Data Cube Indexing of Large-Scale Infosec Repositories

Alfonso Valdes, Martin Fong, Keith Skinner
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025 USA
[valdes, mwfong, skinner]@csl.sri.com

Abstract

Analysts examining large-scale information security repositories for propagating network events are interested in quickly identifying temporal and spatial (IP address and/or port) regions containing interesting phenomena, or correlating events from different time periods. The size of these datasets strains current query capabilities provided by, for example, relational databases. We introduce a scalable, animated data cube representation and viewer, suitable for a broad range of observables, to permit coarse-grain detection and correlation in such data sets. We scale from the LAN to the Internet through flexible, locality-preserving hash algorithms mapping traffic source and destination (IP addresses or IP and port considered simultaneously). Data streams considered include inherently suspicious traffic such as packets rejected at a firewall, IDS alerts, or traffic to unused address space, as well as Netflow data. We display observables as intensity plots, where X and Y coordinates are the hashed source and target address and the intensity is proportional to traffic volume. Source and target address space may or may not be the same and may or may not be mapped the same way. Propagating events have distinct visual signatures that can be enhanced through matched filtering techniques. Future work will correlate cubes efficiently through cell-by-cell multiplication. An analyst will be able to, for example, examine whether plots representing two time periods (hours or days) exhibit similar patterns. Multiplication of a cube with its transpose permits identification of nodes that respond to potentially malicious probes. These data cubes permit coarse-grained detection and correlation without expensive data base queries.

Keywords
Intrusion Detection, Correlation, Visualization, propagating event detection.

Acknowledgments

This work was supported by ARDA under Air Force Research Laboratory Contract No. F30602-03-C-0234. The views expressed are those of the authors and do not necessarily reflect the views of the supporting agencies.

Introduction

Information assurance analysts increasingly examine large data repositories representing scales from the enterprise to the level of multiple autonomous systems. These repositories can be quite large, presently on the order of hundreds of gigabytes per day (compressed) in some cases. Commercial data base tools are unable to process such massive data sets. Queries with custom software to subset these files and identify events of interest require hours. Analysts examining these repositories are particularly interested in events with “broad footprint,” such as propagating events and so-called bot networks.

The analyst community needs a set of tools that quickly show what a particular data set contains, to identify address space and time slices of interest, and to correlate phenomenology across these sets. Of particular interest is a quick answer to the question of whether an event observed in near real time is similar to events in past data sets.

Clearly, such tools must scale to large address spaces and event volumes. A reasonable tradeoff involves visualization and correlation in a coarsely mapped manner, which trades off detail but allows analysis of the entire address space of interest. Of necessity this correlation and visualization is coarse grained, but experience shows that it is of sufficient fidelity to be useful.

In this paper we describe the use of a descriptive data cube as part of standard metadata for such alert repositories. The concept is to hash the address of observables to a limited set of coordinates, and represent intensity of observables on a two-dimensional display, with time as a third dimension. We introduce the EMERALD WholeNet Viewer, a tool presently capable of consuming data in a variety of formats such as native firewall logs or NetFlows. The capability to analyze these data sources means that the viewer has utility at levels from enterprise to peering point.

Observables amenable to analysis and visualization by our techniques include packets, rejected packets, IDS alert signatures, and Netflow summaries.

The mapping preserves a notion of address proximity, so that sources that are close in IP address space hash to close coordinates in our display. We typically represent source on the vertical axis, and target (or other dimension of interest) on the horizontal axis.

By considering time as the third dimension, we are able to provide a data cube summary of an arbitrarily large data set. In principle it is possible to present the entire Internet in the viewer. A typical representation we have used maps address space to 256 coordinates, and takes a time slice every thirty minutes. A day of such data is thus represented as a data cube of dimension 256 x 256 x 48, with each intensity represented as a byte (256 level gray scale). The total cube size is a comparatively modest 3 MB. The cube may be assembled in real time or in batch mode, the latter requiring only one comparatively costly pass through the dataset.

The viewer permits a wide variety of replay, step, frame rate, and pane size controls. An analyst will typically display such a cube at a nominal rate of one frame every two seconds using the viewer, where a framer aggregates data for half an hour. This permits the analyst to scan a day of data in a little over a minute and a half.

There are numerous benefits from including a data cube of this type as part of the standard metadata for large repositories. The analyst may quickly screen datasets to identify time periods or events of likely interest for further investigation. This screening uses the metadata cube only. An additional benefit is that the address mapping inherently preserves anonymity of potentially sensitive data in the repository.

For the remainder of this paper, we will use the notation $CUBE(x,y,t)$ to represent the intensity of the observable at mapped source address *x*, mapped destination address *y*, and time slice *t*.

To analyze traffic between nodes, we can use the same or a different mapping of addresses for the horizontal and vertical axes. Notionally, therefore, an analyst may map the entire Internet to the vertical axis and a class C network segment to the horizontal axis, where the horizontal axis represents one node per coordinate.

Scalable Visualization

We present techniques to view propagating network phenomena as analogous to image processing, where multi-scale processing techniques and high-performance hardware coprocessors are more highly developed. We arbitrarily map sources to the vertical axis and destinations to horizontal. The techniques presented here are suitable for detection at peering points and are inherently scalable and suitable to distributed computation.

The basis of our approach, building on our previous work [Va04a, Va04b], is to represent the networks under examination as images, where coordinates are obtained by a suitable mapping function of the address space. At peering points, the source and destination address space may be considered the same. At gateways between the Internet and the enterprise, we can represent sources from the network and destinations in the enterprise using a different address mapping function for each.

We generated such images for source IP address/destination IP address and source IP address/destination port. In our earlier work, the intensity of the pixels corresponds to the count of connections rejected by our firewall for the source and target (IP address or port). We have since enhanced our capability to comprehend Pix Firewall log format and NetFlow data, as well as IDS alerts.

To obtain images 256 pixels on a side, the source and destination IPv4 addresses were hashed into an unsigned byte using the following algorithm:

```
result = address_byte[0];
for (i = 1; i < num_IPv4_bytes; i++)
    result = leftShift (result, 1) ^ address_byte[i];

result = result % 256;
```

Here, leftShift () performs a 1 bit left-shift. Shifting by a different number of bits enables generation of different image sizes. The properties of this hashing algorithm include byte-order sensitivity and the use of most of the input bits.

The 2-byte port value is hashed with a specialized hash function that splits the dimension in half, with the first half containing the port numbers less than 1024 (the IANA reserved port range) and the second half containing the remainder. In both cases, the port is hashed using the modulus function, which has the relevant property that values less than 128 are maintained without modification, enabling the easy identification of various well-known services (e.g., SMTP, FTP, HTTP).

Value Tracking

Value tracking permits the analyst a “drill down” capability for events of interest. Rather than maintaining all observed values contributing to an event (where events generally manifest as features on the image displays), we employ an approach that captures the values most likely responsible.

Associated with each hashed value we optionally track count and value for the largest contributors (for example, most active source addresses hashing to a particular cell) in the form of a truncated histogram (truncated in the sense that only the largest contributors are retained). While in practice a very large number of values might hash into the same index, in practice the activity at a given index is dominated by one value. Coupled with the zoom capability described below, we are able to ascertain the true value for events of interest in most cases.

It must be pointed out that with value tracking enabled the actual values for key observables (such as source and destination IP addresses) are available to the analyst, which may have confidentiality repercussions.

It is also the case that for peering point data we may have a large number of values with comparable counts, which may lead to histograms with a comparatively large number of entities. This may make it difficult to determine the perpetrator of an event of interest. While this is potentially a problem, we have not observed a serious impact with the limited peering point data we have examined.

Furthermore, it is the case that the number of histograms to be maintained is proportional to a single dimension of the cube (typically 256), not the square of this dimension. For IP addresses, one approach to state space management might be to track address values only at the “/24” level.

In the section on future work, we present a pruning and aging procedure to bound the size of the histograms during value tracking.

Value tracking must be enabled to permit the zooming feature described below.

EMERALD WholeNet Viewer

The WholeNet Viewer is a Java application to display network log data in two-dimensional gray-scale intensity plots, where the user may assign various hash-function algorithms to be applied to the data in each dimension. Currently, the user may select from source address and/or port, destination address and/or port, and IDS alert signature number for the axes. The viewer reads raw data in such formats as Pix Firewall Log messages or Netflows, and optionally saves and reloads data cubes.

As many as six simultaneous plots (referred to as “slots” in the GUI) may be displayed. The underlying data for the image consists of the record count for each hashed cell as determined by the parameters. The intensity (darkness) of the displayed pixel at any location is proportional to this count.

The data display consists of the two-dimensional image with a y-axis histogram on the left and an x-axis histogram below it. Below that are controls to play, pause, and rewind, and to the right of the controls is a text box that is used to display information about the frame being viewed. The user has options to invert (reverse black and white) the image or either histogram, compute logarithm before gray scale binning, remove the row-wise minimum (a form of background noise removal useful for enhancing horizontal scans), and zooming.

To zoom, the analyst selects a rectangular region over the data area, which is then expanded to fill the visible data area. Left-clicking a single data point changes the text-box content to display information about the data point (when the value-tracking checkboxes are selected, the corresponding values that contributed to the data point are displayed).

The analyst controls how the input data records are to be used to create the two-dimensional gray-scale plots for one slot through numerous settable parameters. These parameters define the data source for each dimension and the behavior of the hashing function and data collection for that dimension. There are also settable parameters that control how long a time interval (in data units) is consolidated per displayed frame.

The movie replay controls allow the analyst to set how long to display each frame (not to be confused with the data time interval per frame previously mentioned), as well as functions to pause, step, and back up over interesting sections.

The input data may arrive in a variety of formats, or may be loaded from a previously saved data cube.

Firewall Data

We processed log files from the outward-facing PIX firewall at our laboratory for the display in Figure 1. The underlying data consists of packets directed at unused address space (black hole data), and is thus inherently suspicious. Figure 1 shows a single time step of thirty minutes (a typical value for frame replay) for data from Jan 23-24, 2005.

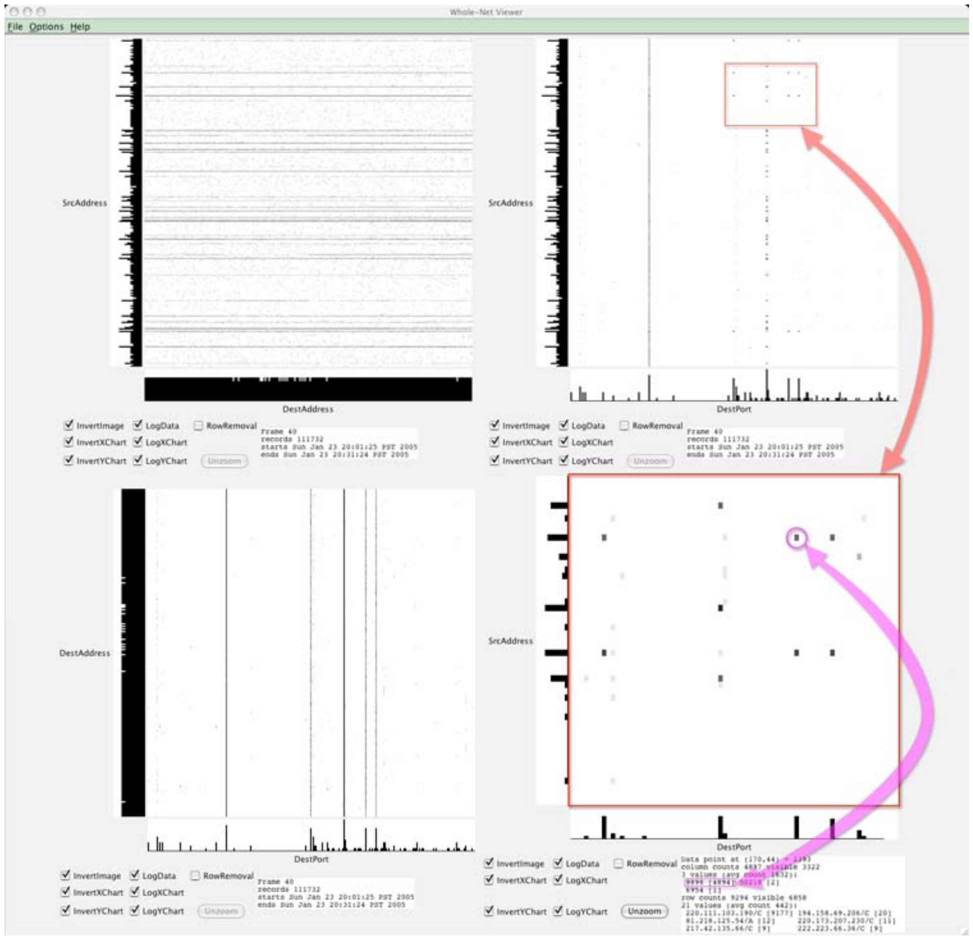
The display shows four panels, in which the rate of rejected packets is displayed in gray scale (darker values indicating a higher rate). Along the left and bottom edges of the display we also provide a histogram. For the figure shown, we observe that there is a "floor" of scanning sources but also a much smaller number of highly active sources, which show up as single bars in the left-edge histogram on the top two panels. The target address range, on the other hand, is comparatively uniform.

The top left panel shows source address to destination address. The horizontal features correspond to source addresses scanning the entire destination address space. The analyst can obtain the value of, say, ports in a bin by clicking on a pixel, usually after selecting a rectangular region and zooming.

The top right panel shows source address to destination port. Target ports are much more concentrated, with this interval showing a vertical feature (corresponding to a large number of sources) scanning for ports 445 and 1023. At 12:30 PST (frame 25), port 445 becomes dominant, and the rate of rejected packets approximately triples. At 16:00 PST (frame 32), activity on port 1433 becomes pronounced but diminishes around 19:00 PST (frame 62). However, unlike activity on port 445, which comes from a wide variety of source IP addresses, port 1433 activity is restricted to 30 or 40 IP bins (out of 256). Concurrent with this activity, at Jan 23rd 20:00 PST (frame 40), the port triplet (1023, 9898, 5554), reflecting the Sasser.E worm, becomes noticeable in a maximum of 4 source IP bins. This port triplet is randomly observed until approximately Jan 24, 3:00 PST (frame 54).

The lower left panel shows destination address versus destination port. The features at ports 445 and 1023 are once again evident, from which the analyst would infer that these are target ports for a large number of scanners. The triplet feature of the upper right panel appears as three vertical lines in the lower left, indicating that the sources scanning for this triplet port pattern look for it over the entire target address space.

The lower right panel is a zoom of the rectangular region identified in the upper right panel. By clicking on the individual pixels, the analyst can ascertain what the actual ports are. For example, the selected hashed pixel represents 4894 scans for port 9898, two scans for port 50218, and one for port 6954 (these ports hashed into the same index, but the counts are dominated by 9898). By examining the other pixels in the triplet, we establish that these four sources are scanning in a manner consistent with Sasser.



Upper left panel:
✓ InvertImage ✓ LogData ☐ RowRemoval
✓ InvertXChart ✓ LogXChart
✓ InvertYChart ✓ LogYChart

```
Frame 40
records 111732
starts Sun Jan 23 20:01:25 PST 2005
ends Sun Jan 23 20:31:24 PST 2005
```

Upper right panel:
✓ InvertImage ✓ LogData ☐ RowRemoval
✓ InvertXChart ✓ LogXChart
✓ InvertYChart ✓ LogYChart

```
Frame 40
records 111732
starts Sun Jan 23 20:01:25 PST 2005
ends Sun Jan 23 20:31:24 PST 2005
```

Lower left panel:
✓ InvertImage ✓ LogData ☐ RowRemoval
✓ InvertXChart ✓ LogXChart
✓ InvertYChart ✓ LogYChart

```
Frame 40
records 111732
starts Sun Jan 23 20:01:25 PST 2005
ends Sun Jan 23 20:31:24 PST 2005
```

Lower right panel:
✓ InvertImage ✓ LogData ☐ RowRemoval
✓ InvertXChart ✓ LogXChart
✓ InvertYChart ✓ LogYChart

```
Data point at (170,44) = 239)
column counts 4897 visible 3322
3 values (avg count 1632):
 9898 [4894] 50218 [2]
 6954 [1]
row counts 9294 visible 6858
21 values (avg count 442):
 220.111.103.190/C [9177] 194.158.69.206/C [20]
 81.210.125.54/A [12]     220.173.207.230/C [11]
 217.42.135.66/C [9]      222.223.66.36/C [9]
```

Figure 1. EMERALD WholeNetViewer Console

NetFlow Data

We preprocessed data from Abilene networks for one site (IPLSng) [AB05] via the flow-export module from flow-tools [FT05]. The output of this module was piped to an awk script, which extracted the timestamp, IP protocol, source and destination IP addresses and ports, number of packets, total size of packets, and duration. This result was then saved as a compressed/gzipped text file compatible with the WholeNet Viewer.

The data are all NetFlows seen at the site, not necessarily limited to suspicious flows.

We downloaded data for January 16, 2005, principally for reasons of data availability and not because of any particularly interesting phenomena of which we were aware. We did observe some apparent port scan activity.

Our java app to generate the data cube processes the Abilene data at about 34,000 records per second on a dual processor Macintosh G5 (1.8 GHz).

Correlation

The data cube enables correlation across time slices, correlation from one data cube to the next, and the special case of correlation to detect bidirectional flows (for example, an analyst may be especially interested in nodes that apparently respond to probes).

Correlation is approximated by multiplication of the intensity at a particular pair of cube coordinates. The point-wise product may then be normalized by the number of points under consideration to provide a coarse correlation estimate for two cubes.

The approximate correlation of two time slices within the same data cube is proportional to

$$CORR(t_1,t_2) \propto \sum_{x,y} CUBE(x,y,t_1) \times CUBE(x,y,t_2)$$

Computing this at various time lags provides a temporal autocorrelation function. This is defined pixel by pixel, or a rough autocorrelation can be computed by summing all pixel-wise correlations over the entire cube.

It is possible to correlate time slices in different data cubes, or to correlate two different data cubes. This makes sense only if both cubes use the same value map. Correlation of different data cubes (or portions thereof) is a rough measure of the similarity of phenomena in the two cubes. This permits the analyst to answer, at least in an approximate sense, whether phenomenology observed on two days is similar, or whether something similar to the present observation has been observed in the past. The correlation between cubes j and k (corresponding to, for example, days j and k in some repository) is proportional to

$$CORR(j,k) \propto \sum_{x,y,t} CUBE_j(x,y,t) \times CUBE_k(x,y,t)$$

The analyst may query the repository to identify all days that are similar (above some correlation threshold) to a cube of interest, and the computation is performed on the meta data only. The analyst may seek to identify days or time periods that correlate to a given data cube (that is, a cube containing an idealized representation of some phenomenology of interest, but does not represent any actual data).

In the case of the same mapping, the display shows which nodes connect to which. If directionality is maintained (which node initiates and responds to the TCP connection), a correlation of the figure with its transpose (achieved by simple multiplication and scaling) indicates which nodes are responding to apparent probes. Probes are ubiquitous today, and analysts have expressed a need to identify probes for which vulnerable nodes respond. Our technique presents a computationally efficient means to accomplish this.

Using a different map for source and destination, the analyst can, for example, represent the address space under his administration on the horizontal axis at much finer grain. In fact, using a display 256 x 256 pixels per time slice can represent an entire 256 node LAN segment on the target axis (horizontal in our convention).

Figure 2 below displays the autocorrelation at lag one of the Kuang data set previously described. Each frame represents 30 minutes of data. The lag one autocorrelation is computed on sliding windows of 10 frames, so the output at time 1 is the lag one autocorrelation of frames 1-10 and 2-11, the next output is analogous for frames 2-11 and 3-12, and so forth. The underlying image on which the autocorrelation is based is the source to destination (external to internal) address map. We note that this data is effectively de-correlated at half hour frames during periods of normal activity (autocorrelation near zero), with autocorrelation increasing to about 0.2 during the frames of the Kuang event. Whether the de-correlation timescale of this traffic is typical of Internet traffic as a whole requires analysis of more data.

Once again we emphasize the scalability advantage of our representation. The computational complexity of this early correlation analysis is independent of the traffic volume and the address space being monitored.

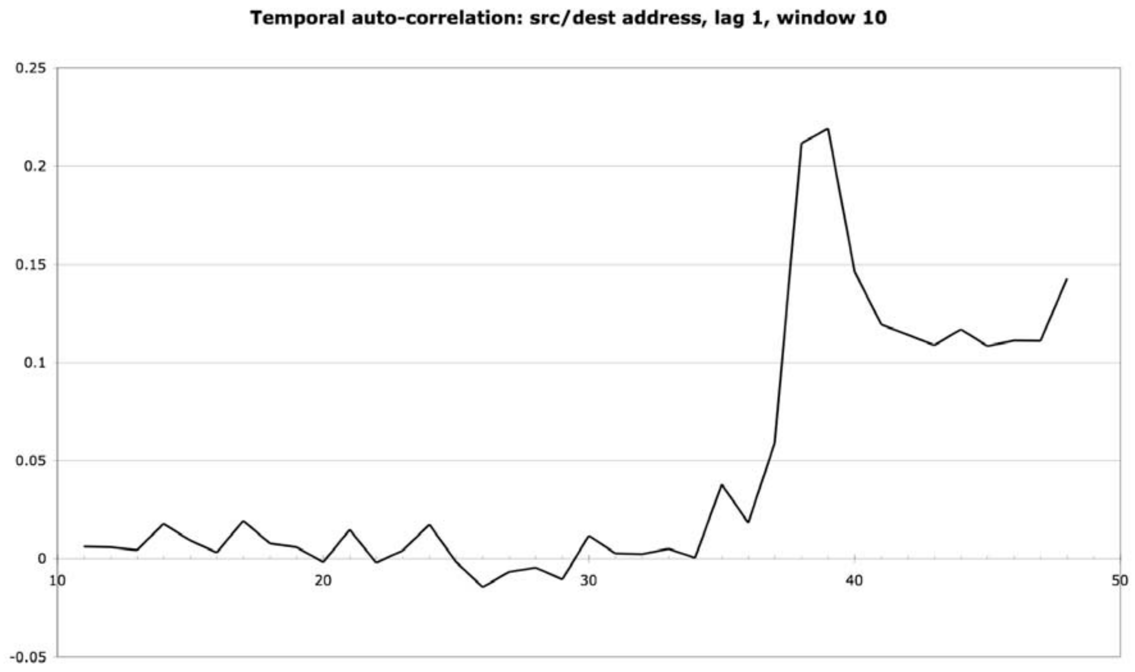


Figure 2: Lag 1 Autocorrelation of Kuang Data Set

Alternate Views

The data represented on the horizontal axis need not be the same as that on the vertical. For example, Figure 1 shows port on the horizontal axis.

We have found it useful to examine IDS alerts in this way. These displays allow one to detect in near real time a large-scale change in the IDS alert mix. We have observed such changes in the case of large-scale propagating events. Furthermore, IDS alerts from the EMERALD Net appliance's Asset Distress Monitor can symptomatically detect events for which there is no IDS signature.

Finally, we have developed a compound map that simultaneously displays, for example, destination IP in bands of a fixed number of pixels and ports to pixels within the band, so that a given pixel in any band corresponds to the same mapped ports (Figure 3). This induces a frequency structure in the display, which is amenable to power spectrum and other frequency transform techniques. For example, attacks hitting the same pattern of ports will appear as energetic bands in the power spectrum. Frequency space techniques have long been in use in the image processing community, and highly efficient algorithms and hardware coprocessors can be brought to bear.

We can generate the displays from time slices and/or decayed moving averages. This permits near-real-time animation displays, and also ensures modest computational and memory requirements for image generation.

The displays can track the raw observable, or in a variant display the data after noise removal. Noise is estimated by continuous normalization of the underlying data (subtracting a recursive estimate of the mean and dividing by the standard deviation). If the attack process is stationary in the statistical sense, the display will appear as white noise (salt and pepper). Emerging attacks are often evident sooner in such a noise-removed display as they disrupt the underlying stationarity. This approach is effective at cuing the analyst to the novel and suspicious.

Future Work

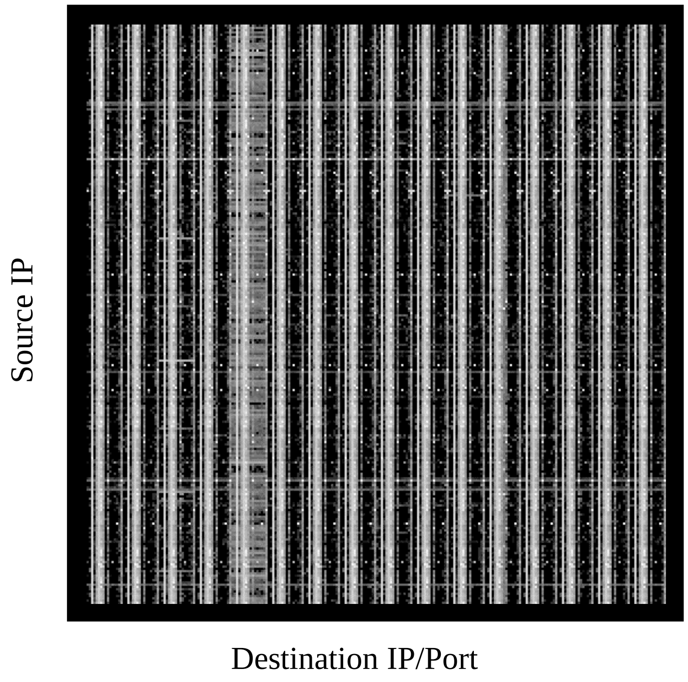


Figure 3. Source IP/Composite Destination IP/Port display, demonstrating induced periodicity

Detrending and Detection of Inflections

A moving average estimate is computed on the fly and removed from a cube cell value. A variant is to compute moving averages with different aging constants. The difference of the short-term and long-term value per cell may indicate an inflection in the process (for example, the early stages of a propagating event). An inflection may be more evident in a de-trended cube than in the unmodified cube.

The moving average *Mov_Avg* at the initial time slice is simply the cube value. Moving averages and de-trended cube values at subsequent time slices are computed as follows.

// Input : $aging\_const \in (0,1]$

$$Mov\_Avg(x,y,t_2) = aging\_const \times Mov\_Avg(x,y,t_1) + (1 - aging\_const) \times Cube(x,y,t_2)$$

$$Detrended\_Cube(x,y,t_2) = Cube(x,y,t_2) - Mov\_Avg(x,y,t_2)$$

Our initial analysis shows that detrending emphasizes the “loud emitters”, which is usually desirable, but that these same loud emitters sometimes overwhelm the short term trend. When they cease their activity, detrending introduces a negative artifact. This situation appears to be analogous to signal capture in the field of signal processing. We need to look further into algorithms for correcting the trend for these emitters.

Pruning and Process Memory

We will employ a periodic prune-and-age procedure to mitigate state space explosion in value tracking by pruning very rare categories and also to down weight observations in the past while giving comparatively more emphasis to recent observations.

The frequency of calls to the prune-and-age process, the drop threshold, and the aging constant used are all algorithm parameters that may be adjusted. For large NetFlow data sets, such as from Abilene, we have a record for each flow that counts the number of packets between two addresses in a five-minute interval.

Pruning and aging tend to retain states that are more active or have been active more recently, which are likely to be the more interesting states. A pruned category that is subsequently observed again is simply appended as a new category with no adverse effect. The number of values tracked is theoretically limited to the reciprocal of the pruning probability, and is usually much less.

Pruning

```
$Input$: Threshold probability for pruning $p_t$
// Compute threshold count for pruning
$count_t = N * p_t$;
// Traverse value list pruning categories with counts below threshold
for(all values $i$){
if $(n_i < count_t)${
    $N = N - n_i$;
    $\mathrm{N}_{\mathrm{CAT}} = \mathrm{N}_{\mathrm{CAT}} - 1$;
    < release category $i$ >
}
}
```

At this point the pruned category list may be sorted for optimized list searches.

Aging

The aging procedure ages category and total effective (that is, aged) counts.

```
$Input$:  Aging factor, $0 < age\_factor \le 1$
// $N =$ Effective number of observations
// $n_i =$ Effective count of observations of value i
// Age the total observation count
$N = N * age\_factor;$
// Traverse list. Age value counts
for (all values $i$) {
    $n_i = n_i * age\_factor;$
}
```

From Representation to Detection

The observed patterns in actual data motivate an approach using image analysis techniques to detect phenomena of interest. Specifically, a horizontal scan appears as a horizontal feature in the source IP to destination IP view. A distributed scan to a single port appears as a vertical feature in the source IP to destination port view. If the scans come from a bot net with "close" addresses (that is, addresses that hash to near values via the algorithm previously presented), we may hypothesize that a bot scanning for a vulnerable port will appear as Figure 3 below.

A matched filter is an idealized representation of a feature that is to be searched in some image. The matched filter is convolved with the image, after both have been subjected to a fast Fourier transform (FFT). In transform space, convolution is simply pixel-by-pixel multiplication. The inverse transform then provides a matched filter field highlighting regions where the original image is similar to the matched filter.

We will explore application of matched filters to look for features of this type in the data cubes.

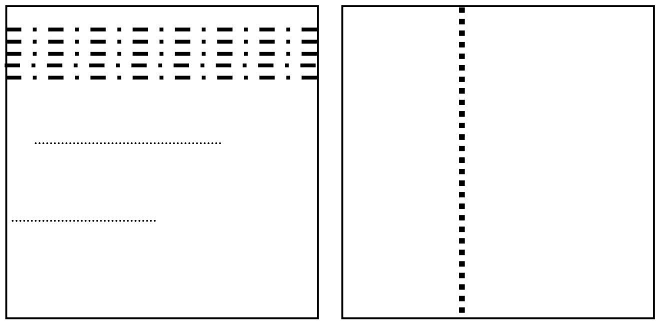


Figure 4. Hypothesized appearance of Bot Scan in Source IP/Dest IP and Source IPDest port views

Related Work

Staniford and his collaborators [Sta01, Sta02] have done important foundational work in the theoretical dynamics of propagating phenomena such as worms. Their results raise serious concerns for the Internet community in demonstrating that worm episodes can saturate the vulnerable population in an interval of time that is far shorter than the advisory/detect/patch response cycle of today.

Moore et al. [Mo03] provides a comprehensive analysis of the propagation dynamics of the Slammer worm. This attack quickly infected most of the vulnerable population on the Internet. Moreover, its propagation had a significant impact on the Internet as a whole, dominating Internet-wide UDP traffic during the period of maximum activity.

Burnett [Bu03] proposes displaying resource usage counters with the Multi Router Traffic Grapher (MRTG), noting that many attacks manifest as changes in graphic traces of these counters, which the user identifies visually. The distribution of certain count-based features such as the return code from a Web server may be suitable to the methods presented here.

Recent work in reversible sketches [Sch04] also addresses the issue of detection in large-scale networks by hashing of the address space. The use of reversible sketches presents a potential alternative to the value tracking mechanism we have employed.

May [Ma01] detects attacks in large-scale networks by exploiting emergent properties that change statistically between normal and attack states. He employs visualization techniques for large networks, although they are different from what we consider here.

Summary

We have proposed a compact data cube as a meta data summary of very large Infosec alert repositories. We have also implemented a tool, the EMERALD WholeNet Viewer, to generate, save, and replay these cubes. We have outlined future directions whereby this capability can be extended for efficient correlation through cell-wise multiplication, and efficient detection through frequency space methods and matched filtering techniques. The viewer is implemented in Java and has been demonstrated with data formats relevant at levels from the enterprise to the peering point.

The key benefits are near-real-time visualization of interesting events, a coarse-grained but meaningful representation of extremely large data sets enabling rapid identification of regions of interest, and rapid coarse-grained correlation of interesting events.

Correlation is efficiently achieved by multiplication and scaling. We have previously described correlation of an image in which source and destination are identically mapped. In that case, potentially vulnerable destinations responding to suspicious probes are evident by correlating the image with its transpose. In general, we link an image (or a video clip of an image animation sequence) with the large files in the repository. A quick visual examination permits the analyst to identify the time slices and address ranges of interest, as desired. Also, an analyst can quickly explore if, for example, a current observation correlates with a past event.

This new approach offers an opportunity to produce an ongoing display, noise filtering, and graphical representation of observables such as incoming alerts and firewall log, giving analysts the ability to visually interpret what is happening on the Internet or in their networks. This could translate into “live” event detection, meaning faster reporting back to the victims.

References

[AB05] Abilene Networks, http://abilene.internet2.edu/

[Bu03] Burnett, Mark. "MRTG for Intrusion Detection with IIS 6", http://www.securityfocus.com/infocus/1721

[FT05] Flow Tools, http://www.splintered.net/sw/flow-tools/

[Ma01] May, J., Peterson, J., and Bauman, J. "Attack Detection in Large Networks". Proceedings of the Second DARPA Information Security Conference and Exposition (DISCEX II), Anaheim, CA, June 2001.

[Mo03] Moore, D., Paxson, V., Savage, S., Shannon, Colleen, Staniford, S., and Weaver, N. "The Spread of the Sapphire/Slammer Worm", www.cs.berkeley.edu/~nweaver/sapphire, 2003.

[Sch04] SCHWELLER, R., GUPTA, A., PARSONS, E., AND CHEN, Y. Reversible sketches for efficient and accurate change detection over network data streams. In Proc. of ACM SIGCOMM IMC (2004)

[Sta01] Staniford, S, Grim, G., Jonkman, R. "Flash Worms: Thirty Seconds to Infect the Internet", http://www.silicondefense.com/flash/

[Sta02a] Staniford, S., Paxson, V., and Weaver, N. "How to Own the Internet in Your Spare Time", Proceedings of the 11th USENIX Security Symposium, 2002.

[Va04a] Valdes, A. and Fong, M. "Scalable, Signature-Free Characterizations of Propagating Internet Phenomena", Fast abstract presented at Dependable Systems and Networks (DSN04), Florence, Italy, July 2004.

[Va04b] Valdes, A. and Fong, M. "Scalable Visualization of Propagating Internet Phenomena", ACM CCS Workshop on Data Mining and Visualization in Computer Security, Fairfax, VA, October 2004.