec'13*, pp. 206–225, 2013.

[20] S. Xu, W. Lu, L. Xu, and Z. Zhan, "Adaptive epidemic dynamics in networks: Thresholds and control," *ACM TAAS*, vol. 8, no. 4, 2014.

[21] G. Da, M. Xu, and S. Xu, "A new approach to modeling and analyzing security of networked systems," in *Proc. HotSoS'14*, pp. 6:1–6:12, 2014.

[22] F. Leitold, A. Arrott, and K. Hadarics, "Quantifying cyber-threat vulnerability by combining threat intelligence, it infrastructure weakness, and user susceptibility," in *24th Annual EICAR Conference*, 2016.

[23] S. Lee, S. Kim, K. Choi, and T. Shon, "Game theory-based security vulnerability quantification for social internet of things," *Future Generation Computer Systems*, vol. 82, pp. 752–760, 2018.

[24] H. Hu, H. Zhang, and Y. Yang, "Security risk situation quantification method based on threat prediction for multimedia communication network," *Multimedia Tools and Applications*, vol. 77, no. 16, pp. 21693–21723, 2018.

[25] M. Frigault and L. Wang, "Measuring network security using bayesian network-based attack graphs," in *Proc. IEEE ICSAC*, pp. 698–703, 2008.

[26] "CVE." Available from MITRE, 2020.

[27] E. Ficke, K. M. Schweitzer, R. M. Bateman, and S. Xu, "Characterizing the effectiveness of network-based intrusion detection systems," in *IEEE MILCOM'2018*, pp. 76–81, IEEE, 2018.

[28] J. Mireles, E. Ficke, J. Cho, P. Hurley, and S. Xu, "Metrics towards measuring cyber agility," *IEEE T-IFS*, vol. 14, no. 12, pp. 3217–3232, 2019.

[29] Z. Zhan, M. Xu, and S. Xu, "Characterizing honeypot-captured cyber attacks: Statistical framework and case study," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1775–1789, 2013.

[30] Z. Zhan, M. Xu, and S. Xu, "A characterization of cybersecurity posture from network telescope data," in *Proc. InTrust*, pp. 105–126, 2014.

[31] Z. Zhan, M. Xu, and S. Xu, "Predicting cyber attack rates with extreme values," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 8, pp. 1666–1677, 2015.

[32] Y. Chen, Z. Huang, S. Xu, and Y. Lai, "Spatiotemporal patterns and predictability of cyberattacks," *PLoS One*, vol. 10, p. e0124472, 05 2015.

[33] M. Xu, K. M. Schweitzer, R. M. Bateman, and S. Xu, "Modeling and predicting cyber hacking breaches," *IEEE T-IFS*, vol. 13, no. 11, pp. 2856–2871, 2018.

[34] S. Xu, "Cybersecurity dynamics," in *Proc. Symposium on the Science of Security (HotSoS'14)*, pp. 14:1–14:2, 2014.

[35] S. Xu, "Emergent behavior in cybersecurity," in *Proc. HotSoS*, pp. 13:1–13:2, 2014.

[36] S. Xu, "Cybersecurity dynamics: A foundation for the science of cybersecurity," in *Proactive and Dynamic Network Defense* (Z. Lu and C. Wang, eds.), vol. 74, pp. 1–31, 2019.

[37] R. Zheng, W. Lu, and S. Xu, "Preventive and reactive cyber defense dynamics is globally stable," *IEEE TNSE*, vol. 5, no. 2, pp. 156–170, 2018.

[38] H. Chen, J. Cho, and S. Xu, "Quantifying the security effectiveness of firewalls and dmzs," in *Proc. HoTSoS'2018*, pp. 9:1–9:11, 2018.

[39] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, "A survey on systems security metrics," *ACM Comput. Surv.*, vol. 49, pp. 62:1–62:35, Dec. 2016.

[40] H. Chen, J. Cho, and S. Xu, "Quantifying the security effectiveness of network diversity," in *Proc. HoTSoS'2018*, p. 24:1, 2018.

[41] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Comput. Netw.*, vol. 34, pp. 579–595, Oct. 2000.

[42] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, pp. 108–116, 2018.

[43] "Suricata — open source ids / ips / nsm engine." https://suricata-ids.org/download/, Mar 2018.

[44] "Welcome to the emerging threats rule server." https://rules.emergingthreats.net/, 2019.