Toward a Query Language for Network Attack Data

Bee-Chung Chen, Vinod Yegneswaran, Paul Barford, Raghu Ramakrishnan
Computer Sciences Department
University of Wisconsin – Madison
{beechung, vinod, pb, raghu}@cs.wisc.edu

Abstract

The growing sophistication and diversity of malicious activity in the Internet presents a serious challenge for network security analysts. In this paper, we describe our efforts to develop a database and query language for network attack data from firewalls, intrusion detection systems and honeynets. Our first step toward this objective is to develop a prototype database and query interface to identify coordinated scanning *activity in network attack data. We have created a set of aggregate views and templatized SQL queries that consider timing, persistence, targeted services, spatial dispersion and temporal dispersion, thereby enabling us to evaluate coordinated scanning along these dimensions. We demonstrate the utility of the interface by conducting a case study on a set of firewall and intrusion detection system logs from Dshield.org. We show that the interface is able to identify general characteristics of coordinated activity as well as instances of unusual activity that would otherwise be difficult to mine from the data. These results highlight the potential for developing a more generalized query language for a broad class of network intrusion data. The case study also exposes some of the challenges we face in extending our system to more generalized queries over potentially vast quantities of data.*

1 Introduction

Malicious activity such as DoS attacks, email viruses and self propagating worms has been prevalent in the Internet for some time. Over the past several years, new malicious code bases (malware) have been marked by their increasing sophistication and broadening set of capabilities, blurring the lines between some of the traditional malware types [9]. The result is that the task of securing networks from attacks is growing more and more difficult.

Our research is focused on the long term objective of improving tools and methods for defending networks from malicious attacks. Over the past several years, we have taken an empirical approach to this work by using trace files collected on a daily basis from firewalls, network intrusion detection systems (NIDS) and honeynets distributed throughout the Internet [2, 17, 11, 15, 16]. These studies, based on huge, multidimensional datasets, have brought into sharp focus for us several of the challenges of sound Internet measurement study described by Paxson in [12]. In particular, there is a significant need for consistent data management and verifiable analysis tools that enable flexible and efficient access to the data.

To address these problems we have undertaken the task of developing a database management system and query interface for our large, diverse set of network attack data. While there are many engineering challenges associated with designing, implementing, managing and tuning a database like this, the research challenge is in developing methods and tools for accessing the data and mining it for information of interest to security researchers and analysts. Our objective in this regard is to develop a comprehensive query language for network attack data. At a high level, this language should facilitate both broad exploratory analysis and efficient drill-down analysis of network attack data.

Our approach to the query language development is to identify touchstone characteristics that are common to many types of network attacks and then build functional capability to analyze these properties. We bootstrapped the process by developing an interface to investigate the extent and composition of *coordinated scanning* activity in the Internet. In our context, the notion of "coordination" can be defined in several ways, but at basis it means multiple hosts (identified by source IP addresses) conducting malicious activity at roughly the same time. Potential root causes of coordinated scanning activity could include botnets, port scanners, spammers, misconfigurations, DDoS attacks and new outbreaks among others. While there is anecdotal evidence that coordination takes place on a large scale and some sources of coordination such as worms have been well studied, coordinated scanning in general has not been well studied to date. We hypothesize that understanding characteristics of coordinated behavior will provide a foundation

for defending networks against repeated attacks from common sources.

Development of our query interface for analyzing coordinated scanning behavior has been focused on designing a stylized language capable of expressing a range of complex SQL queries. Our approach includes the use of multi-level summarization techniques as a means for efficient execution of the queries. We present the outline of the language framework and a set of templatized queries that illustrate the promise of our methods. The key idea is that we must be able to search for coordination over *subsets* of activity, across different candidate sets of sources, targets and time frames.

We demonstrate the value of the templatized queries through the results of a case study using a prototype MySQL database populated with firewall and NIDS logs provided by Dshield.org [14]. The initial results indicate that coordinated scanning is, in fact, a rich phenomenon with both structure and diversity that deserves further investigation. The results also highlight the fact that even our prototype methodology is sufficient for identifying interesting and unique scanning events that might otherwise be hard to find by hand. We provide examples of qualitative and quantitative summaries of coordinated scanning activity generated by our tool from our sample data set.

2 Related Work

While database systems have not been widely used in empirical network research studies there are several well known instances of their use in the network operations domain. The Round Robin Database or RRDTool [10] is used for logging network data such as SNMP or Netflow. It is designed for efficient storage of time-series data and is accessed by writing scripts in languages such as Perl. Another network data management system is Daytona which was developed at AT&T Research [13]. This system is used to archive operational data from AT&T's commercial network. Access to Daytona is provided through the Cymbal query language [7]. Finally, Li *et al*. describe the MIND system for indexing multi-dimensional data collected at distributed sites [8]. While the current focus of our efforts is on a centralized repository, our query language could be extended to take advantage of MIND in the future.

Data stream management [5] is an active field in the database community. Several systems have been built for network traffic analysis. For example, Gigascope [3] was developed at AT&T Research for network monitoring using a SQL-like language. MINDS [4] (not to be confused with MIND above) is an example of a stream-base system developed for network intrusion detection. However, the main focus of these and other stream systems is on *online* monitoring. In contrast, we consider multi-level complex exploratory analysis for *offline* evaluation. The motivations behind our multi-level offline evaluation approach are similar in principle to those of OLAP systems. However, the feasibility of extending such systems to handle the complex exploratory analysis we consider remains somewhat unclear.

3 Coordinated Scanning Design Space

There are several characteristics that affect the degree of coordination and our ability to identify these scans in network data.

1. **Data collection:** The network attack data that will be analyzed initially comes from firewall logs and NIDS logs. This data has been shown to provide a valuable perspective on global scanning activity [17]. An important aspect of this data is that it is assumed to be anomalous *i.e.,* either a result of misconfiguration or malicious attack, and is not combined with otherwise benign traffic. This simplifies our analysis in some respects. However, since the data is collected *passively*, we have to infer the attacker's intentions based on coarse-grained metrics such as the destination port and the scanning footprint. We also do not currently have any good means to accurately detect spoofed sources in this data. In the future, we intend to include data from honeynet systems [15] which provide a richer set of information on attacks by responding to the unsolicited traffic sent to unused address space.

2. **Target service (single/multiple):** One design choice is whether we consider activities of sources on target ports in isolation or collectively. Considering activity on individual target ports in isolation provides significant scalability benefits. Conversely, looking at activity across ports can provide insight on finer characteristics of source behavior such as ordering inherent in target port selection (*pre-cursor* and *follow-up* ports). This would potentially provide additional means to differentiate between coordinated scanning groups.

3. **Target network (single/multiple):** An important characteristic of attack activity is its level of *scoping* – *i.e.,* is the activity local, global or something in between? While worm outbreaks often tend to be global (*i.e.* they traverse a large portion of the IPv4 address space randomly), coordinated scans targeting a single network or a single destination IP address usually suggest a misconfiguration, denial-of-service attack or a localized botnet sweep. Unfortunately, due to limited measurement perspective, it can often be difficult to verify that an observed phenomenon was indeed restricted to a single network.

4. **Duration (short/long/persistent):** The duration of scanning events that we would like to identify is inherently tied to the granularity of detection algorithm. To determine the level of coordination accurately, we need the ability to test for coordination at different time granularities. Persistent events are usually indicators of worm traffic and certain misconfiguration [15].

5. **Time lag:** An important consideration for an algorithm that identifies coordinated scanning events is the relative time-lag between instances of a particular type of coordinated event. For multi-occurrence events that are of short duration, a longer time-lag between instances could be a good indicator of coordination.

6. **Source volume:** We assume that identical events produce roughly similar levels of traffic volume. Likewise, coordinated sources should generate similar traffic volume. One complication is that since provider networks vary in size, the observed traffic volumes might need to be normalized to consider scans at a per destination IP level instead of absolute volume. Our system currently uses a simple ON/OFF metric for measuring source scanning similarity and discounts volume.

7. **Network aggregation (/8, /16, /24):** Considering activity by aggregating at different network levels is an important dimension for coordinated scanning analysis. For example, estimating the spatial dispersion of source addresses could be used to distinguish between worms and botnets. Network and temporal aggregation are key concepts in our templatized query framework and are necessary for efficient execution of queries. Thus, a basic design consideration is that our system and algorithms be flexible to support queries over various levels of network address aggregation.

8. **Accuracy:** One of the fundamental trade-offs for any anomaly detection system is the one between true and false identification. Such considerations hold for our system as well. However, due to limited fine-grained analysis capability afforded by the dataset under consideration, we do not conduct a quantitative evaluation of identification accuracy in this study.

9. **Performance:** The final design consideration is that of performance. Our objective is to develop an interactive system capable of executing queries about coordinated scanning on the order of seconds. These requirements guide our design choices in materializing views and building indexes.

4 Query Language Framework

In this section, we describe the system framework, the functionality and implementation of the current query interface prototype. The two key contributions are the advocacy of a high-level (implementation-independent) query language, and an efficient implementation approach based on multi-level aggregation, which enables interactive complex analysis on massive datasets. We note that both the language proposal and implementation ideas are at a preliminary stage; while we believe the current results make a compelling case for value and feasibility, considerable further work remains before a general, robust system can be developed.

4.1 System Framework

The two key requirements that guide the design of this prototype are (i) ***ability to support complex queries*** and (ii) ***interactive analysis***. In particular, we expect the system to support interactive exploratory analysis over very large datasets, such as the Dshield data we analyze, that contains over 2GB per day. Second, the system should be able to handle complex queries, such as *time-series similarity comparison* that are necessary for expressing notions of temporal coordination.

Our approach to efficient query processing is based on multi-level aggregation. Depending on the analysis task, multi-level aggregate views over the original, or *raw*, data are created and materialized (stored) in the DBMS. For example, suppose that in the raw data we have a record per second for each (source, target) pair of IP addresses. If our goal is to measure the degree of coordination between all pairs of sources (*i.e.* not specific to any target network), we can define an aggregate view in which each record represents the amount of activity for each source over hourly intervals, aggregated over all targets. The raw data is at the [time:second, target:IP-address] level, while the aggregate view is at the [time:hour, target:all] level. Although this view only provides relatively low-resolution information about the raw data, it is an acceptable approximation for our intended analysis, contains significantly fewer records, and supports complex analyses like temporal pattern similarity comparison to be carried out efficiently.

Aggregate views are a powerful tool, but queries (analysis tasks) should not be expressed in terms of the views; we regard these views as implementation artifacts, similar to database indexes. Different queries may need different aggregate views, and as the workload shifts, or new analysis algorithms are added to the system, the set of useful views to maintain may change. We therefore propose an abstract query language to specify the analysis task. Abstract queries are translated into database queries (*e.g.*,

SQL) which make use of the aggregate views to achieve high efficiency. The overall system architecture has the following four components:

1. **Abstract language framework**: The abstract query language together with a form-based interface allows users to specify queries.
2. **Raw data**: The raw data is stored in the system in order to support "drilling down" when interesting patterns are found using abstract queries. Note that the raw data is not necessarily stored in the online DBMS. It can be stored in multiple data files with appropriate indexes.
3. **Multi-level aggregate view subsystem**: Given a workload of abstract queries, this subsystem identifies and maintains a useful collection of compact multi-level aggregate views.
4. **Query translator**: The query translator converts abstract queries into (ideally, the most efficient collection of) database queries that use the aggregate views.

4.2 Abstract Language Design

The design of our abstract language is at a preliminary stage, and the analysis tasks presented in this paper rely only upon the following constructs. A complete language definition and discussion of other language issues is future work.

1. **Time-series expression:** The time-series of the number of packets ($npackets$) from a source network (s) targeting a destination network t at port p aggregated over time-interval($time$) within a time window (w) is expressed as follows: $q_{time:10m,npackets}(s{:}/32, t{:}/24, p, w)$.
2. **Source-set expression:** The set of sources scanning a specific network (t) and a specific port (p) during a time-window (w) is expressed as follows: $q_{\{source:/32\}}(t{:}/24, p, w)$.

The raw data table contains a large number of implicit time series, which are candidates for analysis. The time-series expression makes these implicit series explicit, and encapsulates a time-series generation process that involves selection and aggregation—$time{:}b$ specifies the granularity of time, $npackets$ specifies the series values, and $(s{:}l_s, t{:}l_t, p, w)$ selects the time series at the desired level of aggregation. Similarly, the source-set expression provides an easy and implementation-independent way to represent a generation process for candidate sets (say, of specific sources that exhibit malicious coordination).

In the following subsection, we will show the translation of the above expressions into SQL in our prototype. The SQL statements are much harder to understand and depend on the underlying implementation, *e.g.*, the aggregate views that are currently available.

4.3 Prototype Status

The prototype system built around the MySQL database is under development. We choose MySQL simply because (i) it is freely available and (ii) it is the platform of choice for Dshield. At present, query translation is "hardcoded for specific queries of our interest", and the multi-level aggregate views are also created manually. However, the prototype has allowed us to obtain preliminary results (presented in later sections) that demonstrate the value and feasibility of our approach. Our multi-level aggregate views enable all queries to be answered interactively (few seconds on a modestly configured PC).

This opens a promising research direction with many challenges, including how to optimize the translation from the abstract language to the database language, how to determine what views should be created given a workload and a storage (or processing time) budget, and how to adapt parallel database techniques to the multi-level aggregation framework. Developing general techniques for query translation and efficient multi-level aggregate computation is active research [1]. We now describe the components of the prototype, starting from the raw data, then the queries, aggregate views and query translation.

Raw data: We use two week's data (2004-02-04 to 2004-02-11 and 2004-12-10 to 2004-12-16) obtained from Dshield.org. The data consists of log records from firewalls and NIDS collected with the following attributes: $time(secs)$, $source$, $sourceport$, $target$, $targetport$, $protocol$, $flags$, $author$ and $npackets$ (number of packets). Each row represents a log record generated by a particular log provider(author), recording the number of network packets seen at a particular time(in seconds), from a particular source IP and port destined to a particular target IP and port, using a particular protocol with optional flags.

Several attributes have natural domain hierarchies, which are used in aggregation, and are therefore central to our framework:

– ***Time hierarchy:*** Time can be considered at different granularities, *e.g.*, day, hour, minute, second.

– ***Network hierarchy:*** (for $source$ and $target$): Networks can be considered at different aggregates, *e.g.*, /0 (the entire IPv4 address space); /8, /16 or /24 subnets; or /32(single IP address).

We use $attribute{:}level$ to express an attribute at a given

aggregation level (src:/24 denotes a /24 source network) [1].

Queries: Our analysis of coordinated scanning activity utilizes the following three queries.

– ***Source co-occurrence vs. time-lag***: Given a target network t at level l_t (*e.g.*, /8), a port number p, a time level b (*e.g.*, hour) and a time window $w = [w_1, ..., w_n]$ (*e.g.*, w is some day and w_i is the ith hour interval in that day), the goal is to generate a plot which contains a point $(|i - j|, y)$ for each (w_i, w_j) pair, where y is the size of the intersection of the set of sources targeting t:l_t at port p during w_i and the set of sources targeting the same t:l_t at the same port p but during w_j:

$$y = |q_{\{source:/32\}}(t{:}l_t, p, w_i) \bigcap q_{\{source:/32\}}(t{:}l_t, p, w_j)|$$

We call $|i - j|$ the time lag and y the degree of source co-occurrence. Intuitively, a high degree of source co-occurrence with a long time lag indicates possible coordination.

– ***Source co-occurrence vs. network-distance***: Given a port number p, a time window w (*e.g.*, some hour) and a target network level l_t (*e.g.*, /8), the goal is to generate a plot which contains a point $(|t_i - t_j|, y)$ for each (t_i, t_j) pair of l_t-level target networks, where

$$y = |q_{\{source:/32\}}(t_i{:}l_t, p, w) \bigcap q_{\{source:/32\}}(t_j{:}l_t, p, w)|$$

We call $|t_i - t_j|$, which is the difference between the integer representations of two target networks, the network distance. y is the degree of source co-occurrence. Intuitively, a high degree of source co-occurrence with a long distance indicates possible coordination.

– ***Sources with similar temporal patterns***: Given a port number p, a l_t-level target network t, a time level b, a time window w and a source s, the goal is to find all other sources that have time series similar to s; *i.e.*, to find all s^* such that

$$sim(q_{time:b,npackets}(s{:}/32, t{:}l_t, p, w),$$
$$q_{time:b,npackets}(s^*{:}/32, t{:}l_t, p, w)) \geq \theta,$$

where $sim(x, y)$ is a similarity function measuring the similarity between time series x and y and θ is a threshold value. In this paper, we use a simple similarity function called *on/off similarity*, which is the percentage of time during w that the two sources are both on (having scanning activity) or are both off (having no scanning activity).

Multi-level aggregate views: Due to space limitations, we only show the multi-level aggregate views and query translation for the first query; the other queries are handled similarly.

[1] Extending our framework to handle other hierarchies is straightforward.

The basic idea of multi-level aggregate views is to aggregate the raw data table at the right level so that queries can be efficiently executed. Since MySQL does not have full support for view materialization and view indexing, we materialize the views as ordinary tables. For each target network level $l_t \in \{0, 8, 16, 24, 32\}$ (0 means the entire Internet), each time level $b \in \{10, 30, 60\}$ (unit: minute) and each day d, we create a table $AggView\_l_t\_b\_d$ of attributes $time$, $source$, $target$, $protocol$, $targetport$, $npackets$ with a clustered B-tree index (primary key) on ($target$, $protocol$, $targetport$, $source$, $time$), and populate it using the SQL statement:

```
INSERT INTO AggView_l_t_b_d
   SELECT time_to_sec(time) div (60 · b) as newtime,
      source, target div 2^(32-l_t) as newtarget,
      protocol, targetport, sum(npacket)
   FROM   RawTable_d
   GROUP BY newtime, source, newtarget, protocol, targetport;
```

$time\_to\_sec$ is a function converting the MySQL internal time representation into seconds, $target$ is the target IP address in the 32-bit integer form, and $RawTable\_d$ is the raw data table for day d [2].

Query translation: Currently, the translation from queries expressed in the abstract language to queries over the aggregate views is hard-coded. The SQL statement that generates the points in a plot of source co-occurence vs. time lag, given l_t-level target network t, port number p, time level b and day d is:

```
SELECT (v1.time-v2.time) as timelag,
    count(*) as co-occurrence
 FROM AggView_l_t_b_d v1, AggView_l_t_b_d v2
 WHERE v1.target=t and v1.targetport=p and
    v2.target=t and v2.targetport=p and
    v1.source=v2.source and v1.time ≥v2.time
 GROUP BY v1.time, v2.time;
```

Note that in the WHERE clause, we have "v1.time $\geq$ v2.time" because the source co-occurrence of (v2.time-v1.time) is the same as that of (v1.time-v2.time).

5 Case Study Results

The case study with the prototype system was conducted using 58GB data set spanning two weeks from Dshield.org [14] that contained over 823M records. An important component of our study was the creation of aggregate views in order to reduce the amount of data maintained

[2] Computing all the aggregate views for a given day is similar to computing a data cube [6]. Efficient data cube computation techniques can be applied here.

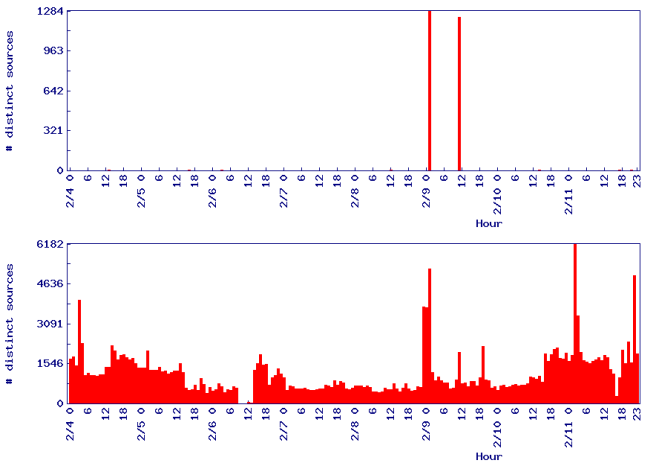

Figure 1. Single network MS-messenger pop-up spam port 1026/UDP (top) vs. all-ports (bottom).

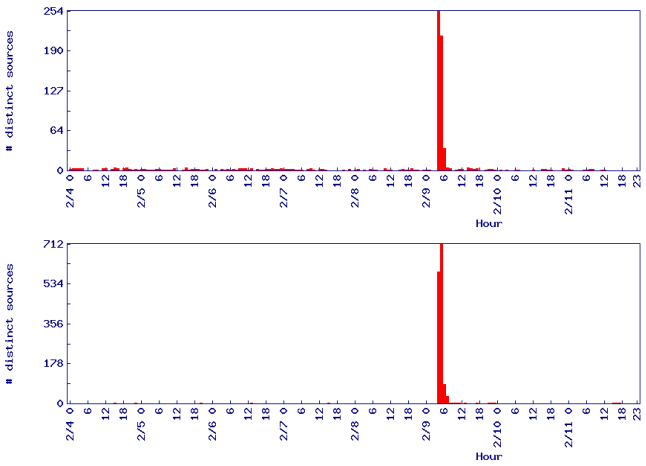

Figure 2. Multi-network Dameware botnet sweep; class B network #1 (top) vs. class B network #2 (bottom).

in the database system and thereby improve query performance. Obviously, the amount of size reduction depends on the level of aggregation. The optimal aggregation strategy balances richness of information with efficiency. In the prototype, we selected the level of aggregation based on our own experience with the data and our objectives for exploring the capabilities of the interface. We expect that individual security analysts will tune this to their own environment.

For the first part of the case study (Section 5.1), we aggregate the data to one-hour time resolution and /24 target networks. The result is a view of size 4.9GB containing 78M records. Note that as the number of days over which we execute queries increases, the time resolution can also be coarsened. As a result, we can still control the size of the aggregate view to be a few gigabytes in principle, although adaptive re-sizing strategies require further research.

For the co-occurrence analysis (Section 5.2), a set of aggregate views at different levels is created for each day. For example, on 2004-12-15, the raw data is at level [time:sec, target:/32], *i.e.* time resolution is in seconds and the targets are /32 (IP address granularity), with size 3.7GB containing 49M records. The size of an aggregate view depends on the level of aggregation, ranging from 313MB with 4M records at the coarsest level [time:hour, target:all]) to 2.1GB with 28M records at the finest level [time:10min, target:/32].

5.1 Phenomenological Exploration of Coordination

The first set of results demonstrate the capability of the interface to identify instance of coordinated scanning along several dimensions. We use a simple threshold-based technique to identify significant elevations in source counts targeting a particular service in each network [3]. These are often associated with botnet incidents or spoofed floods. We call this the β-heuristic where β refers to the ratio between the instantaneous source-count and average source-count [15]. In Figures 1 and 2, we show examples of single-network and multi-network coordinated activity that were detected using our interface [4]. Figure 1 shows UDP traffic that is sent to port 1026/UDP (MS-instant-messenger) in a single network with the intent of spamming This traffic could very well be spoofed but it fits our definition of coordination. Figure 2 is a multi-network event where the same coordinated scan anomaly is detected at different Class B networks that belong to two distant /8 networks. This traffic is sent to port 6129/TCP (Dameware), which is actively targeted by botnets. Based on our prior experience, we speculate that this traffic is not spoofed and that it is in fact a botnet sweep.

In Figure 3, we show an example of data pollution (essentially a misconfiguration) that is detected as a coordinated scan event. In this instance a provider submits logs that include traffic sent to port 27020/UDP (Half-life, a popular network game). The strong diurnal pattern associated with human-activity and the particular port number suggests that this is unlikely to be malicious traffic.

5.2 Qualitative Source Co-occurrence Profiles

The ability to obtain a quick qualitative assessment of coordination seen across different dimensions is one of the principle applications envisioned for our system. To that end, we explored the characteristic behavior of: (i) the *am-*

[3] Such techniques are state-of-the-art in honeynet monitoring systems.

[4] Time-series of the form $q_{time:hour,\#sources}(s : /32, t : /24, p, w)$.

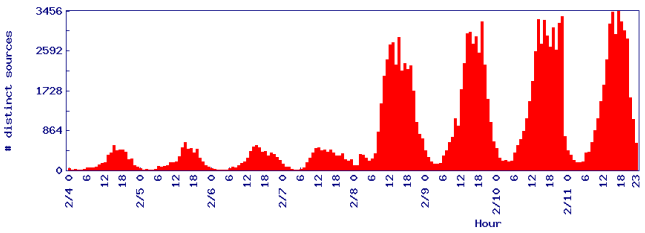

Figure 3. Misconfiguration: diurnal coordinated activity on port 27020/UDP (Half-life).

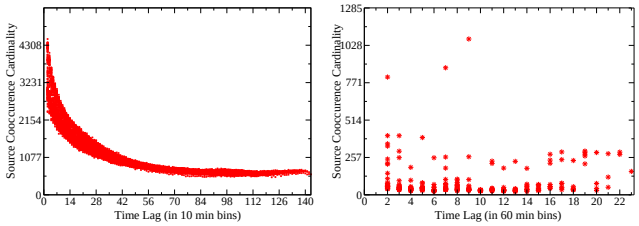

Figure 4. Left: source co-occurrence vs. time-lag for port 445/TCP (NetBIOS-SMB) at 10 min resolution, Right: source co-occurrence vs. time-lag for port 1026/UDP (MS-messenger) at 60 min resolution. Date: 2004-12-15.

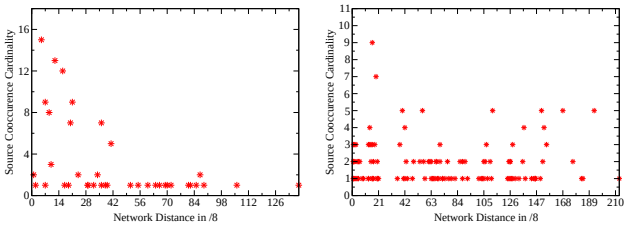

Figure 5. Left: Source co-occurrence vs. network-distance for port 3127 (MyDoom) at 60 min resolution, Right: source co-occurrence vs. network-distance for port 9898 (Sasser) at 60 minute resolution. Date: 2004-12-15.

	10 min		*30 min*	
Port	**# sets**	**μ/σ set size**	**# sets**	**μ/σ set size**
80	89	39.44/1.82	53	37.21/3.42
445	0	0/0	0	0/0
135	18	406.83/150.16	6	669.16/21.96
1026	796	25.63/44.10	58	54.74/56.10
3127	111	23.30/1.63	0	0/0

Table 1. Quantitative summary of source co-occurrence vs time-lag at 10,30 min resolutions (one day: 2004-12-13).

bient coordination from worms and viruses, and (ii) the occasional outliers that become candidates for deeper forensic analysis. It is important to note that the analyses presented in this section are directly related to the queries provided in Section 4.3.

• **Source co-occurrence vs. time-lag:** Our first analysis is based on two notions: (i) the importance of coordinated scan events increases as the time-lag between events increases, and (ii) the importance of coordinated scan events increases with the number of sources involved. Figure 4 shows examples of scatter plots created from our queries. Each point in the plots corresponds to a set of co-occurring sources that were obtained by pairwise comparison across all time intervals in a single day. The plot on the left shows the ambient level of coordinated activity on TCP port 445 (NetBIOS-SMB). This represents well known worms like Sasser and Welchia, and we can see that the cardinality of such co-occurrence can be quite high. The plot on the right shows the co-occurrence for the MS-messenger pop-up spam activity which looks vastly different. In this plot, we see examples of clear outliers that call for more in-depth analysis. Unlike the former, there appears to be limited correlation here between time-lag and expected co-occurrence of sources.

• **Source co-occurrence vs. network-distance:** Next, we consider scanning co-occurrence based on network distance. The intuition here is that since attack sources often sequentially sweep target address space or have strong local preference, adjacent networks are likely to see more co-occurring sources than networks that are farther away. In other words, co-occurrence seen among distant networks is more unusual and suggests further analysis. In Figure 5, we show the source co-occurrence vs. network distance generated for two popular worm back door ports 3127/TCP (MyDoom) and 9898/TCP (Sasser). Our analysis across various ports informs us that the cardinality of source-network co-occurrence is typically much smaller than source-time. This suggests that such anomalies should be easier to identify (and subject to greater scrutiny) when they do occur.

• **Quantitative summary of co-occurrence:** In Figure 4, values close to the y-axis represent activities that extend over multiple adjacent intervals and values close to the x-axis represent normally expected co-occurrence of sources due to Internet background radiation traffic [11]. Our interface provides the ability to automatically select and obtain summaries of the outliers or the level of *anomalous coordination* through a threshold ϕ based on the number of points. For a given threshold ϕ, we compute lag_ϕ and c_ϕ to be the ϕ^{th} percentile values for lag and source counts. Then we count as outliers any sets S such that $S_{lag} > lag_\phi$ and $|S| > c_\phi$. Example summaries for several ports are shown in in Table 1. We envision summaries like these forming the basis for longitudinal tracking of levels of coordinated scanning.

• **Drill-down analysis:** Mitigation (and forensic analysis) of coordinated scanning events will require the ability to drill-down on details of sources participating in the events. The prototype interface is able to automatically selects sources from a given group that have similar ON/OFF patterns (across all time-intervals). A time-volume graph of 5 sources from an example set of 69 “similar” sources

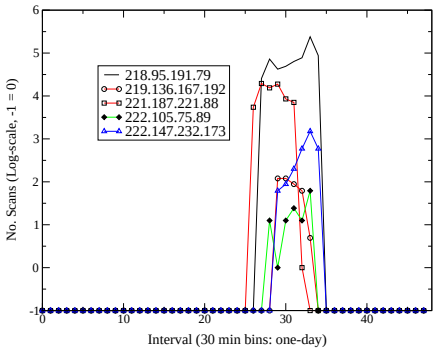

Figure 6. ON/OFF pattern of 5 (out of 69) coordinated sources targeting port 1026/UDP detected using drill-down analysis.

targeting port 1026/UDP (MS-messenger) is shown in Figure 6. Patterns like these serve as indicators for botnet activity and can aid in creation of black lists for firewalls.

6 Summary and Future Work

Malicious activity in the Internet poses many challenges for researchers and security analysts alike. Empirical study of network attacks using a large, diverse set of traces can benefit greatly from consistent data management via database systems, and perhaps more importantly from a query interface that facilitates both broad and deep analysis of the data. To that end, our objective is developing a DBMS and comprehensive query language for multi-dimensional network security data. While this goal is ambitious, we are initially focusing our efforts on an important subset of the problem.

Our immediate interest is in understanding a phenomenon that has received relatively little attention in the literature: coordinated scanning. We describe our efforts to develop a query interface that enables us to investigate coordinated scanning in terms of timing, persistence, targeted services, spatial dispersion and temporal dispersion. We created a set of aggregate views and templatized SQL queries to understand the design trade-offs and to demonstrate the utility of our approach. In applying these queries to a data set from Dshield.org we find that coordinated scanning is indeed a rich phenomenon with many interesting features. Our results also suggest the promise of our efforts to develop a comprehensive query language for network attack data. We continue to build out the language, and to investigate issues such as query translation, budget-based view materialization and parallel query processing in the multi-level aggregation framework. Finally, we are actively using the current interface to track a variety of coordinated scanning behavior including the behavioral history of identified source groups.

Acknowledgements: We acknowledge Johannes Ullrich and the Dshield providers for access to their dataset. We also thank the anonymous referees for insightful comments that improved the presentation of this paper. This work is supported in part by NSF grants CNS-0347252, ANI-0335234 and CCR-0325653. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] Lei Chen, Bee-Chung Chen, and Raghu Ramakrishnan. Composite subset measures. In *UW Technical Report*, 2006.

[2] E. Cooke, M. Bailey, M. Mao, D. Watson, F. Jahanian, and D. McPherson. Toward understanding distributed blackhole placement. In *Proceedings of CCS Workshop on Rapid Malcode (WORM '04)*, Fairfax, VA, October 2004.

[3] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: a stream database for network applications. In *Proceedings of ACM SIGMOD*, 2003.

[4] L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, V. Kumar, J. Srivastava, and P. Dokas. MINDS - Minnesota Intrusion Detection System. *Next Generation Data Mining*, 2004.

[5] L. Golab and M. T. Ozsu. Issues in data stream management. *SIGMOD Record*, 32(2), 2003.

[6] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[7] R. Greer. Daytona and the fourth generation language symbal. In *Proceedings of ACM SIGMOD*, Philadelphia, PA, June 1999.

[8] X. Li, F. Bian, H. Zhang, C. Diot, R. Govindan, W. Hong, and G. Iannaccone. Advanced indexing techniques for wide area network monitoring. In *Proceedings of the First IEEE International Workshop on Networking Meets Databases*, 2005.

[9] Metasploit. http://www.metasploit.com, 2005.

[10] T. Oetiker. Rrdtool. http://people.ee.ethz.ch/õetiker/webtools/rrdtool/, 2005.

[11] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of internet background radiation. In *Proceedings of ACM Internet Measurement Conference*, Taormina, Italy, October 2004.

[12] V. Paxson. Strategies for sound Internet measurement. In *Proceedings of ACM IMC '04*, October 2004.

[13] AT&T Research. Daytona. http://www.research.att.com/projects/daytona, 2005.

[14] J. Ullrich. Dshield. http://www.dshield.org, 2005.

[15] V. Yegneswaran, P. Barford, and V. Paxson. Using honeynets for internet situational awareness. In *Proceedings of the Fourth Workshop on Hot Topics in Networks (HotNets IV)*, College Park, MD, November 2005.

[16] V. Yegneswaran, P. Barford, and D. Plonka. On the design and use of internet sinks for network abuse monitoring. In *Proceedings of the Symposium on Recent Advances in Intrusion Detection (RAID '04)*, 2004.

[17] V. Yegneswaran, P. Barford, and J. Ullrich. Internet intrusions: Global characteristics and prevalence. In *Proceedings of ACM SIGMETRICS*, San Diego, CA, June 2003.