sqrrl Securely explore your data

White Paper

LINKED DATA FOR CYBER DEFENSE

Sqrrl is the Big Data Analytics company that lets organizations pinpoint and react to unusual activity by automatically uncovering hidden connections in their data. Sqrrl's linked data analysis platform gives analysts a way to visually investigate these connections, allowing them to rapidly understand their surrounding contexts and take action. With Sqrrl's cybersecurity solution, users can detect and respond to advanced data breaches associated with cyber-espionage activity, insider threats, and other types of hard-to-detect attacks. At the core of Sqrrl’s architecture are a variety of Big Data technologies, including Hadoop, link analysis, machine learning, Data-Centric Security, and advanced visualization.

For more information contact Sqrrl at info@sqrrl.com.

TABLE OF CONTENTS

I. INTRODUCTION

Cybersecurity has come to the forefront of today's modern, connected lifestyle. If you are connected to the Internet in any way, there are attackers seeking to gain unauthorized access to your assets – by the same token, you are incentivized to protect those assets. Government leaders have cited defense as a top national priority. Industry analysts have surveyed many corporations' intention to increase spending in cybersecurity products, tools, and staff.

Despite all the recent attention to and awareness of the stakes of cybersecurity, data breaches affecting millions of people's personal and financial information continue to hit the news wire on a regular basis. Attackers have both the skill and determination to circumvent traditional security controls, evading detection and hampering the process of cleaning up the mess.

"Prevention always fails", says Brookings Institute Fellow and renowned cybersecurity researcher, Richard Bejtlich. This is not to say that preventive measures are useless, but instead that organizations must arm themselves with proficient detection and response practices for readiness in the inevitable event that prevention fails.

Today's state of the art for cyber incident detection and response is wrought with challenges. For one, many of the current solutions focus solely on alert-oriented data. Alerts are great in theory - they are meant to curate events happening as they occur and give analysts a list of things to attend to. In practice, however, these alerts are difficult to prioritize because they only provide a limited view as to what's going on. An investigator cannot easily ascertain the full context of the alert, the object it's alerting on, and everything else that object relates to. Analysts are left digging through log files, manually jumping from repository to repository in order to find and assemble the pieces to the puzzle.

There is a better approach to detecting and investigating cybersecurity incidents: a technique called Linked Data Analysis. Linked Data Analysis gives cyber "hunters" and incident responders a way to quickly identify the important assets, actors, and events relevant to their organization, accentuating the natural connections between them and providing contextual perspective. With this added context, it becomes much easier to see abnormal activity and assess the blast radius of an attack.

This paper provides an overview of Linked Data Analysis, offering a definition of what it is and the types of questions it enables. Following this overview is discussion on how to build linked data models for cybersecurity, as well as factors to consider in the process. Finally, we'll look at more advanced analyses using linked data to detect different types of cybersecurity anomalies.

II. WHAT IS LINKED DATA AND LINKED DATA ANALYSIS

Linked data was coined by Sir Tim Berners-Lee, Director of the World Wide Web Consortium (W3C) and inventor of the web. Linked data seeks to make data more useful, by organizing in such a way that it allows for semantic queries - queries that are contextual in nature and provide meaning to the things they describe.

As the name implies, linked data describes a format for data representation that highlights the different types of relationships, or links, between entities. In this case, an entity is a logical item of interest, such as a 'user', a 'website', an 'HTTP transaction', and the like. It's sometimes helpful to think of entities as the *nouns* within an analytic model. These entities are then linked via different types of relationships – for example, a user can 'know' another user, an employee can 'work for' a manager, etc. Similarly, it can be helpful to think of relationships as the *verbs* of an analytic model.

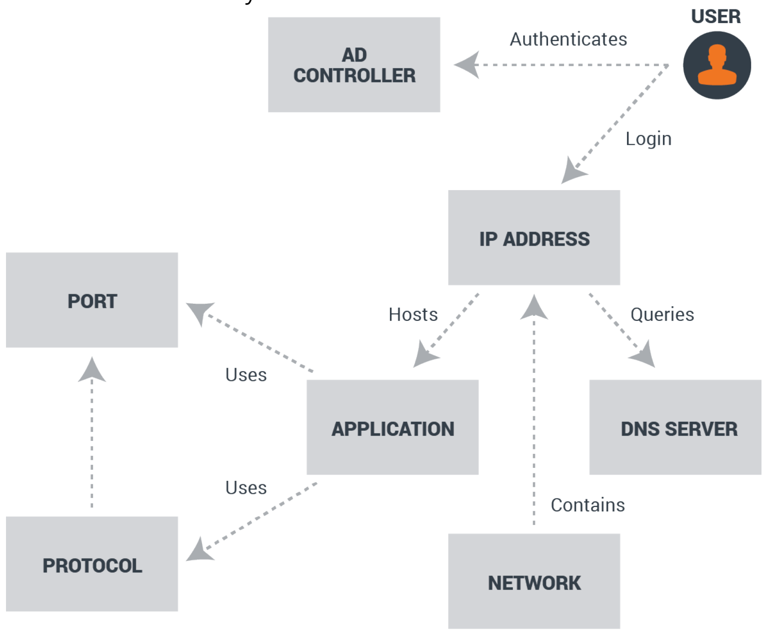

Figure 1: An example linked data model

As opposed to other, traditional data formats like flat files or database tables, linked data is especially helpful in surfacing the meaning and context of information. In other words, linked data is *defined* by its connections.

Linked Data Analysis refers to the process of combing over linked data, traversing the links and deriving new insights. Linked Data Analysis includes several techniques such as:

- Exploration - Here, exploration refers to the process of navigating the context of linked data elements, following the connections to show how things are related and in what way. For example, say you are starting with a 'user' and want to ask the question, "Show me all the websites this user has visited in the past day." A user can then ask dynamically expand out relationships from this data, asking questions like "Show me how all the users that have also visited these websites within the same time window" using a simple operation. Linked exploration is especially powerful when coupled what data visualization techniques, some of which are described later.

- Iterative Linked Search - Linked data search encompasses traditional methods such as lookup by value and full-text search, but also enables multi-hop questions that leverage the connections in the data. For example, "Show me all email attachments sent by a list of senders, and also the users who have opened them." Users can incrementally build upon linked searches in a repeatable fashion. It's then both possible and easy to launch a new question off the responses from an initial search. Building on the last example, finding out something like list of people who commonly communicate with the those who opened the attachment can be achieved with a straightforward follow-on step.

- Statistics and Summarization - A linked data system aggregates metrics about the individual elements present within the model, as well as about the model as a whole. Aggregations on individual data point behaviors can be stored, along with characteristics about the data distribution. For example, sums, averages, and time series behaviors are tracked and linked to the logical items that they pertain to. Further, overall characteristics of the data within the model such as average number of links per data type or relative importance measures are embedded alongside the elements that they describe, making it very easy to put the data into context.

- Inferencing - Inferencing refers to deriving new knowledge from the foundation of data already present. Inferencing is a powerful information processing technique in its own right, but those strengths are accentuated when used over linked data elements. Linked inferencing can draw new conclusions about individual data points or sequences of data by looking outwards across the links. For example, if a subset of users interacts with the same set of websites in the same way, we may be able conclude that these users are similar in some way. There are specialized advanced analytics that leverage linked data models to identify peer groups and to detect anomalies. Examples of both techniques are discussed toward the end of the paper. Like in the case of statistics and summarization, inferenced data is linked back to the elements it describes to enrich the overall available analysis context.

III. WHY LINKED DATA ANALYSIS?

As mentioned earlier, with linked data, the individual relationships between entities are encoded within the actual data representation. This makes it far easier for computers to traverse and process data elements and the connections amongst them without having to rely on complex, human-crafted queries.

Further, this *explicit* relational data representation lends itself very naturally to visualization. As shown later in the paper, since the actual connections or links between data elements are materialized in the underlying representation, a user can easily see how one data point relates to another, and in what way. This visual aspect saves the investigator time in having to manually account for connections during the course of their analysis.

Generally speaking, Linked Data Analysis excels in three categories of functionality:

- Pattern Matching – Given a sequence of criteria to look for, a system will search and retrieve the desired matching result. With linked data, these patterns can be quite intricate and hard to solve with more traditional approaches such as full-text search and relational database queries. For example, "go find me the set of all users who have logged into machines which have uploaded greater than 10MB of traffic to servers located overseas in the past day, and also show me how those users, machines and servers are related". Linked data makes these sequences and the questions an analyst would ask of them much easier to specify without the need to craft a complex "join" statement or have a database system execute a computationally intensive task.

- Pattern Discovery – This refers to the process of determining what sets of criteria ought to be used to drive pattern matching activities. Linked data pattern discovery can be more powerful than its more general, non-linked counterparts in that it can factor in not only what a single entity is connected to, but also what links exist – or don't exist – among the set of the primary target's connections. Using linked data models, both analysts and algorithms are able to identify interesting event sequences, connection paths, and data geometries that serve as new type of rules for matching patterns against.

- Anomaly Detection –This is the act of creating a baseline of "normal" behavior within a target set of data, and detecting abnormal behavior relative to that baseline. Within the realm of linked data, anomaly detection becomes especially powerful. When a model has different types of entities and different types of relationships, the level of granularity in the baselines that can be built is greatly expanded. An analyst is able to compare how the behavior of an individual user or machine has changed over time relative to its own previously exhibited behavior – or perhaps the analyst is more interested in how these individual users or machines compare to the behavior of their peer groups. Using linked data, these behavioral profiles can also factor in the connectedness of the data. For example, with linked data, a user can ask "show me abnormal login activity among web servers that have a high overseas user base" or "show me abnormal file transfer activity originating from machines with low overall usage".

IV. APPLYING LINKED DATA TO CYBERSECURITY

The first step in applying the linked data approach to any domain is to identify the desired analytic model to build. A linked data model can be thought of as an *ontology*: a representation of a particular business domain. This ontology is a collection of *entities* (the logical items of interest), *relationships* (connections between these entities), and *features*, properties that describe each of the entities and relationships within the model. Determining which features should be built into the model is both an art and a science, and some considerations for feature selection are discussed later in the paper.

As mentioned earlier, linked data analysis provides enhanced context for an analyst to better understand previously hidden patterns in data. From a cyber perspective, an investigator can use linked data to integrate telemetry from network equipment, endpoint devices, applications, and user activity to convey inherent connections between these entities. By deriving knowledge from raw data and building links across this knowledge model, she can then contextualize potential indicators of compromise and make them actionable.

The next few sections prescribe a methodology to apply when building a linked data model for cybersecurity analytics. One of the powers of linked data analysis is that the process of model building can be iterative - more dimensions can be scaffolded on in a straightforward way for additional context and perspective.

A. Choosing Entities and Relationships by Asking the Right Questions

Building a linked data model involves specifying the types of entities and relationships that define it. As described above, entities and relationships are the core elements of a linked data model. Entities are unique logical items of interest, and relationships are connections between entities. You may know many of these up front, but a good way to craft your model is to enumerate the questions that you know you will frequently want to answer.

For example, you might be interested in *common patterns of login activity, so you can detect aberrations from that pattern.* What constitutes a login? One way to think about a "login" is identifying the human actor executing the login, and the machine asset the human actor is logging into. While not all logins have human actors behind them (service accounts used for automation, for example), let's consider this model for the sake of argument.

Even with these assumptions to simplify the example, there are certain complications. First, are all machines created equal? There are differences between laptops, workstations, and servers. Even amongst those types, there are smaller peer groups such as web servers vs. database servers. Do each of these variations need to be a separate type of entity?

Second, what identifies a machine? Is it a hostname? A network identifier like a MAC or IP address? There's no real right or wrong answer to this question, but it highlights an interesting point: many of the features of a machine are dynamic in nature. At the end of the day, there will be some tradeoffs in the modeling process.

We will revisit this distinction later in talking about model variations/augmentations, but for the sake of this example, assume that we have a single *Host* entity type representing all machines, and each host is identifiable by 1 IP address at a given point in time.

Going back to the concern of what constitutes a login, let's now focus on the human actor. Therefore, it also makes sense to track *User* entities, as each unique user is a logical item of interest. What defines a user? Is it their full name? Their address, date of birth, or Social Security Number? Their login name or employee ID? Again, we have several potential valid options here, as each of these properties is certainly descriptive of a particular user. For the purposes of this example, we will keep it simple and say that a *User* is defined by their full name.

In order to characterize patterns in login activity, we have both *User* and *Host* entities. Naturally, we will also have a *User logged into Host* relationship that connects users with hosts when there are logins present. We can visualize this very simple starting model in the following way:

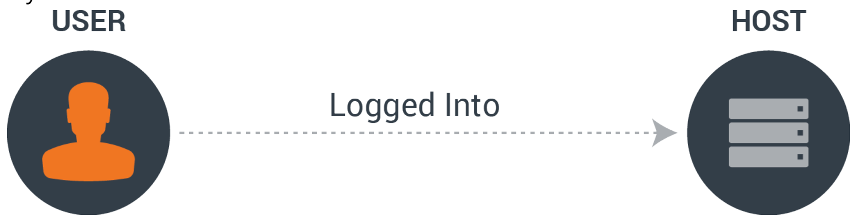

Figure 2: A simple starting model

What if we wanted to also ask about where the logins originated? Users may physically walk up to a machine and log in, but for web applications and servers, this is typically not the case. Users may log into machines *from* other machines. Another way to think of this relationship is that a machine logged into another one, or a *Host logged into Host* relationship. To accommodate this added use case, we would need to add onto our model:

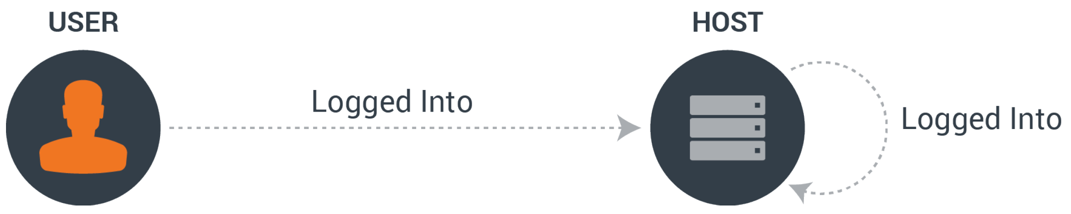

Figure 3: Adding a new relationship

This model works well if all you care about is when a user logs into a host. In practice though, this is probably not the case. You'll likely want additional context such as: which hosts are in communication with one another, where the users fall in the overall organizational structure, what files a user has access to, what ports and protocols are most commonly used in a local branch, etc. In order to get the answers to all these questions, you'll need to include additional types of entities and relationships.

There are bits and pieces of information about these entities and relationships scattered across many data sources, but the events represented in any single data source alone cannot comprehensively provide full context. So, in order to explore our model, we must first be sure it is populated with the right data.

B. Identify Relevant Data Sources

The next step in building a linked model is determining which data sources contain information about the things that you want to analyze. These sources usually are comprised of the *raw data* that will be germane to security incident detection and investigation. Here, "raw data" refers to the telemetry being generated by users, systems, and network devices on your network. Inside this raw data, there are nuggets of 'truth' that contribute to the overall context of entities and relationships, but they need to be mined out and organized in the right way. At the same time, keeping raw data around in its full fidelity can be useful for forensic activity to corroborate the conclusions drawn from linked models.

Luckily, many of the data sources are already identified by an enterprise's security organization for use in an intrusion detection system (IDS) or security incident events management system (SIEM). Those same data sources can be used in linked data models, as well as potentially other sources that these more traditional solutions cannot handle.

Network summary information from proxies, routers, and switches can be combined with system/host log information to attribute network activity with user behavior. Diagnostic information from DNS, DHCP, and SMTP can add valuable context to the picture, including requests and state changes across these services. Modeling the patterns in how critical network services function and fluctuate with one another can provide leads to an investigation. Networks are only part of the story, though - folding in user records from applications and email will allow for behavioral profiling of the actors using the network. All of these sources can be further augmented with network packet capture (PCAP) and process call stacks to corroborate any hypotheses that may be gleaned from a higher-level view.

While not "raw" data per se, but there is a wealth of information being generated by monitoring tools, IDS systems, SIEMs, and open-source threat intelligence that relates to the information previously discussed. In and of itself, this data is useful, but often times lacks full context about the various actors, assets, and events. By integrating security intelligence data with the rest of the raw data sources, you can maximize investment in existing security tools by making data you already have much more actionable.

C. Data Integration

Once the relevant data sources are enumerated and the entities and relationships that comprise the model have been determined, the next step is to map data into the model. The process of data integration involves combining data in different formats from a variety of sources into a unified, cohesive view.

Transforming data from sources into models can be thought of in two ways. One is model-centric: data is organized around the entities and relationships that are part of the model. Another is source-centric: one or more parts of raw data records are decomposed so they can projected into the model. Data integration is the process of merging these two perspectives.

As previously discussed, lots of information about individual entities and relationships is dispersed across many log records, databases, and other systems of record. Sometimes, each record itself contains information about *multiple* of these things. Consider an example record of network summary information, like NetFlow or IPFIX. Each of these records contains information about two distinct host entities: a sender and a receiver. It also contains a relationship depicting the nature of communication: when it occurred, how many bytes were sent, what ports and protocols were used, etc. Later, we'll discuss possibilities for creating even more entities and relationships from this single record, but this example shows how something as simple as one log entry can be projected into multiple dimensions in a linked data model.

D. Feature Selection and Model Variations

As discussed above, there are many properties that describe an entity or relationship. Some of these can be used as the identifier, but there are others that are purely descriptive. We call these descriptive properties *features* of an entity or relationship.

In the case of a user, if he or she is identified by their login name, things like first and last name, age, address, favorite color, etc., could all be considered features of the user. In essence, features provide more information about an entity or relationship -- they are a type of *context.* Besides the static feature examples above, dynamic features that describe *behaviors*, such as the average amount of foreign websites visited per day, or summary metrics on series of daily packets that are dropped by a firewall device, provide extremely useful context and serve as the foundation for advanced analytic tasks. This topic is reviewed in a subsequent section of the paper.

Let's go back to the example of depicting common login patterns that we used to derive our example model. What if you wanted to ask more descriptive questions of these patterns? For example, what is the average number of login attempts per user per day? Or per hour?

While these can be computed on an ad hoc basis by looking at the raw data and calculating a result, another approach is to pre-compute the answers to commonly asked questions by linking them directly into the model. Features can capture this dynamic context as well, providing readily accessible answers to common, specific questions.

Let's take a look at our example model with some features added on to the entity and relationship descriptions.

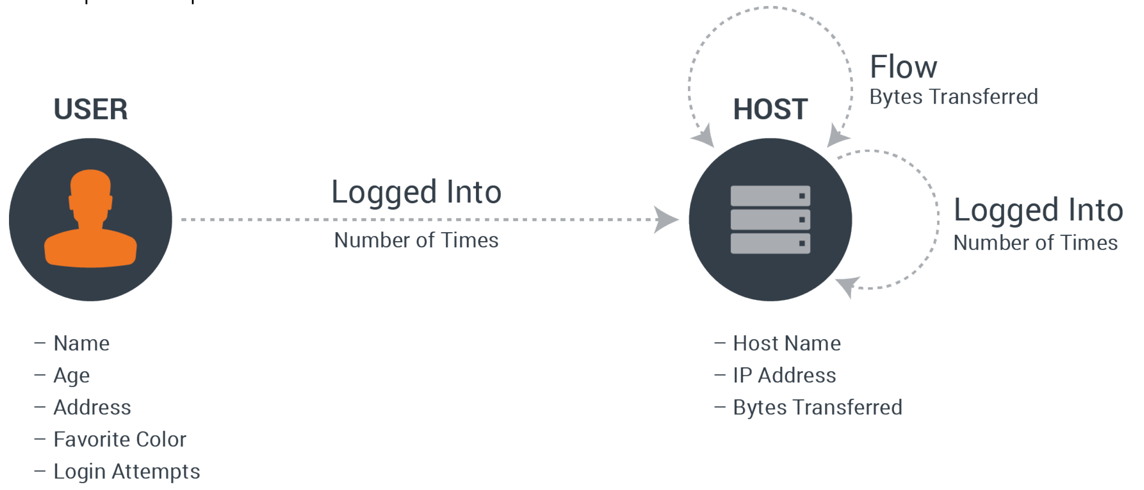

Figure 4: Enriching a model with features

Notice the loginAttempts feature of *User*. This allows for rapid evaluation of the time series of login attempts per user, and even opens the door to building behavioral profiles across the user base. Similarly, *Host* contains a bytesTransferred feature. This feature has some added dimensionality, a condition which groups the information by network port. This allows the user to quickly characterize the behavior of each machine's communication on each port.

What if you're interested in the total amount of traffic on a given between *two distinct Hosts?* Given that a relationship is meant to capture interactions between two entities, this can be stored as a feature on the relationship. Note the bytesTransferred feature on the *Host flow Host* relationship, which allows for the evaluation of traffic between any two hosts, broken down by which port it occurred on.

Let's visit the case discussed earlier on different types of users. There may be categorically different classes of users: employees vs. contractors, executives, systems administrators, etc. While it is possible to describe these user classifications using features, it is also possible to capture the distinction *using additional entity types*. Similarly, what if you cared about the difference between communications between two hosts on TCP vs. UDP? These too could be features on your flow relationships, or each could be modeled as a *separate type of relationship*.

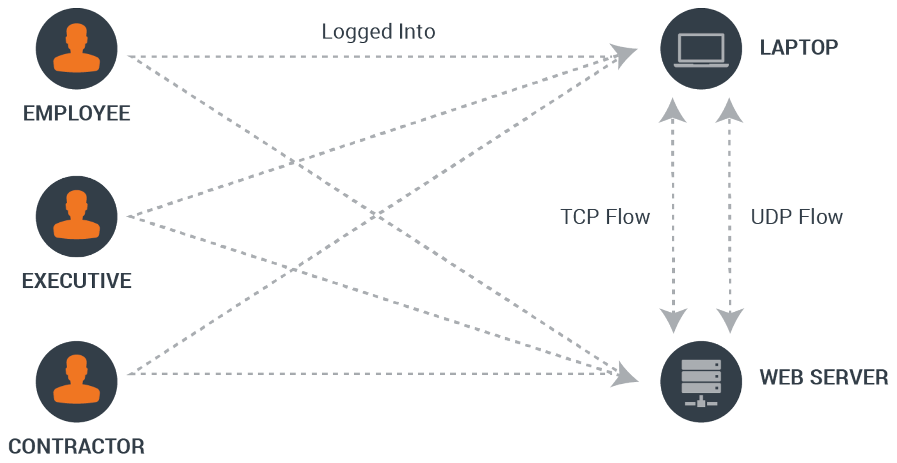

Figure 5: Model variation with different relationship types

In fact, the choice between modeling a descriptor as a feature or as a different type of entity/relationship is not a mutually exclusive one. In the world of linked data, there is a duality of sorts between features and proper relationships, as both serve as additional context to the overall model. As opposed to other data architecture, it is acceptable and often times useful to denormalize your data. Introducing more dimensions in the number of entity and relationship types can add additional perspectives to your analytic model: different visual patterns could emerge to the analyst, and as discussed later, algorithms can have additional paths to traverse to factor into their computations.

Ultimately, there are tradeoffs in the choices for building a model in terms of what features to use, what types of entities and relationships to identify, and the sources that are used to populate data. The modeling process is somewhat more of an art than a science, as the model highly depends upon the use case, the nature of the data stored in the model, the types of questions that need to be asked, and even the technology implementation behind it. As a general recommendation:

- Start with as simple a model as possible for the questions you know you want to ask.
- Incrementally adjust your model to accommodate questions you cannot yet answer.
- More features and more types of entities and relationships is not necessarily better than fewer.

E. Linked Data Analysis and Anomaly Detection

Linked Data Analysis provides unique advantages in the area of anomaly detection. Anomaly detection loosely describes determining what normal is, and finding things that fall outside that boundary. The linked data approach introduces more dimensions for building these baselines of "normal", as both features and sequences of relationships can be factored into the analysis. Now that we understand more about what goes into building a linked data model, let's take a look at some example cases of how to leverage the entities, relationships, and features for anomaly detection activities.

- Instance-based anomaly detection - An *instance* is any unique entity or relationship: the user Jane, the host 192.168.1.1, the Jane logged into 192.168.1.1 relationship. Instance-based anomaly detection involves figuring out whether any individual instance's behavior has deviated from its previously exhibited behavior. For example, if Jane typically accesses 100 files per day, but one day she suddenly accesses 8000, this could be indicative of account compromise or malicious behavior. As previously discussed, if daily file accesses are a metric we're interested in, we could choose to model that as a feature on the user entity to streamline the tracking and computation of this baseline.

- Peer-based anomaly detection - This refers to the process of determining whether or not the behavior of a particular instance is different enough from its peer group to be considered abnormal. For example, which user is unlike the others, or which network flow is unlike the others? We can also use instance links to factor into the overall analysis. In other words, the behavior of an entity's connections can be used to influence the assessment that entity's own behavior. If Jane commonly visits websites that are known be associated with malicious activity, we might cascade that judgment down to Jane as well.

  Another area where links can be used is in actually determining what constitutes a "peer group". It was earlier discussed how there may be different types of entities in the categorical sense - different classifications of 'users', for example - but it may also be the case that we can use algorithmic techniques to figure out which entities are similar enough to be considered a grouping. In other words, if 100 users have logged into a compromised host within a given time window, perhaps we want to know which of their other activities are materially different across that set of users, so we can track down and remediate strange behaviors.

- Structural anomaly detection - This type of detection focuses on the "shapes" formed between the entities within the model via the relationships between them. - which types of paths, sequences, and patterns are common across the links in the model, and where we see deviations from this pattern.

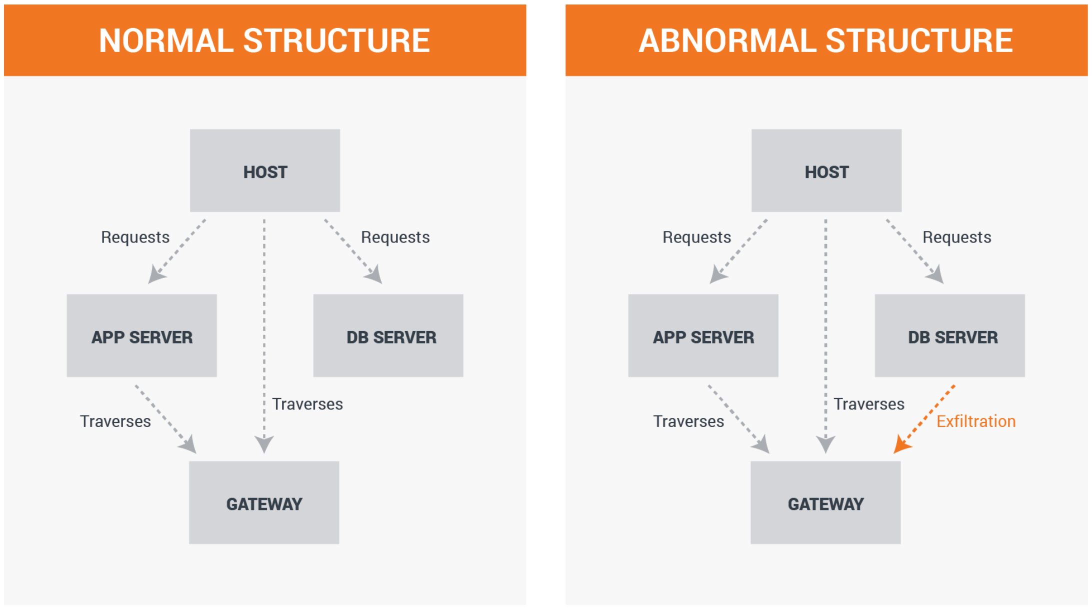

Figure 6: Structural anomaly detection example.

Here we have a more descriptive yet common model. If we can baseline that the expected behavior is that database server communication flows tend to be self-contained in the internal network, we can, by looking at the links within the model, determine that if there's a relationship directly to an instance that isn't contained on the internal network, there may be an indicator of exfiltration. This result may seem obvious, but the subtlety is in how it is detected - not by a rule, but automatically via the nature of the types of entities and relationships built into the model. This approach can be used on any other type of "shape" that emerges.

- Layered anomaly detection - Each of these detection approaches can be used in tandem to provide higher confidence measures in flagging an 'anomaly' and reducing false positive rates.

V. CONCLUSION

As you have seen, linked data gives security professionals a way to visually navigate through the full context of the assets, actors, and events relevant to their organization. Armed with the powerful capabilities of Linked Data Analysis for discovering patterns, matching patterns, and detecting anomalies, security teams can more easily prioritize alerts by filtering out the noise. A streamlined, prioritized work queue reduces the mean time to know about a data breach, get a handle on its effects, and investigate its cause. A linked data approach allows for a unifying the incident detection and response practices by enabling analysts to inspect the full context what they need to know.