Seurat: A Pointillist Approach to Anomaly Detection

Yinglian Xie[1], Hyang-Ah Kim[1],
David R. O'Hallaron[1,2], Michael K. Reiter[1,2], and Hui Zhang[1,2]

[1] Department of Computer Science
[2] Department of Electrical and Computer Engineering
Carnegie Mellon University
{ylxie,hakim,droh,reiter,hzhang}@cs.cmu.edu

Abstract. This paper proposes a new approach to detecting aggregated anomalous events by correlating host file system changes across space and time. Our approach is based on a key observation that many host state transitions of interest have both temporal and spatial locality. Abnormal state changes, which may be hard to detect in isolation, become apparent when they are correlated with similar changes on other hosts. Based on this intuition, we have developed a method to detect similar, coincident changes to the patterns of file updates that are shared across multiple hosts. We have implemented this approach in a prototype system called *Seurat* and demonstrated its effectiveness using a combination of real workstation cluster traces, simulated attacks, and a manually launched Linux worm.

Keywords: Anomaly detection, Pointillism, Correlation, File updates, Clustering

1 Introduction

Correlation is a recognized technique for improving the effectiveness of intrusion detection by combining information from multiple sources. For example, many existing works have proposed correlating different types of logs gathered from distributed measurement points on a network (e.g., [1–3]). By leveraging collective information from different local detection systems, they are able to detect more attacks with fewer false positives.

In this paper, we propose a new approach to anomaly detection based on the idea of correlating host state transitions such as file system updates. The idea is to correlate host state transitions across both space (multiple hosts) and time (the past and the present), detecting similar coincident changes to the patterns of host state updates that are shared across multiple hosts. Examples of such coincident events include administrative updates that modify files that have not been modified before, and malware propagations that cause certain log files, which are modified daily, to cease being updated.

Our approach is based on the key observation that changes in host state in a network system often have both temporal and spatial locality. Both administrative updates and malware propagation exhibit spatial locality, in the sense that similar updates tend to occur across many of the hosts in a network. They also exhibit temporal locality in the sense that these updates tend to be clustered closely in time. Our goal is to identify atypical such aggregate updates, or the lack of typical ones.
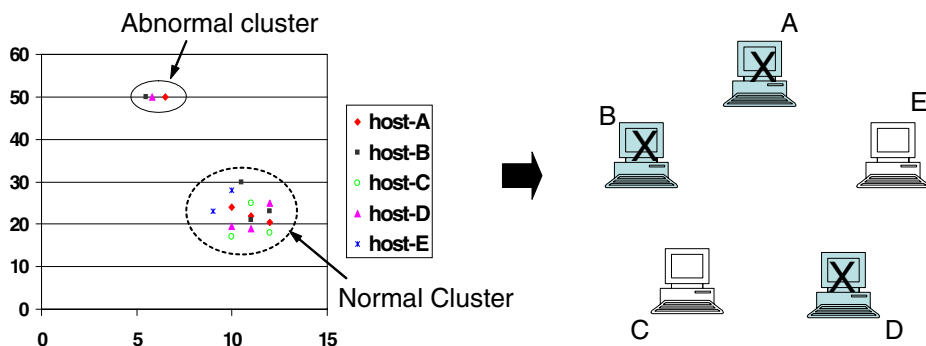
Fig. 1. Pointillist approach to anomaly detection: Normal points are clustered by the dashed circle. The appearance of a new cluster consisting of three points suggests anomalous events on host A, B, and D.

By exploring both the temporal and spatial locality of host state changes in a network system, our approach identifies anomalies *without foreknowledge of normal changes* and *without system-specific knowledge*. Existing approaches focus on the temporal locality of host state transitions, while overlooking the spatial locality among different hosts in a network system. They either define a model of normal host state change patterns through learning, or specify detailed rules about normal changes. The learning based approaches train the system to learn characteristics of normal changes. Since they focus only on the temporal locality of single-host state transitions, any significant deviation from the normal model is suspicious and should raise an alarm, resulting in a high false positive rate. Rule-based approaches such as Tripwire [4] require accurate, specific knowledge of system configurations and daily user activity patterns on a specific host. Violation of rules then suggests malicious intrusions. Although rule-based anomaly detection raises fewer false alarms, it requires system administrators to manually specify a set of rules for each host. The correlation capability of our approach across both space and time allows us to learn the patterns of normal state changes over time, and to detect those anomalous events correlated among multiple hosts due to malicious intrusions. This obviates the need for specific rules while eliminating the false alarms caused by single host activity pattern shifts.

The correlation is performed by clustering points, each representing an individual host state transition, in a multi-dimensional feature space. Each feature indicates the change of a file attribute, with all features together describing the host state transitions of an individual machine during a given period (e.g., one day). Over time, the abstraction of point patterns inherently reflects the aggregated host activities. For normal host state changes, the points should follow some regular pattern by roughly falling into several clusters. Abnormal changes, which are hard to detect by monitoring that host alone, will stand out when they are correlated with other normal host state changes. Hence our approach shares some flavor of *pointillism* – a style of painting that applies small dots onto a surface so that from a distance the dots blend together into meaningful patterns.

Figure 1 illustrates the pointillist approach to anomaly detection. There are five hosts in the network system. We represent state changes on each host daily as a point in a 2-dimensional space in this example. On normal days, the points roughly fall into

the dash-circled region. The appearance of a new cluster consisting of three points (indicated by the solid circle) suggests the incidence of anomaly on host A, B, and D, which may all have been compromised by the same attack. Furthermore, if we know that certain hosts (e.g., host A) are already compromised (possibly detected by other means such as a network based IDS), then we can correlate the state changes of the compromised hosts with the state changes of all other hosts in the network system to detect more infected hosts (e.g., host B and D).

We have implemented a prototype system, called *Seurat*[1], that uses file system updates to represent host state changes for anomaly detection. Seurat successfully detects the propagation of a manually launched Linux worm on a number of hosts in an isolated cluster. Seurat has a low false alarm rate when evaluated by a real deployment. These alarms are caused by either system re-configurations or network wide experiments. The false negative rate and detection latency, evaluated with simulated attacks, are both low for fast propagating attacks. For slowly propagating attacks, there is a tradeoff between false negative rate and detection latency. For each alarm, Seurat identifies the list of hosts involved and the related files, which we expect will be extremely helpful for system administrators to examine the root cause and dismiss false alarms.

The rest of the paper is organized as follows: Section 2 describes Seurat threat model. Section 3 introduces the algorithm for correlating host state changes across both space and time. Section 4 evaluates our approach. Section 5 discusses the limitations of Seurat and suggests possible improvements. Section 6 presents related work.

2   Attack Model

The goal of Seurat is to automatically identify anomalous events by correlating the state change events of all hosts in a network system. Hence Seurat defines an anomalous event as an unexpected state change close in time across *multiple* hosts in a network system.

We focus on rapidly propagating Internet worms, virus, zombies, or other malicious attacks that compromise multiple hosts in a network system at a time (e.g., one or two days). We have observed that, once fast, automated attacks are launched, most of the vulnerable hosts get compromised due to the rapid propagation of the attack and the scanning preferences of the automated attack tools. According to CERT's analysis [5], the level of automation in attack tools continues to increase, making it faster to search vulnerable hosts and propagate attacks. Recently, the Slammer worm hit 90 percent of vulnerable systems in the Internet within 10 minutes [6]. Worse, the lack of diversity in systems and softwares run by Internet-attached hosts enables massive and fast attacks. Computer clusters tend to be configured with the same operating systems and softwares. In such systems, host state changes due to attacks have strong temporal and spatial locality that can be exploited by Seurat.

Although Seurat is originally designed to detect system changes due to fast propagating attacks, it can be generalized to detect slowly propagating attacks as well. This can be done by varying the time resolution of reporting and correlating the collective host state changes. We will discuss this issue further in Section 5. However, Seurat's

---

[1] Seurat is the 19th century founder of pointillism.

global correlation can not detect abnormal state changes that are unique to only a single host in the network system.

Seurat represents host state changes using *file system updates*. Pennington et al. [7] found that 83% of the intrusion tools and network worms they surveyed modify one or more system files. These modifications would be noticed by monitoring file system updates. There are many security tools such as Tripwire [4] and AIDE [8] that rely on monitoring abnormal file system updates for intrusion detection.

We use the file name, including its complete path, to identify a file in the network system. We regard different instances of a file that correspond to a common path name as a same file across different hosts, since we are mostly interested in system files which tend to have canonical path names exploited by malicious attacks. We treat files with different path names on different hosts as different files, even when they are identical in content.

For the detection of anomalies caused by attacks, we have found that this representation of host state changes is effective and useful. However, we may need different approaches for other applications of Seurat such as file sharing detection, or for the detection of more sophisticated future attacks that alter files at arbitrary locations as they propagate. As ongoing work, we are investigating the use of file content digests instead of file names.

3   Correlation-Based Anomaly Detection

We define a d-dimensional feature vector $\boldsymbol{H}_{ij} = \langle v_1, v_2, \ldots, v_d \rangle$ to represent the file system update attributes for host i during time period j. Each $\boldsymbol{H}_{ij}$ can be plotted as a point in a d-dimensional feature space. Our pointillist approach is based on correlating the feature vectors by clustering. Over time, for normal file updates, the points follow some regular pattern (e.g., roughly fall into several clusters). From time to time, Seurat compares the newly generated points against points from previous time periods. The appearance of a new cluster, consisting only of newly generated points, indicates abnormal file updates and Seurat raises an alarm.

For clustering to work most effectively, we need to find the most relevant features (dimensions) in a feature vector given all the file update attributes collected by Seurat. We have investigated two methods to reduce the feature vector dimensions: (1) *wavelet-based selection*, and (2) *principal component analysis* (PCA).

In the rest of this section, we first present how we define the feature vector space and the distances among points. We then describe the methods Seurat uses to reduce feature vector dimensions. Finally, we discuss how Seurat detects abnormal file updates by clustering.

3.1   Feature Vector Space

Seurat uses binary feature vectors to represent host file updates. Each dimension in the feature vector space corresponds to a unique file (indexed by the full-path file name). As such, the dimension of the space d is the number of file names present on any machine in the network system. We define the *detection window* to be the period that we are

interested in finding anomalies. In the current prototype, the detection window is one day. For each vector $\boldsymbol{H}_{ij} = \langle v_1, v_2, \ldots, v_d \rangle$, we set v_k to 1 if host i has updated (added, modified, or removed) the k-th file on day j, otherwise, we set v_k to 0.

The vectors generated in the detection window will be correlated with vectors generated on multiple previous days. We treat each feature vector as an independent point in a set. The set can include vectors generated by the same host on multiple days, or vectors generated by multiple hosts on the same day. In the rest of the paper, we use $\boldsymbol{V} = \langle v_1, v_2, \ldots, v_d \rangle$ to denote a feature vector for convenience. Figure 2 shows how we represent the host file updates using feature vectors.

$$
\begin{array}{ccccccc}
 & & F_1 & F_2 & F_3 & F_4 & F_5 \\
V_1 = H_{11} = & < & 1, & 1, & 0, & 1, & 1 > \\
V_2 = H_{21} = & < & 1, & 1, & 1, & 0, & 0 > \\
V_3 = H_{12} = & < & 1, & 1, & 0, & 1, & 0 >
\end{array}
$$

Fig. 2. Representing host file updates as feature vectors: F_1, F_2, F_3, F_4, F_5 are five different files (i.e., file names). Accordingly, the feature vector space has 5 dimensions in the example.

The correlation is based on the distances among vectors. Seurat uses a cosine distance metric, which is a common similarity measure between binary vectors [9, 10]. We define the distance $D(\boldsymbol{V}_1, \boldsymbol{V}_2)$ between two vectors $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ as their angle θ computed by the cosine value:

$$
D(\boldsymbol{V}_1, \boldsymbol{V}_2) = \theta = cos^{-1}\left(\frac{\boldsymbol{V}_1 \cdot \boldsymbol{V}_2}{|\boldsymbol{V}_1||\boldsymbol{V}_2|}\right)
$$
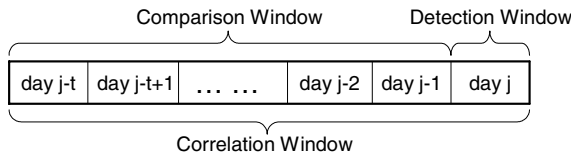
Fig. 3. Detection window, comparison window, and correlation window. The detection window is day j. The comparison window is from day $j - t$ to day $j - 1$. The correlation window is from day $j - t$ to day j.

For each day j (the detection window), Seurat correlates the newly generated vectors with vectors generated in a number of previous days $j - 1, j - 2, \ldots$. We assume that the same abnormal file update events on day j, if any, have not occurred on those previous days. We define the *comparison window* of day j as the days that we look back for comparison, and the *correlation window* of day j as the inclusive period of day j

and its comparison window. Vectors generated outside the correlation window of day j are not used to identify abnormal file updates on day j. Figure 3 illustrates the concepts of detection window, comparison window, and correlation window.

Since each vector generated during the comparison window serves as an example of normal file updates to compare against in the clustering process, we explore the temporal locality of normal update events by choosing an appropriate comparison window for each day. The comparison window size is a configurable parameter of Seurat. It reflects how far we look back into history to implicitly define the model of normal file updates. For example, some files such as `/var/spool/anacron/cron.weekly` on Linux platforms are updated weekly. In order to regard such weekly updates as normal updates, administrators have to choose a comparison window size larger than a week. Similarly, the size of the detection window reflects the degree of temporal locality of abnormal update events.

Since Seurat correlates file updates across multiple hosts, we are interested in only those files that have been updated by at least two different hosts. Files that have been updated by only one single host in the network system throughout the correlation window are more likely to be user files. As such, we do not select them as relevant dimensions to define the feature vector space.

3.2   Feature Selection

Most file updates are irrelevant to anomalous events even after we filter out the file updates reported by a single host. Those files become noise dimensions when we correlate the vectors (points) to identify abnormal updates, and increase the complexity of the correlation process. We need more selective ways to choose relevant files and reduce feature vector dimensions. Seurat uses a wavelet-based selection method and principal component analysis (PCA) for this purpose.
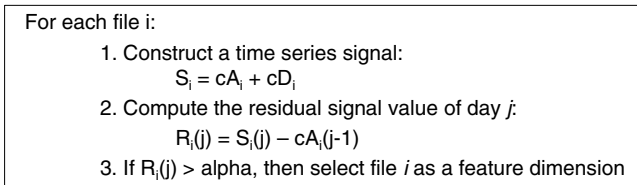
Wavelet-Based Selection. The wavelet-based selection method regards each individual file update status as a discrete time series signal S. Given a file i, the value of the signal
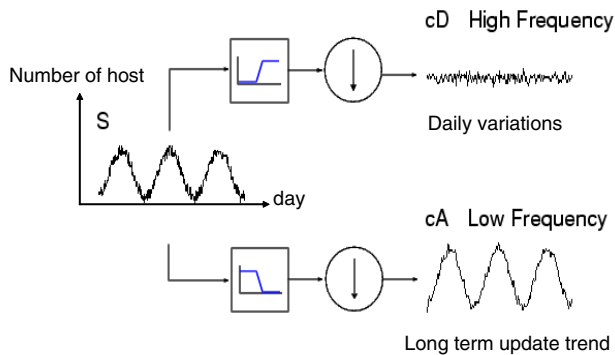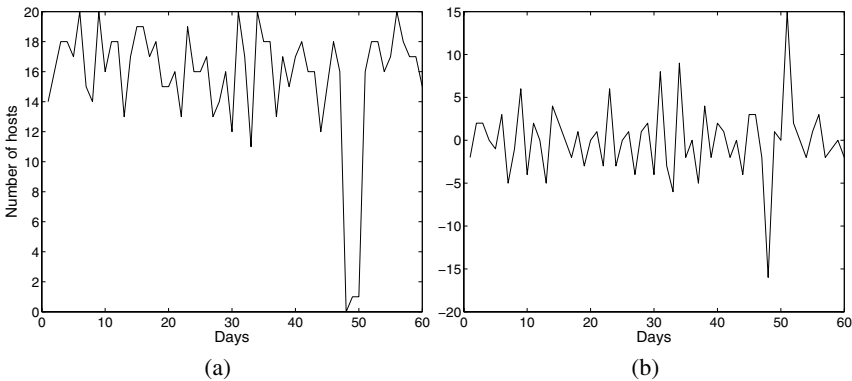

Fig. 4. Representing file update status with wavelet transformation: The original signal is S, which can be decomposed into a low frequency signal cA reflecting the long term update trend, and a high frequency signal cD reflecting the daily variations from the long-term trend.

on day n, denoted by $S_i(n)$, is defined as the total number of hosts that update file i on day n in the network system. Each such signal S_i can be decomposed into a low frequency signal cA_i reflecting the long term update trend, and a high frequency signal cD_i reflecting the day-to-day variation from the long term trend. (see Figure 4). If the high frequency signal cD_i shows a spike on a certain day, we know that a significantly larger number of hosts updated file i than on a normal day. We then select file i as a relevant feature dimension in defining the feature vector space.

Seurat detects signal spikes using the residual signal of the long-term trend. The same technique has been used to detect disease outbreaks[11]. To detect anomalies on day j, the algorithm takes as input the list of files that have been updated by at least two different hosts in the correlation window of day j. Then, from these files the algorithm selects a subset that will be used to define the feature vector space.

For each file i:
  1. Construct a time series signal:
    $S_i = cA_i + cD_i$
  2. Compute the residual signal value of day *j*:
    $R_i(j) = S_i(j) - cA_i(j-1)$
  3. If $R_i(j) > $ alpha, then select file *i* as a feature dimension

---

Fig. 5. Wavelet-based feature selection.

(a)

(b)

Fig. 6. Wavelet transformation of file update status: (a) The original signal of the file update status (b) The residual signal after wavelet transformation.

Figure 5 shows the steps to select features by wavelet-based method. Given a fixed correlation window of day j, the algorithm starts with constructing a time series signal S_i for each file i, and decomposes S_i into cA_i and cD_i using a single-level wavelet transformation as described. Then we compute the residual signal value $R_i(j)$ of day j by subtracting the trend value $cA_i(j-1)$ of day $j-1$ from the original signal value $S_i(j)$ of day j. If $R_i(j)$ exceeds a pre-set threshold α, then the actual number of hosts who have updated file i on day j is significantly larger than the prediction $cA_i(j-1)$

based on the long term trend. Therefore, Seurat selects file i as an interesting feature dimension for anomaly detection on day j. As an example, Figure 6 shows the original signal and the residual signal of a file using a 32-day correlation window in a 22-host teaching cluster. Note the threshold value α of each file is a parameter selected based on the statistical distribution of historical residual values.

PCA-Based Dimension Reduction. PCA is a statistical method to reduce data dimensionality without much loss of information [12]. Given a set of d-dimensional data points, PCA finds a set of d' orthogonal vectors, called *principal components*, that account for the variance of the input data as much as possible. Dimensionality reduction is achieved by projecting the original d-dimensional data onto the subspace spanned by these d' orthogonal vectors. Most of the intrinsic information of the d-dimensional data is preserved in the d'-dimensional subspace.

We note that the updates of different files are usually correlated. For example, when a software package is updated on a host, many of the related files will be modified together. Thus we can perform PCA to identify the correlation of file updates.

Given a d-dimensional feature space $\mathcal{Z}_2^d$, and a list of m feature vectors $\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_m \in \mathcal{Z}_2^d$, we perform the following steps using PCA to obtain a new list of feature vectors $\boldsymbol{V}'_1, \boldsymbol{V}'_2, \ldots, \boldsymbol{V}'_m \in \mathcal{Z}_2^{d'} (d' < d)$ with reduced number of dimensions:

1. Standardize each feature vector $\boldsymbol{V}_k = \langle v_{1k}, v_{2k}, \ldots, v_{dk} \rangle \ (1 \leq k \leq m)$ by subtracting each of its elements v_{ik} by the mean value of the corresponding dimension $u_i (1 \leq i \leq d)$. We use $\overline{\boldsymbol{V}}_k = \langle \overline{v}_{1k}, \overline{v}_{2k}, \ldots, \overline{v}_{nk} \rangle \in \mathcal{Z}_2^d$ to denote the standardized vector for the original feature vector $\boldsymbol{V}_k$. Then,

$$\overline{v}_{ik} = v_{ik} - u_i \ \ (\text{where } u_i = \frac{\sum_{j=1}^{m} v_{ij}}{m}, \ 1 \leq i \leq d)$$

2. Use the standardized feature vectors $\overline{\boldsymbol{V}}_1, \overline{\boldsymbol{V}}_2, \ldots, \overline{\boldsymbol{V}}_m$ as input data to PCA in order to identify a set of principal components that are orthogonal vectors defining a set of transformed dimensions of the original feature space $\mathcal{Z}_2^d$. Select the first d' principal components that count for most of the input data variances (e.g., 90% of data variances) to define a subspace $\mathcal{Z}_2^{d'}$.
3. Project each standardized feature vector $\overline{\boldsymbol{V}}_k \in \mathcal{Z}_2^d$ onto the PCA selected subspace $\mathcal{Z}_2^{d'}$ to obtain the corresponding reduced dimension vector $\boldsymbol{V}'_k \in \mathcal{Z}_2^{d'}$.

Note that PCA is complementary to wavelet-based selection. Once we fix the correlation window of a particular day, we first pick a set of files to define the feature vector space by wavelet-based selection. We then perform PCA to reduce the data dimensionality further.

3.3   Anomaly Detection by Clustering

Once we obtain a list of transformed feature vectors using feature selection, we cluster the vectors based on the distance between every pair of them.

We call the cluster a *new cluster* if it consists of multiple vectors only from the detection window. The appearance of a new cluster indicates possibly abnormal file updates occurred during the detection window and should raise an alarm.

There are many existing algorithms for clustering, for example, K-means [13, 14] or Single Linkage Hierarchical Clustering [10]. Seurat uses a simple iterative algorithm, which is a common method for K-means initialization, to cluster vectors without prior knowledge of the number of clusters [15]. The algorithm assumes each cluster has a hub. A vector belongs to the cluster whose hub is closest to that vector compared with the distances from other hubs to that vector. The algorithm starts with one cluster whose hub is randomly chosen. Then, it iteratively selects a vector that has the largest distance to its own hub as a new hub, and re-clusters all the vectors based on their distances to all the selected hubs. This process continues until there is no vector whose distance to its hub is larger than the half of the average hub-hub distance.

We choose this simple iterative algorithm because it runs much faster, and works equally well as the Single Linkage Hierarchical algorithm in our experiments. The reason that even the simple clustering algorithm works well is that the ratio of inter-cluster distance to intra-cluster distance significantly increases after feature selection. Since the current clustering algorithm is sensitive to outliers, we plan to explore other clustering algorithms such as K-means.

Once we detect a new cluster and generate an alarm, we examine further to identify the involved hosts and the files from which the cluster resulted. The suspicious hosts are just the ones whose file updates correspond to the feature vectors in the new cluster. To determine which files possibly cause the alarm, we only focus on the files picked by the wavelet-based selection to define the feature vector space. For each of those files, if it is updated by all the hosts in the new cluster during the detection window, but has not been updated by any host during the corresponding comparison window, Seurat outputs this file as a candidate file. Similarly, Seurat also reports the set of files that have been updated during the comparison window, but are not updated by any host in the new cluster during the detection window.

Based on the suspicious hosts and the selected files for explaining root causes, system administrators can decide whether the updates are known administrative updates that should be suppressed, or some abnormal events that should be further investigated. If the updates are caused by malicious attacks, administrators can take remedial counter measures for the new cluster. Furthermore, additional compromised hosts can be identified by checking if the new cluster expands later and if other hosts have updated the same set of candidate files.

4   Experiments

We have developed a multi-platform (Linux and Windows) prototype of Seurat that consists of a lightweight data collection tool and a correlation module. The data collection tool scans the file system of the host where it is running and generates a daily summary of file update attributes. Seurat harvests the summary reports from multiple hosts in a network system and the correlation module uses the reports for anomaly detection.

We have installed the Seurat data collection tool on a number of campus office machines and a teaching cluster that are used by students daily. By default, the tool

scans the attributes of all system files on a host. For privacy reasons, personal files under user home directories are not scanned. The attributes of a file include the file name, type, device number, permissions, size, inode number, important timestamps, and a 16-byte MD5 checksum of file content. The current system uses only a binary bit to represent each file update, but the next version may exploit other attributes reported by the data collection tool. Each day, each host compares the newly scanned disk snapshot against that from the previous day and generates a file update summary report. In the current prototype, all the reports are uploaded daily to a centralized server where system administrators can monitor and correlate the file updates using the correlation module.

In this section, we study the effectiveness of Seurat's pointillist approach for detecting aggregated anomalous events. We use the daily file update reports from our real deployment to study the false positive rate and the corresponding causes in Section 4.1. We evaluate the false negative rate with simulated attacks in Section 4.2. In order to verify the effectiveness of our approach on real malicious attacks, we launched a real Linux worm into an isolated cluster and report the results in Section 4.3.

4.1   False Positives

The best way to study the effectiveness of our approach is to test it with real data. We have deployed Seurat on a teaching cluster of 22 hosts and have been collecting the daily file update reports since Nov 2003. The teaching cluster is mostly used by students for their programming assignments. They are also occasionally used by a few graduate students for running network experiments.

For this experiment, we use the file update reports from Dec 1, 2003 until Feb 29, 2004 to evaluate the false positive rate. During this period, there are a few days when a couple of hosts failed to generate or upload reports due to system failure or reconfigurations. For those small number of missing reports, we simply ignore them because they do not affect the aggregated file update patterns.

We set the correlation window to 32 days in order to accommodate monthly file update patterns. That is, we correlate the update pattern from day 1 to day 32 to identify abnormal events on day 32, and correlate the update pattern from day 2 to day 33 to detect anomalies on day 33, etc. Thus, our detection starts from Jan 1, 2004, since we do not have 32-day correlation windows for the days in Dec 2003.

Dimension Reduction. Once we fixed the correlation window of a particular day, we identify relevant files using wavelet-based selection with a constant threshold $\alpha = 2$ to define the feature vector space for simplicity. We then perform PCA to reduce the data dimensionality further by picking the first several principal components that account for 98% of the input data variances.

Throughout the entire period of 91 days, 772 files with unique file names were updated by at least two different hosts. Figure 7 (a) shows the number of hosts that updated each file during the data collection period. We observe that only a small number files (e.g.,`/var/adm/syslog/mail.log`) are updated regularly by all of the hosts, while most other files (e.g., `/var/run/named.pid`) are updated irregularly, depending on the system usage or the applications running.
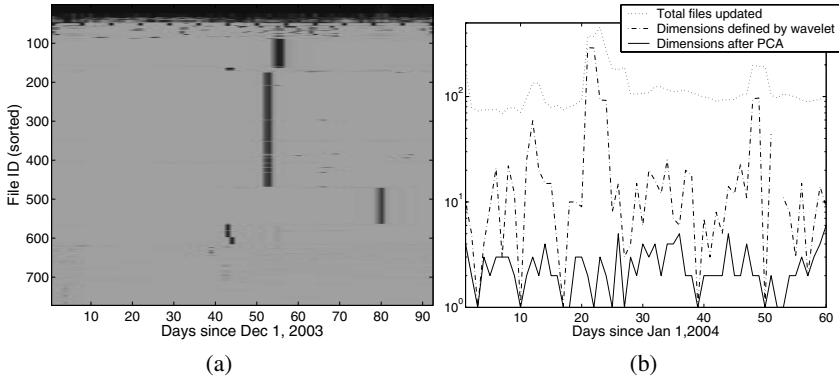
Fig. 7. Feature selection and dimension reduction: (a) File update patterns. Files are sorted by the cumulative number of hosts that have updated them throughout the 91 days. The darker the color is, the more hosts updated the corresponding file. (b) The number of feature vector dimensions after wavelet-based selection and PCA consecutively.

Figure 7 (b) shows the results of feature selection. There were, on average, 140 files updated by at least two different hosts during each correlation window. After wavelet-based selection, the average number of feature dimensions is 17. PCA further reduces the vector space dimension to below 10.

False Alarms. After dimension reduction, we perform clustering of feature vectors and identify new clusters for each day. Figure 8 illustrates the clustering results of 6 consecutive days from Jan 19, 2004 to Jan 24, 2004. There are two new clusters identified on Jan 21 and Jan 23, which involve 9 hosts and 6 hosts, respectively. Since Seurat outputs a list of suspicious files as the cause of each alarm, system administrators can tell if the new clusters are caused by malicious intrusions.

Based on the list of files output by Seurat, we can figure out that the new clusters on Jan 21 and Jan 23 reflect large scale file updates due to a system reconfiguration at the beginning of the spring semester. For both days, Seurat accurately pinpoints the exact hosts that are involved. The reconfiguration started from Jan 21, when a large number of binaries, header files, and library files were modified on 9 out of the 22 hosts. Since the events are known to system administrators, we treat the identified vectors as normal for future anomaly detection. Thus, no alarm is triggered on Jan 22, when the same set of library files were modified on 12 other hosts. On Jan 23, the reconfiguration continued to remove a set of printer files on 6 out of the 22 hosts. Again, administrators can mark this event as normal and we spot no new cluster on Jan 24, when 14 other hosts underwent the same set of file updates.

In total, Seurat raises alarms on 9 out of the 60 days under detection, among which 6 were due to system reconfigurations. Since the system administrators are aware of such events in advance, they can simply suppress these alarms. The 3 other alarms are generated on 3 consecutive days when a graduate student performed a network experiment that involved simultaneous file updates at multiple hosts. Such events are normal but rare, and should alert the system administrators.
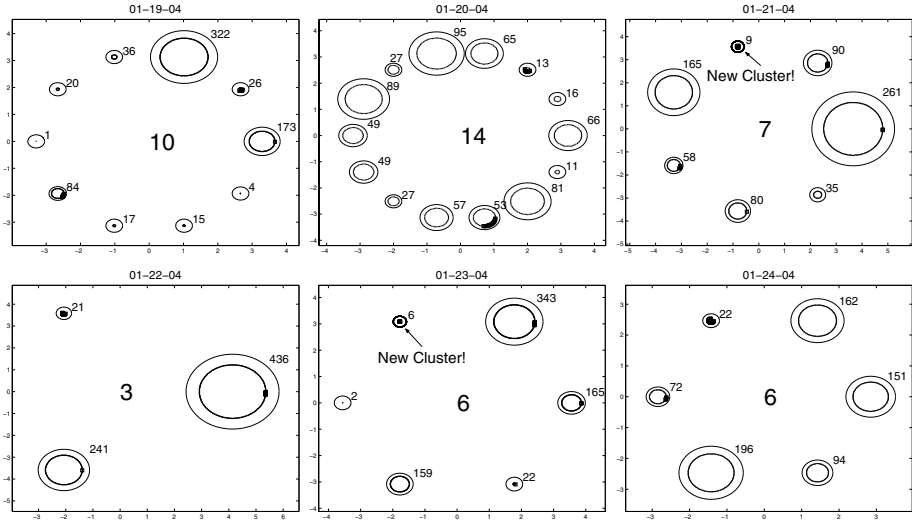
Fig. 8. Clustering feature vectors for anomaly detection: Each circle represents a cluster. The number at the center of the figure shows the total number of clusters. The radius of a circle corresponds to the number of points in the cluster, which is also indicated beside the circle. The squared dots correspond to the new points generated on the day under detection. New clusters are identified by a thicker circle.

4.2   False Negatives

The primary goal of this experiment is to study the false negative rate and detection latency of Seurat as the stealthiness of the attack changes. We use simulated attacks by manually updating files on the selected host reports, as if they were infected.

We first examine the detection rate of Seurat by varying the degree of attack aggressiveness. We model the attack propagation speed as the number of hosts infected on each day (the detection window), and model the attack stealthiness on a local host as the number of new files installed by this attack. Our simulation runs on the same teaching cluster that we described in Section 4.1. Since the aggregated file update patterns are different for each day, we randomly pick ten days in Feb 2004, when there was no intrusion. On each selected day, we simulate attacks by manually inserting artificial new files into a number of host reports on only that day, and use the modified reports as input for detection algorithm. We then remove those modified entries, and repeat the experiments with another day. The detection rate is calculated as the number of days that Seurat spots new clusters over the total ten days.

Figure 9 shows the detection rate of Seurat by varying the number of files inserted on each host and the number of hosts infected. On one hand, the detection rate monotonically increases as we increase the number of files inserted on each host by an attack. Since the inserted files do not exist before, each of them will be selected as a feature dimension by the wavelet-based selection, leading to larger distances between the points of infected host state changes and the points of normal host state changes. Therefore,
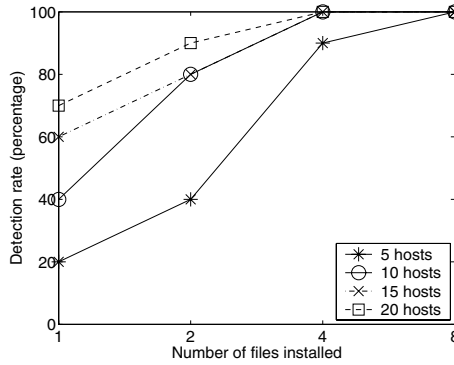
Fig. 9. Detection rate: We vary the number of hosts infected and the number of files inserted on each host by the simulated attacks.

the more new files are injected by an attack, the higher the detection rate gets. On the other hand, as we increase the number of infected hosts, the number of points for abnormal host state changes becomes large enough to create an independent new cluster. Thus, rapidly propagating attacks are more likely to be caught. Accordingly, detecting a slowly propagating attack requires a larger detection window, hence longer detection latency, in order to accumulate enough infected hosts. We revisit this issue in Section 5.

We further evaluate the detection rate of Seurat on six Linux worms with simulated attacks. To do so, we compile a subset of files modified by each worm based on the descriptions from public Web sites such as Symantec [16] and F-Secure information center [17]. We then manually modify the described files in a number of selected host reports to simulate the corresponding worm attacks. Again, for each worm, we vary the number of infected hosts, and run our experiments on the teaching cluster with ten randomly selected days.

Worms	Adore	Ramen-A	Ramen-B	Slapper-A	Slapper-B	Kork
Files modified	10	8	12	3	4	5
2 infected hosts	80%	80%	90%	30%	40%	30%
4 infected hosts	100%	100%	90%	70%	80%	70%
8 infected hosts	100%	100%	100%	100%	100%	100%

Fig. 10. Detection rate of emulated worms: We vary the number of hosts compromised by the attacks.

Table 10 shows the number of files modified by each worm and the detection rate of Seurat. In general, the more files modified by a worm, the more likely the worm will be detected. But the position of a file in the file system directory tree also matters. For example, both Slapper-B worm and Kork worm insert 4 new files into a compromised host. However, Kork worm additionally modifies /etc/passwd to create accounts with root privileges. Because there are many hosts that have updated /etc/passwd

during a series of system reconfiguration events, the inclusion of such files in the feature vector space reduces the distances from abnormal points to normal points, resulting in higher false negative rates. We discuss this further in Section 5.

4.3   Real Attacks

Now we proceed to examine the efficacy of Seurat during a real worm outbreak. The best way to show this would be to have Seurat detect an anomaly caused by a new worm propagation. Instead of waiting for a new worm's outbreak, we have set up an isolated computer cluster where, without damaging the real network, we can launch worms and record file system changes. This way, we have full control over the number of hosts infected, and can repeat the experiments. Because the isolated cluster has no real users, we merge the data acquired from the isolated cluster with the data we have collected from the teaching cluster in order to conduct experiments.

We obtained the binaries and source codes of a few popular worms from public Web sites such as whitehats [18] and packetstorm [19]. Extensively testing Seurat, with various real worms in the isolated cluster, requires tremendous effort in setting up each host with the right versions of vulnerable software. As a first step, we show the result with the Lion worm [20] in this experiment.

The Lion worm was found in early 2001. Lion exploits a vulnerability of BIND 8.2, 8.2-P1, 8.2.1, 8.2.2-Px. Once Lion infects a system, it sets up backdoors, leaks out confidential information (`/etc/passwd`, `/etc/shadow`) via email, and scans the Internet to recruit vulnerable systems. Lion scans the network by randomly picking the first 16 bits of an IP address, and then sequentially probing all the 2^{16} IP addresses in the space of the block. After that, Lion randomly selects another such address block to continue scanning. As a result, once a host is infected by Lion, all the vulnerable hosts nearby (in the same IP address block) will be infected soon. Lion affects file systems: the worm puts related binaries and shell scripts under the `/dev/.lib` directory, copies itself into the `/tmp` directory, changes system files under the `/etc` directory, and tries to wipe out some log files.

We configured the isolated cluster with three Lion-vulnerable hosts and one additional machine that launched the worm. The vulnerable machines were running RedHat 6.2 including the vulnerable BIND 8.2.2-P5. The cluster used one C class network address block. Every machine in the cluster was connected to a 100Mbps Ethernet and was running `named`. The Seurat data collection tool generated a file system update report on every machine daily.

After we launched the Lion worm, all three vulnerable hosts in the isolated cluster were infected quickly one after another. We merge the file update report by the each compromised host with a different normal host report generated on Feb 11, 2004, when we know there was no anomaly. Figure 11 shows the clustering results of three consecutive days from Feb 10, 2004 to Feb 12, 2004 using the merged reports.

On the attack day, there are 64 files picked by the wavelet-based selection. The number of feature dimensions is reduced to 9 after PCA. Seurat successfully detects a new cluster consisting of the 3 infected hosts. Figure 12 lists the 22 files selected by Seurat as the causes of the alarm. These files provide enough hints to the administrators to confirm the existence of the Lion worm. Once detected, these compromised hosts as
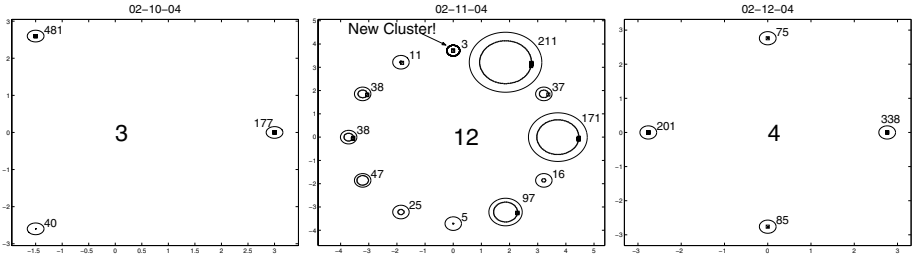
Fig. 11. Intrusion detection by Seurat: Seurat identified a new cluster of three hosts on Feb 11, 2004, when we manually launched the Lion worm.

File ID.	File name	File ID.	File name
1	/sbin/asp	12	/var/spool/mail
2	/dev/.lib	13	/dev/.lib/bindx.sh
3	/dev/.lib/star.sh	14	/tmp/ramen.tgz
4	/var/spool/mail/root	15	/dev/.lib/scan.sh
5	/dev/.lib/bind	16	/dev/.lib/pscan
6	/etc/hosts.deny	17	/var/spool/mqueue
7	/dev/.lib/randb	18	/dev/.lib/hack.sh
8	/sbin	19	/dev/.lib/.hack
9	/var/log	20	/dev/.lib/index.html
10	/dev/.lib/bindname.log	21	/dev/.lib/asp62
11	/dev/.lib/index.htm	22	/var/log/sendmail.st

Fig. 12. Suspicious files for the new cluster on Feb 11, 2004.

well as the list of suspicious files can be marked for future detection. If, in the following days, there are more hosts that are clustered together with the already infected machines, or experience the same file updates, then we may conclude they are infected by the same attack.

5   Discussion

5.1   Vulnerabilities and Limitations

By identifying parallel occurrences of coincident events, Seurat will be most successful in detecting virus or worm propagations that result in file modifications at multiple hosts. Certain attacks (e.g., password guessing attacks) that succeed only once or a few times in a network system may evade Seurat detection. The current prototype of Seurat also has limited detection capability to the following types of attacks.

Stealthy attack. Attackers may try to evade detection by slowing attack propagation. If an attacker is patient enough to infect only one host a day in the monitored network system, Seurat will not notice the intrusion with the current one-day detection window because Seurat focuses only on anomalous file changes common across multiple hosts.

A larger detection window such as a couple of days or a week can help to catch slow, stealthy attacks. Note, however, that Seurat notices the attacks only after multiple hosts in the network system are compromised. In other words, if an attack propagates slowly, Seurat may not recognize the attack for the first few days after the initial successful compromise. There is thus a tradeoff between detection rate and detection latency.

Mimicry attack. An attacker can carefully design his attack to cause file updates that look similar to regular file changes, and mount a successful *mimicry attack* [21]. There are two ways to achieve a mimicry attack against the current prototype. First, an attacker may try to fool Seurat's feature selection process by camouflaging all intrusion files as frequently, regularly updated files. Those concealed files, even when they are modified in an unexpected way (e.g., entries removed from append-only log files), will not be selected as feature vector dimensions because of current use of the binary feature representation. Note that Seurat's data collection tool provides additional information on file system changes, such as file size, file content digest, and permissions. By incorporating the extra information in representing host state transition, Seurat can make such mimicry attacks harder. Second, an attacker may find a way to cloak abnormal file updates with many normal but irregular changes during Seurat's clustering process. For example, in Section 4.2, we observed that the false negative rate of detecting the Kork worm was relatively higher due to the interference of irregular system reconfiguration. We leave it as future work to quantify this type of mimicry attack and the effectiveness of possible counter measures.

Random-file-access attack. Seurat correlates file updates based on their complete path names. Thus attackers can try to evade Seurat by installing attack files under different directories at different hosts, or replacing randomly chosen existing files with attack files. Many recent email viruses already change the virus file names when they propagate to a new host; we envision similar techniques could be employed by other types of attacks soon. Note, however, that even the random-file-access attack may need a few anchor files at fixed places, where Seurat still has the opportunity to detect such attacks. A more robust representation of a file, for example, an MD5 checksum, could help Seurat detect random-file-access attacks.

Memory-resident attack. Memory-resident and BIOS-resident only attacks make no file system updates. Thus Seurat will not be able to detect memory resident attacks by examining host file updates, nor those attacks that erase disk evidence before the Seurat data collection tool performs the scheduled disk scan.

Kernel/Seurat modification attack. The effectiveness of Seurat relies on the correctness of reports from the data collecting tools running on distributed hosts. So the host kernels and the Seurat data collection tools should run on machines protected by trusted computing platforms [22]. An alternative solution is to monitor file system changes in real time (will be discussed further in Section 5.2) and to protect file update logs using secure audit logging [23].

5.2   Future Work

Real-time anomaly detection. The current prototype periodically scans and reports file system updates with a 1-day cycle, which may be slow to detect fast propagating at-

tacks. To shorten detection latency, we are enhancing the Seurat data collection module to monitor system calls related with file updates, and report the changes immediately to the correlation module. The reported file updates will be instantly reflected by setting the corresponding bits in the feature vectors at the Seurat correlation module, which continuously performs clustering of the new feature vectors for real time anomaly detection.

Distributed correlation module. Currently, Seurat moves the daily reports from distributed data collection tools to a centralized server, where the correlation module computes and clusters the host vectors. Despite the simplicity of centralized deployment, the centralized approach exposes Seurat to problems in scalability and reliability. First, the amount of report data to be transferred to and stored at the centralized server is large. In our experience, a host generates a file update report of 3K-140KBytes daily in a compressed format, so the aggregate report size from hundreds or thousands of hosts with a long comparison window will be large. The report size will be larger when Seurat's data collection tool reports the host state changes in real time. Second, the monitored hosts could be in different administrative domains (i.e., hosts managed by different academic departments or labs) and it is often impractical to transfer the detailed reports from all the hosts to one centralized server due to privacy and confidentiality issues. Third, the centralized server can be a single point-of-failure. It is important for Seurat to work even when one correlation server is broken or a part of network is partitioned. A distributed correlation module will cope with those issues. We are now investigating methods to correlate file update events in a distributed architecture such as EMERALD [1], AAFID [24], and Mingle [25].

Other applications. The approach of clustering coincident host state changes can be generalized to other types of applications such as detecting the propagation of spyware, illegal file sharing events, or erroneous software configuration changes. We are currently deploying Seurat on Planetlab [26] hosts for detecting software configuration errors by identifying host state vectors that do not fall into an expected cluster.

6   Related Work

Seurat uses file system updates to represent a host state change. File system updates have been known to be useful information for intrusion detection. Tripwire [4], AIDE [8], Samhain [27] are well-known intrusion detection systems that use file system updates to find intrusions. Recently proposed systems such as the storage-based intrusion detection systems [7] and some commercial tools [28] support real-time integrity checking. All of them rely on a pre-defined rule set to detect anomalous integrity violation, while Seurat diagnoses the anomaly using learning and correlation across time and space.

Leveraging the information gathered from distributed multiple measurement points is not a new approach. Many researchers have noticed the potential of the collective approaches for intrusion detection or anomaly detection. Graph-based Intrusion Detection System (GrIDS) [29] detects intrusions by building a graph representation of network activity based on the report from all the hosts in a network. Different from Seurat, GrIDS uses the TCP/IP network activity between hosts in the network to infer

patterns of intrusive or hostile activities based on pre-defined rules. Other systems, such as Cooperative Security Managers (CSM) [30], Distributed Intrusion Detection System (DIDS) [31], also take advantage of a collective approach to intrusion detection. They orchestrate multiple monitors watching multiple network links and track user activity across multiple machines.

EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) [1] and Autonomous Agents For Intrusion Detection (AAFID) [24] have independently proposed distributed architectures for intrusion detection and response capability. Both of them use local monitors or agents to collect interesting events and anomaly reports (from a variety of sources; audit data, network packet traces, SNMP traffic, application logs, etc.). The architectures provide the communication methods to exchange the locally detected information and an easy way to manage components of the systems. AAFID performs statistical profile-based anomaly detection and EMERALD supports a signature-based misuse analysis in addition to the profile-based anomaly detection. Note that Seurat starts with similar motivation. But Seurat focuses more on the technique for correlating the collective reports for anomaly detection, and infers interesting information on the system state from learning, rather than relying on a pre-defined set of events or rules. We envision Seurat as a complementary technique, not as a replacement of the existing architectures that provide global observation sharing.

Correlating different types of audit logs and measurement reports is another active area in security research. Many researchers have proposed to correlate multiple heterogeneous sensors to improve the accuracy of alarms [3, 32, 33, 2, 34]. In this work, we attempt to correlate information gathered by homogeneous monitors (especially, the file system change monitors) but we may enhance our work to include different type of measurement data to represent individual host status.

Wang et al. [35] also have noticed the value of spatial correlation of multiple system configurations and applied a collective approach to tackle misconfiguration trouble shooting problems. In their system, a malfunctioning machine can diagnose its problem by collecting system configuration information from other similar and friendly hosts connected via a peer-to-peer network. The work does not target automatic detection of the anomaly, but rather it aims at figuring out the cause of a detected problem.

7   Conclusions

In this paper, we presented a new "pointillist" approach for detecting aggregated anomalous events by correlating information about host file updates both space and time. Our approach explores the temporal and spatial locality of system state changes through learning and correlation. It requires neither prior knowledge about normal host activities, nor system specific rules.

A prototype implementation, called *Seurat*, suggests that the approach is effective in detecting rapidly propagating attacks that modify host file systems. The detection rate degrades as the stealthiness of attacks increases. By trading off detection latency, we are also able to identify hosts that are compromised by slowly propagating attacks. For each alarm, Seurat identifies suspicious files and hosts for further investigation, greatly facilitating root cause diagnosis and false alarm suppression.

References

1. Porras, P.A., Neumann, P.G.: EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In: Proceedings of the 20th National Information Systems Security Conference. (1997)
2. Abad, C., Taylor, J., Sengul, C., Zhou, Y., Yurcik, W., Rowe, K.: Log Correlation for Intrusion Detection: A Proof of Concept. In: Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA (2003)
3. Kruegel, C., Toth, T., Kerer, C.: Decentralized Event Correlation for Intrusion Detection. In: International Conference on Information Security and Cryptology (ICISC). (2001)
4. Tripwire, Inc.: Tripwire. (`http://www.tripwire.com`)
5. CERT Coordination Center: Overview of Attack Trends. `http://www.cert.org/archive/pdf/attack_trends.pdf` (2002)
6. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the Slammer Worm. IEEE Security and Privacy **1** (2003) 33–39
7. Pennington, A., Strunk, J., Griffin, J., Soules, C., Goodson, G., Ganger, G.: Storage-based intrusion detection: Watching storage activity for suspicious behavior. In: Proceedings of 12th USENIX Security Symposium, Washington, DC (2003)
8. Lehti, R., Virolainen, P.: AIDE - Advanced Intrusion Detection Environment. (`http://www.cs.tut.fi/~rammer/aide.html`)
9. Berry, M.W., Drmac, Z., Jessup, E.R.: Matrices, vector spaces, and information retrieval. SIAM Review **41** (1999)
10. Kamber, M.: Data mining: Concepts and techniques. Morgan Kaufmann Publishers (2000)
11. Zhang, J., Tsui, F., Wagner, M.M., Hogan, W.R.: Detection of Outbreaks from Time Series Data Using Wavelet Transform. In: AMIA Fall Symp., Omni Press CD (2003) 748–752
12. Jolliffe, I.T.: Principle component analysis. Spring-Verlag, New York (1986)
13. Forgy, E.: Cluster analysis of multivariante data: Efficiency vs. Interpretability of classifications. Biometrics **21** (1965)
14. Gersho, A., Gray, R.: Vector Quantization and Signal Compresssion. Kluwer Academic Publishers (1992)
15. Moore, A.: K-means and Hierarchical Clustering. `http://www.cs.cmu.edu/~awm/tutorials/kmeans09.pdf` (available upon request) (2001)
16. Symantec: Symantec Security Response. (`http://securityresponse.symantec.com`)
17. F-Secure: F-Secure Security Information Center. (`http://www.f-secure.com/virus-info`)
18. Whitehats, Inc.: Whitehats Network Security Resource. (`http://www.whitehats.com`)
19. PacketStorm: Packet Storm. (`http://www.packetstormsecurity.org`)
20. SANS Institute: Lion Worm. `http://www.sans.org/y2k/lion.htm` (2001)
21. Wagner, D., Dean, D.: Mimicry Attacks on Host-Based Intrusion Detection Systems. In: Proceedings of ACM Conference on Computer and Communications Security (CCS). (2002)
22. Trusted Computing Platform Alliance: Trusted Computing Platform Alliance. (`http://www.trustedcomputing.org`)
23. Schneier, B., Kelsey, J.: Cryptographic Support for Secure Logs on Untrusted Machines. In: The Seventh USENIX Security Symposium. (1998)
24. Balasubramaniyan, J.S., Garcia-Fernandez, J.O., Isacoff, D., Spafford, E., Zamboni, D.: An architecture for intrusion detection using autonomous agents. In: Proceedings of the 14th IEEE Computer Security Applications Conference. (1998)

25. Xie, Y., O'Hallaron, D.R., Reiter, M.K.: A Secure Distributed Search System. In: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing. (2002)
26. Planetlab: PlanetLab. (http://www.planet-lab.org)
27. Samhain Labs: Samhain. (http://la-samhna.de/samhain)
28. Pedestal Software: INTACT^{TM}.
    (http://www.pedestalsoftware.com/products/intact)
29. Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Rowe, J., Staniford-Chen, S., Yip, R., Zerkle, D.: The Design of GrIDS: A Graph-Based Intrusion Detection System. Technical Report CSE-99-2, U.C. Davis Computer Science Department (1999)
30. White, G., Fisch, E., Pooch, U.: Cooperating security managers: A peer-based intrusion detection system. IEEE Network **10** (1994)
31. Snapp, S.R., Smaha, S.E., Teal, D.M., Grance, T.: The DIDS (distributed intrusion detection system) prototype. In: the Summer USENIX Conference, San Antonio, Texas, USENIX Association (1992) 227–233
32. Valdes, A., Skinner, K.: Probabilistic Alert Correlation. In: Recent Advances in Intrusion Detection, Volume 2212 of Lecture Notes in Computer Science, Springer-Verlag (2001)
33. Andersson, D., Fong, M., Valdes, A.: Heterogeneous Sensor Correlation: A Case Study of Live Traffic Analysis. Presented at IEEE Information Assurance Workshop (2002)
34. Ning, P., Cui, Y., Reeves, D.S.: Analyzing Intensive Intrusion Alerts Via Correlation. In: Recent Advances in Intrusion Detection, Volume 2516 of Lecture Notes in Computer Science, Springer-Verlag (2002)
35. Wang, H., Hu, Y., Yuan, C., Zhang, Z.: Friends Troubleshooting Network: Towards Privacy-Preserving, Automatic Troubleshooting. In: Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS). (2004)