2018 10th International Conference on Cyber Conflict
CyCon X: Maximising Effects
T. Minárik, R. Jakschis, L. Lindström (Eds.)
2018 © NATO CCD COE Publications, Tallinn

Mission-Focused Cyber Situational Understanding via Graph Analytics

Steven Noel
Cyber Solutions Technical Center
The MITRE Corporation
McLean, Virginia, United States

Paul D. Rowe
Cyber Solutions Technical Center
The MITRE Corporation
McLean, Virginia, United States

Stephen Purdy
Software Engineering Technical Center
The MITRE Corporation
McLean, Virginia, United States

Michael Limiero
Cyber Solutions Technical Center
The MITRE Corporation
McLean, Virginia, United States

Travis Lu
Software Engineering Technical Center
The MITRE Corporation
McLean, Virginia, United States

Will Mathews
Cyber Solutions Technical Center
The MITRE Corporation
McLean, Virginia, United States

Abstract: This paper describes CyGraph, a prototype tool for improving network security posture, maintaining situational understanding in the face of cyberattacks, and focusing on protection of mission-critical assets. CyGraph captures complex relationships among entities in the cyber security domain, along with how mission elements depend on cyberspace assets. Pattern-matching queries traverse the graph of interrelations according to user-specified constraints, yielding focused clusters of high-risk activity from the swarm of complex interrelationships. Analytic queries are expressed in CyGraph Query Language (CyQL), a domain-specific language for expressing graph patterns of interest, which CyGraph translates to the backend native query language. CyGraph automatically infers the structure of its underlying graph model through analysis of the ingested data, which it presents to the user for generating queries in an intuitive way. CyGraph has been experimentally validated in both enterprise and tactical military environments.

Keywords: *common operating picture, situational understanding, mission assurance, graph analytics*

1. INTRODUCTION

Through centuries of experience and modern advances in technology, military commanders can rely on a fairly sophisticated common operating picture (COP) of the kinetic battlespace. However, significant challenges remain for extending the COP to include cyberspace as an operational domain [1]. Such an extended COP is needed for achieving appropriate levels of resilience to attack, maintaining situational awareness and understanding, and providing command and control of cyber (and joint cyber/kinetic) operations [2]. A cyber-extended COP needs to support the analysis of complex interactions among disparate data for decision making.

This paper describes CyGraph, a prototype tool for improving cyber resilience, maintaining situational awareness in the face of cyberattacks, and focusing on protection of mission-critical assets. CyGraph builds rich graph models from various network and host data sources, fusing isolated data and events into a unified model. From this, cyber operators can apply powerful graph queries that uncover multi-step graph reachability from threats to key cyber assets, as well as other patterns of cyber risk. In this way, the tool correlates and prioritizes alerts in the context of vulnerabilities and key assets. CyGraph analytics extract 'needle in haystack' patterns of cyber risk focused on mission assets, with interactive visualization of query results, giving a common operating picture of cyberspace.

Traditional graph formulations with entities (vertices) and relationships (edges) of a single homogeneous type lack the expressiveness required for representing the rich structures involved in analyzing cyber risk. CyGraph employs *property graphs*, i.e., attributed, multi-relational graphs with vertices and edges having arbitrary properties [3]. Property graphs have the power needed for expressing a range of heterogeneous vertex and edge types, which arise from combining data from a variety of sources into a coherent unified cyber security graph model.

Unlike previous graph-based tools that focus on specific analytic use cases against fixed data models, CyGraph employs a schema-free design with a property-graph data model. The specific security data model is defined implicitly, according to how source data are transformed to a property graph. To help analysts more easily work with such complex models, CyGraph automatically infers the underlying data model for a populated graph. It's domain-specific query language provides a simplifying layer of abstraction from the native query language of the graph database implementation.

CyGraph has been tested in military environments, including at the enterprise backbone and tactical command levels. In this paper, for sensitivity reasons, we

describe CyGraph analytics through simulated data; these datasets mimic patterns that we have observed in real data.

2. PREVIOUS WORK

There has been considerable previous work in graph-based approaches to cyber security. For example, a review in 2013 [4] describes hundreds of papers that employ various kinds of graph representations for security, with over 30 categories just for the specific case of modelling network attacks and defenses with acyclic graphs. A more recent study [5] examines over 50 proposed graph-based security models, each having key differences in representation. The state of practice has reached a level of maturity such that various off-the-shelf tools (both commercial and governmental) have emerged for graph analytics in operational environments [6] [7] [8] [9] [10] [11] [12] [13].

The wide range of proposed graph representations address the fundamental issue that classical graph algorithms alone are insufficient for solving analytic problems in cyberspace. Instead, specific data models are needed that capture the structure and semantics of the various kinds of entities and their relationships. But a significant limitation of the current generation of tools is that they have fixed data models, which limits their scope and ability to adapt to changes in operational environments and analytic requirements.

The idea of leveraging graph database technology for cyber security analysis is first explored in 2015 [14]. A proof-of-concept version of the CyGraph tool, which is implemented as a Java-based application running on a single host, is described [15] [16]. The proof-of-concept tool was applied for some security use cases, using simulated data or isolated examples of real operational data [17] [18]. A particular limitation of these preliminary examples is that mission functional dependency relationships are analyzed separately from cyberspace relationships.

Based on our initial success in proving the CyGraph concept, we have developed a more mature and capable CyGraph tool. This advanced prototype is a web-based (JavaScript) client-server application, distributed across three machines (user web browser as GUI, middle-tier intermediary service, and back-end database service). Leveraging this architecture, we have implemented multiple technologies for the CyGraph back-end graph database, including support for Apache Rya [19] within the Big Data Platform (BDP) [20] developed by the US Defense Information Systems Agency (DISA). The advanced CyGraph prototype also integrates with the Elastic Stack [21] (for Neo4j) or Accumulo [22](for DISA BDP) for scalable data ingest and

intermediate storage. A high-level overview of this tool architecture is described in [2], although no specific analytic results are given there.

The new web-based CyGraph tool has been validated using real data in operational network environments, at enterprise-level scale. The analytic examples that we describe in this paper are abstracted versions of the kinds of results we obtained for real data (abstracted here to protect the sensitive real data). This includes the development and validation of joint models for cyberspace and mission functions, e.g., for showing mission risk and/or impact as we describe. The present work also experimentally validates that CyGraph's loosely-coupled client-server architecture can support multiple back-end graph persistence technologies, while insulating the front-end functionality from the choice of back-end implementation. This in turn provides flexibility in matching the analytic architecture to the performance and scaling requirements for a given organization.

3. CYGRAPH MODEL

We begin by defining the formal structures that form the basis of an instance of a CyGraph model. A *graph* $G = (N, E)$ is a pair of sets of nodes and edges. The edges are, themselves, ordered pairs of nodes (n_1, n_2) from N. A *property graph* is a graph in which the nodes and edges come equipped with attributes, that is, arbitrary key/value pairs describing properties of the elements. We generally assume that nodes and edges have some minimal structure. Namely, nodes have attributes for a unique identifier and a type. Edges also have an attribute describing their type. They additionally have attributes identifying their source and destination nodes. Additional attributes may include such things as location information, mission criticality, or traffic packet counts.

A CyGraph model instance is defined by the properties attributed to the nodes and edges, as well as any constraints that may be in effect. Typically, particular property graphs conforming to a CyGraph model instance are progressively built from heterogeneous data sources with records containing information about the nodes and edges. Rather than requiring a fixed schema for the data sources, CyGraph applies data transformations that map elements of the source data to nodes, edges, and their properties. Thus, these data transformations implicitly define an instantiated CyGraph model.

To better understand how a property graph is built, consider the process of reading in a record r from a data source. Assume the graph built so far is $G = (N, E)$, and the transform T extracts information about two nodes, n_1 and n_2, and an edge e between them. The new graph is $G' = (N \cup \{n_1, n_2\}, E \cup \{e\})$, where the properties of n_1, n_2,

and e are defined by the transform T. If n_1 or n_2 was already in N then we simply update their properties according to any extra information contained in record r.

In general, any property (of nodes or edges) that has potential analytic value (in the sense of constraining graph queries) can be included as a node or edge property. The type for a node or edge can then be defined as an arbitrary function of its properties. Thus, the node and edge types depend on the source data, via the transformation to a CyGraph property graph.

For example, alerts from Host Based Security System (HBSS) [23] yield node types describing the category of the alert for the destination node, e.g., whether they are reconnaissance events (such as port scans) or represent actual host compromise. Network flow records yield the region in which the node is located (US, non-US, country of concern) or indicate that the node is key terrain (based on knowledge of services hosted there and mission dependencies). This transform prioritizes type information from HBSS alerts over the other two, so that if a host is in the US and is compromised, it is simply identified as compromised. One could easily define a different transform that extracts a type in the Cartesian product of the types defined by each data source. The choice of transform depends on the sort of questions one wants answered regarding the graph (i.e., the analytic queries).

Once CyGraph has constructed the property graph from its data sources, an analyst can explore the graph with *queries* expressed in CyGraph Query Language (CyQL), a domain-specific query language. An important aspect of graph structure pertains to reachability. CyQL allows for the specification of structural features of trajectories through a graph. When a query Q is applied to a graph G it results in a (possibly empty) subgraph $G' \subseteq G$. This matching subgraph is then displayed in the user interface.

A *directed trajectory* is an alternating sequence of nodes and edges (n_0, e_1, n_1,…, e_k, n_k) in which, for every $0 < i \leq k$, the source of e_i is n_{i-1} and the destination is n_i. An undirected trajectory is similar, except for any edge e_i, its source and destination may be n_i and n_{i-1} respectively. The length of a trajectory is the number of edges. The *graph of a trajectory* is ($\{n_0,\ldots,n_k\}$, $\{e_1,\ldots,e_k\}$) in which the sequence information has been forgotten. A trajectory t is a trajectory of graph $G = (N, E)$, if the trajectory's graph (N', E') is a subgraph of G (i.e. $N' \subseteq N$ and $E' \subseteq E$).

CyQL specifies trajectories by constraining the number of hops, and the types of the initial node, the end node, and the edges. Queries are built from the following clauses with their associated semantics:

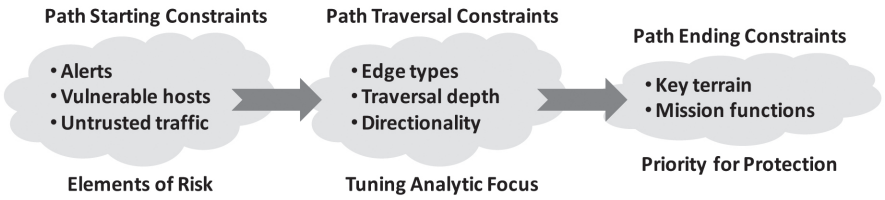
- **`hops($numHops)`**: A trajectory satisfies this clause if its length is **`$numHops`**.
- **`hops($minHops,$maxHops)`**: A trajectory satisfies this clause if its length is between **`$minHops`** and **`$maxHops`** (inclusive).
- **`startType($type)`**: Trajectory (n_0, e_1,…,n_k) satisfies this clause if n_0 is of type **`$type`**.
- **`endType($type)`**: Trajectory (n_0, e_1,…,n_k) satisfies this clause if n_k is of type **`$type`**.
- **`startId($id)`**: Trajectory (n_0, e_1,…,n_k) satisfies this clause if the unique node identifier $u(n_0)$ of node n_0 is equivalent to **`$id`**.
- **`endId($id)`**: Trajectory (n_0, e_1,…,n_k) satisfies this clause if the unique node identifier $u(n_k)$ of node n_k is equivalent to **`$id`**.
- **`edgeTypes($types)`**: A trajectory satisfies this clause if each edge is of one of the types in the comma separated list **`$types`**.
- **`undirected()`**: By default, satisfying trajectories must be directed. When this clause is used, undirected trajectories also satisfy the query.

A CyQL clause is a concatenated sequence of such clauses. A trajectory t satisfies a CyQL query Q (written $t \mid = Q$) if it satisfies all of the clauses. The result of applying Q to graph G is simply the union of all trajectories of G that satisfy Q. That is:

$$Q(G) = \{t \in \text{trajectories of } G\text{: } t \mid = Q\}.$$

CyQL provides a key aspect of risk analysis in CyGraph. In terms of the semantics of attack paths, query trajectory through the property graph corresponds to multi-step attack (or attack reachability) through the network. Conceptually, the aspects of CyQL can be organized as shown in Figure 1.

FIGURE 1. GRAPH TRAJECTORY PATH CONSTRAINTS IN CYGRAPH QUERY LANGUAGE (CYQL)



The left side of Figure 1 represents elements of risk within a network; i.e., things that we are protecting against. By specifying such risk elements as constraints on the starting points of a query traversal, trajectories represent ‘downstream’ relationships

emanating from risk points. Conversely, the right side represents high-valued assets within the environment; i.e., things that we are trying to protect. Defining those things as constraints on the traversal ending points cause paths to be focused on those assets as reachable from the risky elements. CyQL clauses that occur between these starting and ending extremes generally serve to constrain path trajectories in particular ways that help tune analytic focus; e.g., for managing the trade-off between comprehensiveness of query results and cognitive overload.

CyQL queries involve identifying trajectories that start from nodes representing risk elements, and end in nodes representing priorities for protection. The set of trajectories can be further refined by specifying additional traversal constraints regarding the edge types or total path length. This serves to focus an analyst's attention on the relationships that matter the most. By visualizing the results of CyQL queries, CyGraph allows users to quickly identify known risky patterns or anomalous structures that warrant further investigation.

For example, given the appropriate data sources, CyQL makes it straightforward to identify the set of hosts with vulnerabilities that reside within the same connected component as a key cyber asset. By limiting the query to vulnerable hosts within two hops of key cyber assets, one can more easily identify the vulnerable hosts that pose the greatest risks. Queries may also help to identify clusters within the graph that have interesting properties. A highly connected cluster of hosts with host-based alerts may be an indication of vigorous adversarial exploration and exploitation.

4. CYGRAPH ARCHITECTURE

CyGraph ingests data from various sources and normalizes it. It then transforms the elements of the normalized model into a graph model specific to the cyber security domain. Graph queries are issued from the client front end (translated from CyQL to native query language in a middle-tier service) and then executed on the backend database. The resulting query matches are then visualized in the web client (browser).

In this agile architecture, the graph model is defined by how the data sources are transformed into a property graph, rather than conforming to a predetermined schema. Model extensions are simply the creation of additional of nodes, relationships, and properties in the property graph model, and require no schema changes or other database renormalizing. CyGraph supports two options for backend data storage and query processing:

- Neo4j graph database [24] with normalized data in Elasticsearch [21].
- Apache Rya [19] RDF store with normalized data in Apache Accumulo [22].

Each of these options is available as open-source software, and (with the exception of Rya) have commercial support available. The second option (Rya+Accumulo) is available as part of DISA's Big Data Platform (BDP) [20].

In the CyGraph front-end analyst dashboard, graph pattern-matching queries are expressed in CyQL, which CyGraph compiles to Cypher [25] (for Neo4j) or SPARQL [26] (for Rya). This presents a simplifying layer of abstraction, designed specifically for the desired risk analysis, freeing the analyst from learning a complex general-purpose query language.

Typical inputs to CyGraph fall under four categories:

1. Network Infrastructure. This captures the configuration and policy aspects of the network environment.
2. Security Posture. Specification of network infrastructure is combined with vulnerability data to map potential attack paths through the network.
3. Cyber Threats. This captures events and indicators of actual cyberattacks, which are correlated with security posture to provide context for risk analysis and attack response.
4. Mission Dependencies. This captures how elements of enterprise missions depend on cyber assets.

CyGraph relies on other tools and data sources to build its cyber security graphs. For example, the TVA/Cauldron tool [6] [7] [8] [9] can build network attack graphs from host vulnerabilities, firewall rules and network topology. CyGraph can ingest data for both potential and actual threats, including Splunk [27], Wireshark [28], the National Vulnerability Database (NVD) [29], and Common Attack Pattern Enumeration and Classification (CAPEC) [30]. For capturing mission dependencies on cyber assets [17] [18], CyGraph ingests models developed through other tools [16], including Crown Jewels Analysis (CJA) [31] and Cyber Command System (CyCS) [32].

The CyGraph implementation is schema-free, so the model is decoupled from the storage implementation. The particular way in which the data is transformed to a property graph determines an instantiated CyGraph model. So, for example, not all of the data sources in the four categories listed above are necessarily needed for useful analysis – often only a single data source is ingested.

Data is continually streaming in that must be analysed for cyber risk correlation and prioritization by CyGraph. Leveraging the open source Elastic Stack, the Beats platform provides agents for gathering data, with Logstash for transformation and ingest into Elasticsearch. A CyGraph web service then creates a property graph model and imports it into the CyGraph graph data base (Neo4j).

There is a similar analytic flow for CyGraph deployment on BDP, in which data streams are processed by Apache Storm [33], stored in Accumulo [22] and queried in Rya [19]. In this analytic flow, the CyGraph data model is mapped to RDF. The result of a query is a combination of alerts, network flows and vulnerabilities represented as graph nodes and edges. The matching subgraphs for queries are typically orders of magnitude smaller than the full graph stored in Neo4j or Rya.

5. CYGRAPH OPERATION

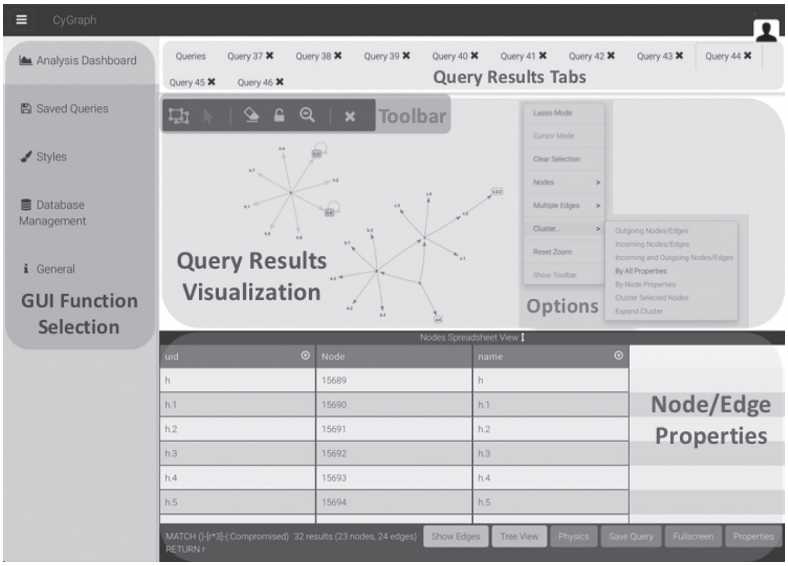
After ingesting data from various sources, CyGraph maps the data to a property graph stored in a graph database. It automatically infers the underlying graph model through inspection of the graph database. It then presents the model to the user in the browser user interface as an interactive graph visualization.

The analyst can interact with this graph model to generate queries in the domain-specific CyQL query language. In particular, user-selected combinations of edge types (diamonds) populate the CyQL **`edgeTypes($types)`** clause, which specifies edge types to be matched in a query. For example, edges of type **`IN`** define relationships between **`Machine`** nodes and **`Domain`** nodes, i.e., network machine membership in protection domains (e.g., subnets) [14].

Core clauses in CyQL define patterns of reachability through a graph, i.e., **`hops($numHops)`**, **`hops($minHops,$maxHops)`**, **`startType($type)`**, **`endType($type)`**, **`startId($id)`**, **`endId($id)`**, **`edgeTypes($type)`**, and **`undirected()`**. CyQL includes other features for matching patterns in the cyber security domain [15], including keywords for host names, IP addresses, subnet address ranges, arbitrary Boolean combinations of clauses and wildcards in parameter values. CyGraph queries are stored for sharing and reuse.

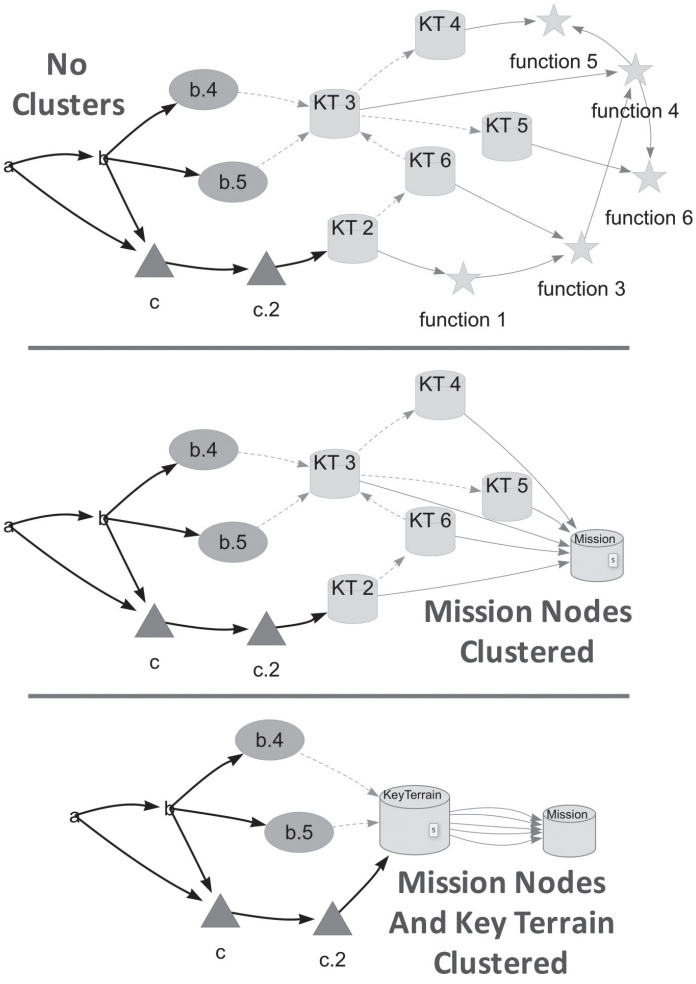
Once a query is executed, CyGraph displays the query results, as shown in Figure 2. Each query submission creates a new query pane, with tabs for selecting panes. The query results (matched subgraph) are visualized in a main panel. Optionally, the properties for selected nodes or edges are displayed below the graph visualization.

FIGURE 2. CYGRAPH WEB USER INTERFACE (QUERY RESULTS)



One of the user-interface options is to cluster elements of the visualized graph in particular ways, i.e., by user-selected nodes, incoming or outgoing edges for a node, or by node type. This is illustrated in Figure 3.

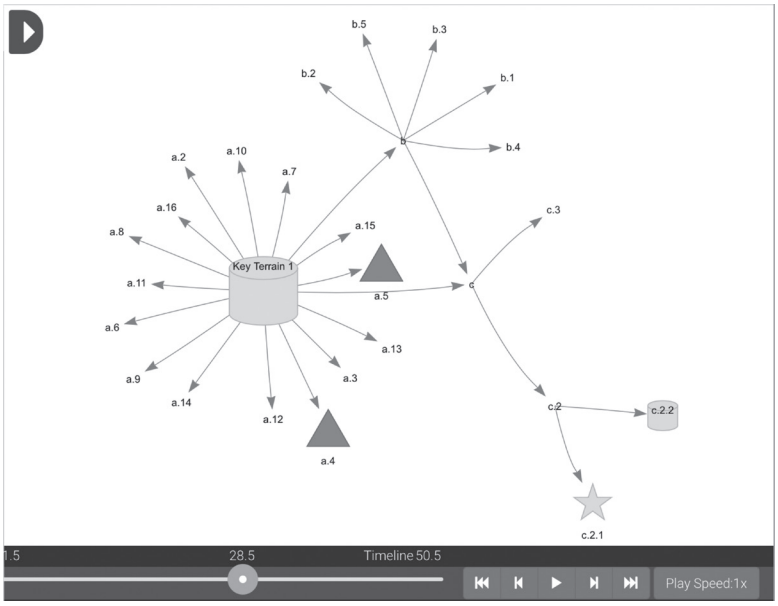
FIGURE 3. CLUSTERING NODES IN GRAPH QUERY VISUALIZATION



The top of Figure 3 is a query result, before clustering is applied. In the middle, clustering is applied via a node property denoting mission functions. At the bottom of the figure, additional clustering is applied, based on a node property denoting key terrain. Visually, such a clustering merges a set of nodes to a single one, with adjacent edges to other (non-clustered) nodes preserved. This kind of interactive visual clustering helps manage the complexity of graph analytics in CyGraph. For example, in Figure 3, clustering the mission and key terrain nodes helps focus attention on the alert destinations (triangles) and vulnerable hosts (ellipses) that are potential risks.

For time-varying models, CyGraph can dynamically visualize the evolving graph state. This is shown in Figure 4. This capability depends on time stamps being defined for edges during the ingestion process. Then, when a query result has a time defined for each edge, the user interface enables the timeline feature. This feature builds a time tick for each discrete event (unique value of time in the query result edge set). The timeline then provides video controls (e.g., play, single step forward/back, speed) for displaying the graph as edges appear over time.

FIGURE 4. INTERACTIVE TIMELINE FOR VISUALIZING GRAPH EVOLUTION OVER TIME

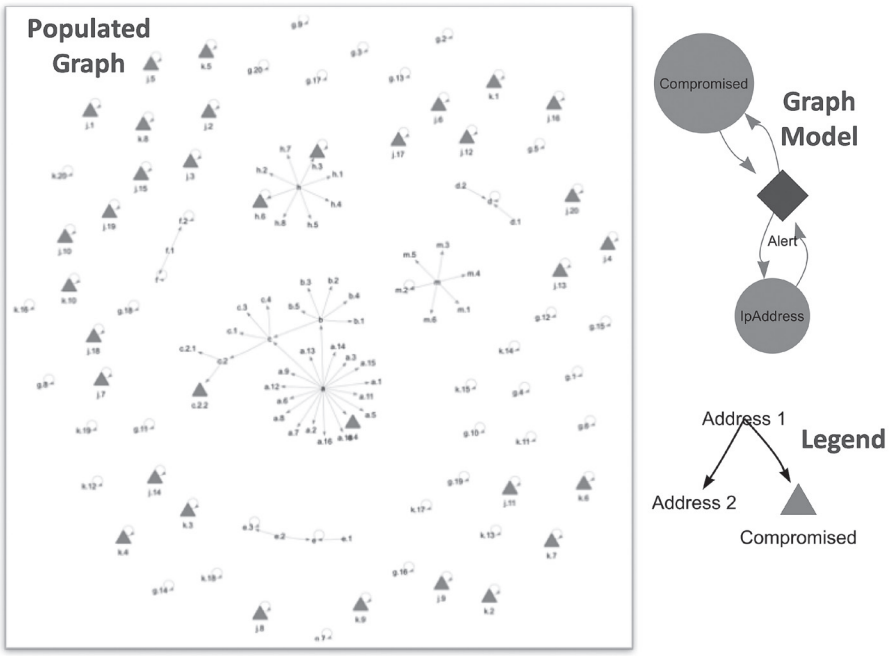


6. EXAMPLE CYGRAPH ANALYTICS

In this section, we describe a number of example applications of CyGraph for security analytics. These examples all use simulated data sets (thus avoiding sensitivity issues), which are designed to mimic patterns that we have observed in real datasets.

The first example (Figure 5) is based on intrusion detection alerts. CyGraph automatically infers the model (top left of the figure) from the populated graph. Nodes are typed as either **`Compromised`** (for destinations of alerts reporting compromise) or **`IpAddress`** (sources/destinations for other general kinds of alerts), rendered as in the legend. An edge is one or more alerts from source to destination.

FIGURE 5. GRAPH MODEL BASED ON INTRUSION ALERTS

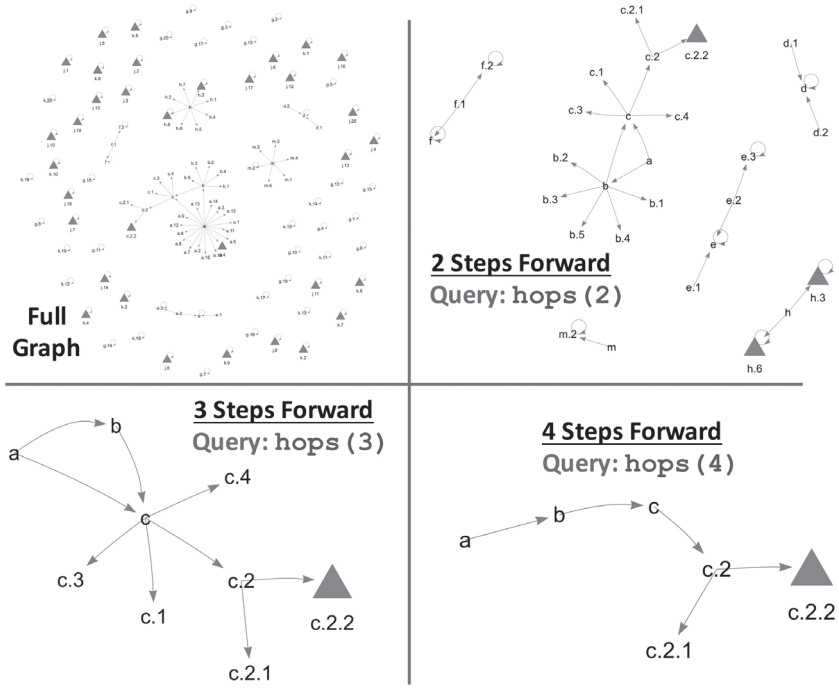


One use for risk analysis is identifying attack reachability in a particular direction, consistent with adversary lateral movement. This pattern is expressed in the CyQL query language via the **`hops($numHops)`** clause, as examined in Figure 6.

The upper left of Figure 6 shows the results for the full (unconstrained) query. The other portions of the figure show query results for **`hops(2)`** (upper right), **`hops(3)`** (lower left), and **`hops(4)`** (lower right). Queries with larger values of **`$numHops`** are more tightly constrained, in the sense of matching deeper traversal. Smaller (more loosely constrained) values of $numHops yield larger matching subgraphs.

Operationally, an analyst can adjust trajectory depth according to analytic need. One can begin with a larger value of **`$numHops`** to discover deep network infiltration as a higher-priority incident. Then, as deeper-level (and more rarely occurring) incidents are resolved, more shallow ones can be investigated. For example, an organization can set the trajectory depth such that there are available resources available to investigate the resulting graph query match.

FIGURE 6. GRAPH QUERY RESULTS FOR DIFFERENT TRAJECTORY DEPTHS



Clauses in CyGraph can be combined for further constraining query results. Semantically, this is a conjunction (Boolean AND), in the sense that conditions in all clauses must match in the query results. This is examined in Figure 7. Here, we combine the **`hops()`** clause with **`endType()`**, which constrains matching paths to end with a node of type **`Compromised`**.

As a use case for operational security, this example focuses on a more severe intrusion alert category as the locus of potential lateral movement by an adversary. Comparing the upper left of Figure 7 (no **`endType`** constraint) with the upper right of Figure 6 (with **`endType`** constraint), we see the result of constraining the **`endType`** (for trajectory depth 2). The query result is much smaller, with all trajectories ending at nodes of type **`Compromised`**. In terms of security analysis, this focuses on paths leading to (reportedly) compromised hosts, e.g., for investigating events leading up to those in question. We see the same kind of result for Figure 7 (upper right) versus Figure 6 (lower left), this time with a trajectory depth of 3. For alert response, this is tracing the investigation deeper into the potential attack.

We now consider a more complex CyGraph example, shown in Figure 8. Like real-world data, such an unconstrained graph visualization is difficult to understand in its entirety. This underscores the need for CyGraph to extract 'needle in haystack' patterns of cyber risk, focused on mission protection.

FIGURE 7. MULTIPLE CLAUSES IN QUERIES

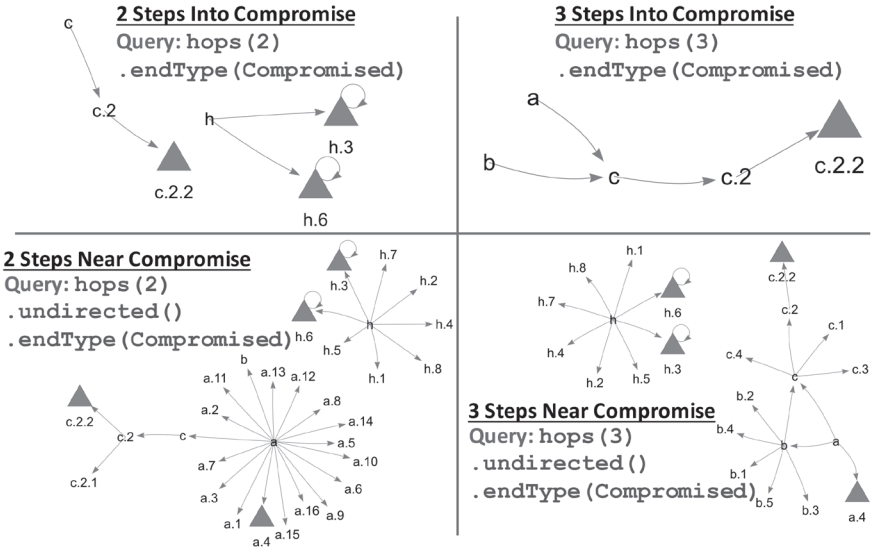
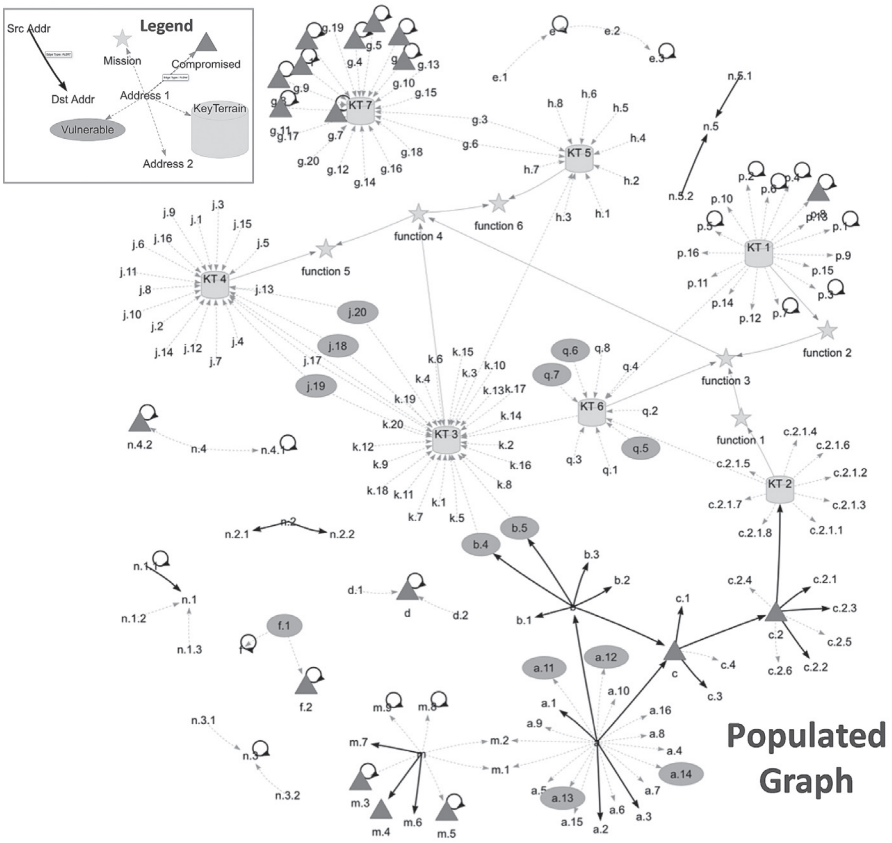
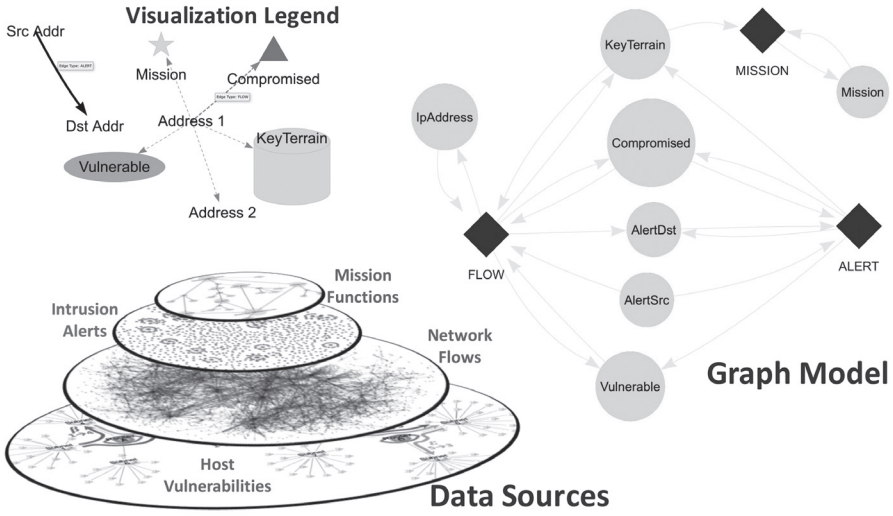


FIGURE 8. GRAPH POPULATED FROM MISSION FUNCTIONS, INTRUSION ALERTS, NETWORK FLOWS, AND HOST VULNERABILITIES



The graph in Figure 8 is populated via a process that transforms host vulnerabilities, network flows, intrusion alerts and mission functional dependencies (i.e., the data sources in Figure 9) to a property-graph model. CyGraph automatically infers the model from the populated graph database, which is the right side of Figure 9.

FIGURE 9. GRAPH VISUALIZATION LEGEND, DATA SOURCES, AND GRAPH MODEL FOR FIGURE 8



In this model, mission nodes are connected to one another (and to key cyber terrain) in terms of their dependencies (from 'provides' to 'needs'). Alert edges connect source and destination nodes of various types: key terrain, compromised (assumed vulnerable), vulnerable (not compromised), and other general alerts sources and destinations. General addresses observed in network flows which are not associated with alerts are connected to each other and to alert addresses via flow edges. In this way, network flows serve to fill in potential gaps from adversary activity not detected by intrusion detection (false negatives).

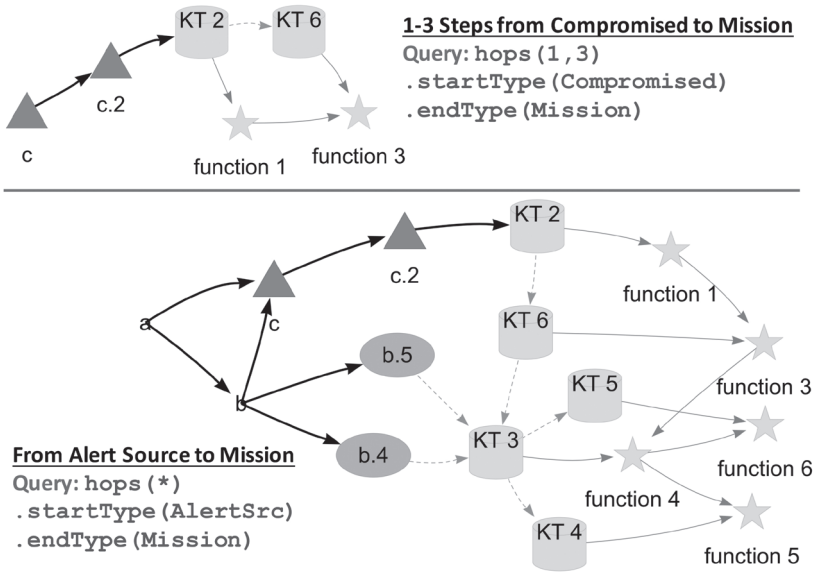
We now apply queries to the graph in Figure 8, in which various combinations of CyQL clauses match subgraphs of interest for analyzing this richer security model. These query clauses generally follow the pattern of constraining paths to start at risky elements and end at high-value mission elements, with intermediate constraints that tune analytic focus. Our examples here also demonstrate another kind of strategy for operational security – tightly constraining queries to initially focus on riskier patterns, then subsequently relaxing constraints to uncover new patterns of the next higher priority.

The top of Figure 10 is the query result for a significantly risky pattern – reported compromises that lead to mission functions within three steps. This query result shows that a compromised node is the source of another alert whose destination is key cyber terrain which supports a mission function. There is also traffic flow (dashed arrow) to

another key terrain node that supports a mission function. The traffic from *KT 2* to *KT 6* might warrant deeper inspection for potential missed detections (false negatives).

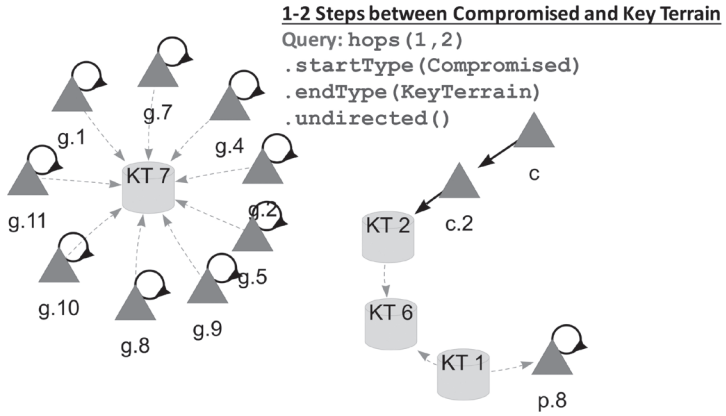
In the bottom of Figure 10, query constraints are relaxed somewhat to expand the analytic scope. In particular, the unlimited trajectory depth via **`hops(*)`** admits paths of any depth leading to mission nodes, and **`startType(AlertSrc)`** has paths starting at alert sources (any severity of alert) rather than compromised destinations. This query result shows additional alert trajectories (all starting from node a), including ones that end on vulnerable hosts, which have traffic to other key terrain supporting other mission functions.

FIGURE 10. RISKY PATHS TO MISSION FUNCTIONS



Next, we apply the **`undirected()`** clause of CyQL, which explores nearness by ignoring path directionality. This is shown in Figure 11. Here, we again apply **`startType(Compromised)`**, along with **`endType(KeyTerrain)`**, which stops at key terrain rather than going beyond to mission functions that depend on them. We also apply a more constrained **`hops(1,2)`** that admits only paths of depths one or two.

FIGURE 11. IGNORING DIRECTIONALITY IN QUERIES

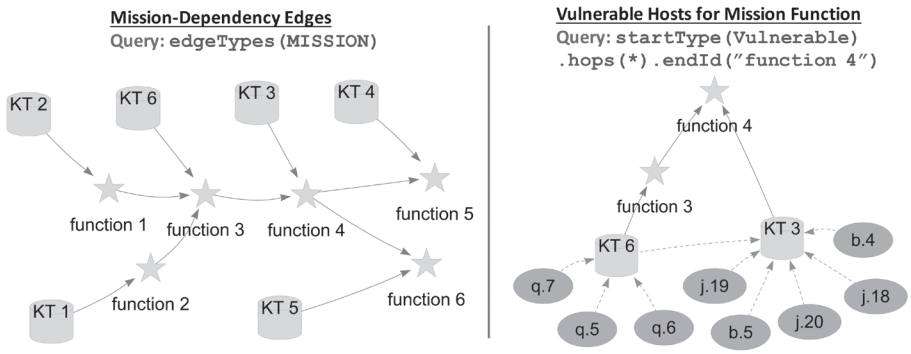


This query finds compromise-related paths in addition to those in Figure 10. This includes nine compromised nodes that communicate with key terrain *KT 7*, which we find by having the query end at key terrain rather than mission functions. As we show in Figure 12, key terrain *KT 7* does not have a known mission function that it supports, so this query identifies risk to such nodes. The query in Figure 11 also finds compromised node *p.8*, which communicates with *KT 1*. In this case, the network flow has *KT 1* as the source (e.g., the initiator of the flow). By ignoring directionality, this admits the possibility of general communication types, e.g., involving attacks against client-side vulnerabilities.

The left side of Figure 12 shows all mission dependencies in this graph model. Mission dependencies are represented as edges of type **`MISSION`**, between key terrain or mission functions, oriented from 'provides' to 'needs.' Thus, the CyQL clause **`edgeType(MISSION)`**, with no other query conditions, finds all such dependencies. Formally, because there is no **`hops`** clause, the query result is the union of edges rather than path trajectories.

The right side of Figure 12 finds all vulnerable hosts that are relevant to a particular mission function. The clause **`startType(Vulnerable)`** causes paths to start at vulnerable nodes. The clause **`endId('function 4')`** causes paths to end at *function 4*. The **`hops(*)`** allows paths of any depth.

FIGURE 12. ALL MISSION DEPENDENCIES (LEFT) AND VULNERABILITIES FOR A PARTICULAR MISSION FUNCTION (RIGHT)



7. SUMMARY AND CONCLUSIONS

Maintaining situational understanding and a common operating picture in cyberspace requires making sense of complex relationships among aspects as varied as security posture, cyber threats, security alerts, and mission dependencies on cyber assets. The volume and complexity of data needed for security operations are far too large for manual inspection or analysis. These challenges multiply with the need to go beyond considering isolated events, matching single-step rules, or generating summary statistics, which yield limited insight into complex adversary actions.

CyGraph creates a unified multi-relational graph model of cyber terrain, events, and mission dependencies. This rich repository of relationships among cyberspace and mission elements supports advanced analytic and visual capabilities. Through pattern-matching queries, CyGraph discovers clusters of high-risk activity from the swarm of complex interrelationships. This allows cyber operators to more easily understand evolving cyberattack situations, and to recommend best courses of action to commanders. By including mission dependencies on cyber assets, CyGraph shows how cyberspace activities influence mission success.

In CyGraph, domain-specific graph queries extract nuggets of important patterns from the swarm of data through query clauses that fine-tune graph path trajectories during query matching. These queries uncover multi-step graph reachability from vulnerabilities and threats to key cyber assets and mission functions. The domain-specific language provides a layer of abstraction that simplifies the operational burden. CyGraph also infers the underlying data model from a populated graph database, presenting that to the analyst to further aid in formulating queries.

CyGraph has a schema-free data model for flexibility in combining various types of relationships, aimed at addressing a wide variety of analytical questions. It is implemented as a 3-tier client-server web application with a graph database or triple store backend and interactive graphical interface in the browser. CyGraph employs a combination of powerful graph-based queries and advanced interactive visualization. It thus provides a significant capability to enable the storage and processing of diverse, mission-relevant cyber data at scale while making the technology readily accessible to cyber analysts. This in turn enables more accurate and rapid decision making for command and control.

CyGraph includes a number of custom capabilities for interactively visualizing and navigating graph query results. This includes clustering nodes according to various criteria, and dynamic rendering of time-varying graph evolution. Overall, these analytic and visual capabilities enable the discovery and understanding of 'needle in haystack' patterns of cyber risk focused on mission assets.

REFERENCES

[1] G. Conti, J. Nelson, and D. Raymond, 'Towards a Cyber Common Operating Picture,' in *5th NATO International Conference on Cyber Conflict*, Tallinn, Estonia, 2013.

[2] S. Noel, D. Bodeau, and R. McQuaid, 'Big-Data Graph Knowledge Bases for Cyber Resilience,' in *NATO IST-153 Workshop on Cyber Resilience*, Munich, Germany, 2017.

[3] M. Rodriguez and J. Shinavier, 'Exposing Multi-Relational Networks to Single-Relational Network Analysis Algorithms,' *Journal of Informetrics*, vol. 4, no. 1, pp. 29-41, 2009.

[4] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, 'DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees,' *Computer Science Review*, Vols. 13-14, 2014.

[5] H. S. Lallie, K. Debattista, and J. Bal, 'An Empirical Evaluation of the Effectiveness of Attack Graphs and Fault Trees in Cyber-Attack Perception,' *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, 2017.

[6] S. O'Hare, S. Noel, and K. Prole, 'A Graph-Theoretic Visualization Approach to Network Risk Analysis,' in *IEEE Workshop on Visualization for Computer Security*, Cambridge, MA, 2008.

[7] S. Jajodia, S. Noel, P. Kalapa, B. O'Berry, M. Jacobs, E. Robertson, and R. Weierbach, 'Network Attack Modeling, Analysis, and Response'. US Patent 7,904,962, 8 March 2011.

[8] S. Noel and S. Jajodia, 'Attack Graph Aggregation'. US Patent 7,904,962, 1 December 2009.

[9] S. Noel and S. Jajodia, 'Metrics Suite for Network Attack Graph Analytics,' in *9th Annual Cyber and Information Security Research Conference (CISRC)*, Oak Ridge National Laboratory, Tennessee, 2014.

[10] K. Ingols, R. Lippmann, and K. Piwowarski, 'Practical Attack Graph Generation for Network Defense,' in *Annual Computer Security Applications Conference*, 2006.

[11] RedSeal Networks, [Online]. Available: http://www.redsealnetworks.com/. [Accessed 18 February 2018].

[12] Skybox Security, [Online]. Available: http://www.skyboxsecurity.com/. [Accessed 18 February 2018].

[13] Sqrrl, [Online]. Available: https://sqrrl.com. [Accessed 18 February 2018].

[14] S. Noel, E. Harley, K. H. Tam, and G. Gyor, 'Big-Data Architecture for Cyber Attack Graphs: Representing Security Relationships in NoSQL Graph Databases,' in *IEEE Symposium on Technologies for Homeland Security (HST)*, Boston, Massachusetts, 2015.

[15] S. Noel, E. Harley, K. H. Tam, M. Limiero, and M. Share, 'CyGraph: Graph-Based Analytics and Visualization for Cybersecurity,' in *Cognitive Computing: Theory and Applications, Handbook of Statistics 35*, Elsevier, 2016.

[16] S. Noel and W. Heinbockel, 'An Overview of MITRE Cyber Situational Awareness Solutions,' in *NATO Cyber Defence Situational Awareness Solutions Conference*, Bucharest, Romania, 2015.

[17] W. Heinbockel, S. Noel, and J. Curbo, 'Mission Dependency Modeling for Cyber Situational Awareness,' in *NATO IST-148 Symposium on Cyber Defence Situation Awareness*, 2016.
[18] S. Noel, J. Ludwig, P. Jain, D. Johnson, R. K. Thomas, J. McFarland, B. King, S. Webster, and B. Tello, 'Analyzing Mission Impacts of Cyber Actions (AMICA),' in *NATO IST-128 Workshop on Cyber Attack Detection, Forensics and Attribution for Assessment of Mission Impact*, Istanbul, Turkey, 2015.
[19] R. Punnoose, A. Crainiceanu, and D. Rapp, 'Rya: A Scalable RDF Triple Store for the Clouds,' in *1st International Workshop on Cloud Intelligence*, Istanbul, Turkey, 2012.
[20] Defense Information Systems Agency (DISA), 'DISA's Big Data Platform and Analytics Capabilities,' [Online]. Available: http://www.disa.mil/newsandevents/2016/Big-Data-Platform. [Accessed 30 May 2017].
[21] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*, Sebastopol, CA: O'Reilley Media, 2015.
[22] The Apache Software Foundation, 'Apache Accumulo®,' [Online]. Available: https://accumulo.apache.org. [Accessed 30 May 2017].
[23] Wikipedia, 'Host Based Security System,' [Online]. Available: https://en.wikipedia.org/wiki/Host_Based_Security_System. [Accessed 31 May 2007].
[24] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*, Second ed., Sebastopol, CA: O'Reilly Media, 2015.
[25] E. Eifrem, 'Meet openCypher: The SQL for Graphs,' [Online]. Available: https://neo4j.com/blog/open-cypher-sql-for-graphs/. [Accessed 30 May 2017].
[26] W3C Recommendation, 'SPARQL 1.1 Query Language,' 21 March 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/. [Accessed 30 May 2017].
[27] 'What Is Splunk?,' [Online]. Available: https://www.splunk.com. [Accessed 31 May 2017].
[28] 'About Wireshark,' [Online]. Available: https://www.wireshark.org. [Accessed 31 May 2017].
[29] 'NVD – National Vulnerability Database,' [Online]. Available: https://nvd.nist.gov.
[30] S. Noel, 'Interactive Visualization and Text Mining for the CAPEC Cyber Attack Catalog,' in *ACM Interactive User Interfaces Workshop on Visual Text Analytics*, 2015.
[31] The MITRE Corporation, 'Crown Jewels Analysis,' [Online]. Available: http://www.mitre.org/publications/systems-engineering-guide/enterprise-engineering/systems-engineering-for-mission-assurance/crown-jewels-analysis. [Accessed 31 May 2017].
[32] The MITRE Corporation, 'Cyber Command System (CyCS),' [Online]. Available: http://www.mitre.org/research/technology-transfer/technology-licensing/cyber-command-system-cycs. [Accessed 31 May 2017].
[33] The Apache Software Foundation, 'Apache Storm,' [Online]. Available: http://storm.apache.org. [Accessed 1 June 2017].
[34] Defense Information Systems Agency, 'Assured Compliance Assessment Solution (ACAS),' [Online]. Available: http://www.disa.mil/cybersecurity/network-defense/acas. [Accessed 24 May 2017].
[35] S. McGillicudy, 'Flow Data is Top Source for Network Analysis,' [Online]. Available: https://www.kentik.com/flow-data-is-top-source-for-network-analysis/. [Accessed 31 May 2017].
[36] Sandia National Laboratories, 'Computer & Information Sciences (Labs Accomplishments May 2017),' [Online]. Available: http://www.sandia.gov/news/publications/lab_accomplishments/articles/2017/. [Accessed 11 July 2017].