Know Your Achilles' Heel: Automatic Detection of Network Critical Services

Ali Zand
UC Santa Barbara
ali.zand@gmail.com

Amir Houmansadr
UMass Amherst
amir@cs.umass.edu

Giovanni Vigna
UC Santa Barbara
vigna@cs.ucsb.edu

Richard Kemmerer
UC Santa Barbara
kemm@cs.ucsb.edu

Christopher Kruegel
UC Santa Barbara
chris@cs.ucsb.edu

ABSTRACT

Administrators need effective tools to quickly and automatically obtain a succinct, yet informative, overview of the status of their networks to make critical administrative decisions in a timely and effective manner. While the existing tools might help in pointing out machines that are heavily used or services that are failing, more subtle relationships, such as indirect dependencies between services, are not made apparent. In this paper, we propose novel techniques to automatically provide insights into the state of a network and the importance of the network components. We developed a tool, called PARIS, which receives traffic information from various off-the-shelf network monitoring devices. PARIS computes an importance metric for the network's components based on which the administrators can prioritize their defensive and prohibitive actions. We evaluated PARIS by running it on a mid-size, real-world network. The results show that PARIS is able to automatically provide situation awareness in a timely, effective manner.

1. INTRODUCTION

Today's computer networks have turned into complex infrastructures providing complex inter-dependent services, which are often transparent or hidden. This complexity is expected to further increase as engineers tend to build larger and more complex services by combining the smaller, simpler ones. Even a typical end-user service may be composed of multiple underlying services with complex dependencies. Computer networks are also under ever-increasing attacks [39]. The ever-increasing complexity of distributed systems, combined with the constant increase in the volume and sophistication of attacks to computer networks, create serious challenges for network administrators in taking timely, appropriate actions.

For example, a webmail service usually consists of a web server (to provide the web-based front-end), an email server (to send and receive emails), a file server (to store the emails), a Kerberos server (to authenticate users), and a DNS server (to resolve host addresses). Failure of each component of a complex composite service (a service built up from simpler services) can cause a failure in the composite service itself.

Therefore, network administrators need tools to monitor the state of their assets to detect the occurrence of malicious activities and to foresee the required corrective actions. Traditional network monitoring tools provide an overwhelming amount of fine-grained details about the occurrences of malicious events and activities [41]. However, they do not provide a succinct, high-level understanding of the current and future states of the network.

In this paper, we study the design of tools that provide *situation awareness* for network administrators. In the context of computer networks, situation awareness aims to provide a decision maker (e.g., a network administrator) with a high-level overview and understanding of her computer network. More precisely, situation awareness can be defined as *the perception of the network elements, the understanding of their meaning and importance, and the projection of their status into the near future* [13].

Network situation awareness aims to help a network administrator identify important network *assets*, analyze their dependencies, and understand the importance of different assets in carrying out the underlying organization's *mission*s. A "network asset" broadly describes both the hardware elements (e.g., servers and routers) and the software elements (e.g., applications and services) that constitute a computer network. In the context of this work we only consider network services as assets, since the other types of assets are not affected by our analysis. A network mission is a collection of tasks (services) that are carried out by different network components of an organization to achieve a specific goal.

The term "mission" reveals the military roots of the idea of situation awareness. However, the concept of a mission is general and equally applies to civilian networks. For example, providing web mail services to students can be one of the missions of a university network. In this case, one of the tasks might be to send/receive emails, while a second task is to provide a web interface to users to access their email accounts. From working with administrators of organizational networks, we realize that identifying network missions and ensuring their continuous operation is significantly important. However, doing such through manual investigation by human administrators is in many ways impractical

ACSAC '15, December 07-11, 2015, Los Angeles, CA, USA
 ISBN 978-1-4503-3682-6/15/12...$15.00
DOI: http://dx.doi.org/10.1145/2818000.2818012

and inefficient due to the large size of today's networks, the error-proneness of human operators, and the lack of documentation for various network components.

In this paper, we devise novel techniques and tools for *automated* situation awareness. Our tool uses data collected from various network devices to 1) identify network missions, and, 2) continuously monitor their correct operation; to the best of our knowledge, we are the first to design tools for automated detection and monitoring of network missions. Unlike previous work [22, 18] that require higher level information such as expert knowledge, up-to-date documentation, and human operator interaction to operate, our approach only uses the reports made by network monitoring devices. Also, our approach does not require any access to the network hosts and does not generate extra traffic. Our tool, called PARIS, provides three main capabilities by using a variety of techniques including statistical analysis and clustering tools. First, PARIS processes low-level network traffic data, i.e., NetFlow records, in order to identify and characterize network services.

Second, PARIS determines relationships between the identified services. In particular, it finds service *dependencies* and service *redundancies* (i.e., services implementing the same functionality). The knowledge about these relationships is crucial to properly determine the importance of different services. Consider, for example, a service that is directly responsible for the success of a particular mission. Quite likely, the system administrators know that this service is important. However, this service might also depend on two other services. These services are not directly involved in the mission, but they are equally important for its success. It is important to accurately identify these relationships, which often are not obvious.

The third capability of PARIS is to *rank* the discovered network services. This ranking is based on the importance of the services towards achieving the organization's missions. PARIS automatically infers importance scores for services based on their activity as well as their relations with each other.

We implemented PARIS and evaluated its performance on the network of a large organization,[1] which contained NetFlow records for 1.6 billion network connections produced by more than 593 hosts (distinct IP addresses). Our tool identified a variety of interesting services and missions, as well as their relationships, whose correctness and comprehensiveness were validated by the organization's network administrators.

The rest of this paper is organized as follows. In Sections 2 to 4, we describe how PARIS extracts network services, infers their relationships, and ranks them based on their importance, respectively. We present the implementation results of PARIS in Section 5 and discuss different issues in Section 6. The related work is discussed in Section 7, and we conclude the paper in Section 8.

2. EXTRACTING NETWORK SERVICES

In the first step, PARIS extracts a list of network services by analyzing NetFlow [1] records. These NetFlow records are collected from various network monitoring devices, such as routers and switches. Each NetFlow record contains high-level information about an observed network connection, including the connection start time, end time, source/destination IP addresses and port numbers, and the number of exchanged packets and bytes.

PARIS considers a distinct triple `(PROTO, IP, PORT)` as a network service if it is frequently seen in the dataset. Our service extraction is similar to Orion [10].

Service profiles: PARIS generates a service profile for each of the extracted network services, capturing its observed traffic activity during the analysis period T_E. The purpose of service profiles is twofold: First, PARIS uses service profiles as a basis to determine correlated activities between different services. This is needed to determine interesting relationships between services and to detect missions. Second, service profiles are used to recognize similar services, i.e., services that likely implement the same application. This is important for the identification of backup services and meta-missions.

In particular, the service profile includes the number of bytes sent/received by the service, the number of packets sent/received by the service, number of clients, and the number of requests handled by the service. This information is provided as a time series by dividing the time dimension into non-overlapping *evaluation slot*s of length Δ and evaluating these features for each of the slots.

3. INFERRING RELATIONSHIPS

The next task of the analysis is to find relationships among the detected services. In particular, we are interested in services that operate together to implement higher-level functionality, as well as in redundant services.

3.1 Detecting Correlated Services

Correlated services are services that exhibit synchronized activity patterns, being active or inactive at roughly the same times. When we consistently observe correlated traffic patterns in the data, we assume that the corresponding services operate together. Typically, this is because both (or multiple) services are needed in order to achieve a specific goal. In our web mail example, whenever there is a spike in the number of clients that send email, we see corresponding increases in the activity of the web server, the mail server, and the authorization service. Of course, this does not mean that the mail server or the web server will be idle when no web mail is sent. However, overall, we expect noticeable increases in web mail activity to result in an increase of web and mail service activity.

Correlation graphs. To detect correlated activity between services, we use time series analysis. We first divide the connections associated with each service into short, Δ-length time slots. Δ should be small enough so that bursts in activity stand out and are not "smoothed out" over too long of a period. On the other hand, Δ should be long enough to allow for sufficient tasks (or individual activities) to complete. Otherwise, it is not possible to observe and distinguish increased activity. In our implementation of PARIS, we use a Δ value of five minutes.

For each time slot, we compute three discrete time series: R_{S_i}, P_{S_i}, and B_{S_i}, which correspond to the number of requests for a service S_i, the number of packets sent by S_i, and the number of bytes sent by S_i, respectively. To capture service activity, we focus on what the service sends

[1] The name of the organization is anonymized for peer-review purposes.

back to the client. We ignore the bytes and packets that the service receives, since this does not show the activity of the service itself (e.g., there may be no response to the received packets/bytes).

The time series for each service, over the entire analysis period T_E, could be directly used to compute the correlation between services. However, this would only identify services that always operate together. It would miss services that operate together periodically, e.g., for one task per day (or week). It would also miss relationships where service A and service B operated together during some part of the evaluation interval before B was switched with service C. Therefore, examining the time series over the entire period might not be desirable. Instead, we divide the analysis period T_E into smaller intervals T_I. We use an interval length of one hour for T_I. This time is short enough so that short-term, correlated activities are properly captured. Moreover, it is long enough so that the time series contain sufficient data points ($\frac{T_I}{\Delta} = 12$) to obtain correlation coefficients with enough confidence. Another desirable property of T_I is that it is suitable to detect periodic services, because periodic services are usually set to run on intervals that divide or are dividable by an hour, which makes it more manageable for the administrators.

To select the appropriate value for Δ, we experimented with values of 1, 5, and 10 minutes. We then looked at the average and standard deviation values of the correlations. The changes in these values were negligible (the difference of averages was 0.0119, and the difference of standard deviations was 0.027). The resulting correlation matrix was robust and not sensitive to the changes in Δ. We chose 5 minutes for Δ because we experimentally determined that the sensor device timers can be skewed by at most one minute.

We define the correlation of two network services S_i and S_j, using their corresponding time series, as:

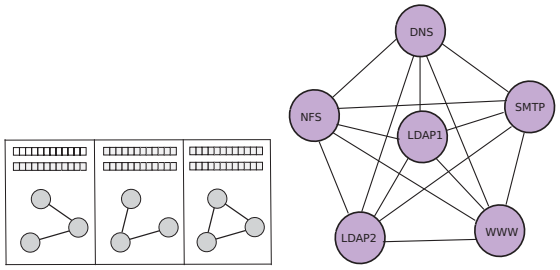
$$C(S_i, S_j) = \max\{C^P(R_{S_i}, R_{S_j}), C^P(P_{S_i}, P_{S_j}), C^P(B_{S_i}, B_{S_j})\} \quad (1)$$

where $C^P(\cdot,\cdot)$ is the Pearson product-moment correlation given by:

$$C^P(A, B) = \frac{\sum_{i=1}^{n}(A_i - E(A))(B_i - E(B))}{\sqrt{\sum_{i=1}^{n}(A_i - E(A))^2}\sqrt{\sum_{i=1}^{n}(B_i - E(B))^2}} \quad (2)$$

In Equation 2, $E(\cdot)$ evaluates the mean value of the expression, and n is the common length of the sequences A and B, which is equal to $\frac{T_I}{\Delta}$ in our case. PARIS considers two services S_i and S_j to be *correlated* if $C(S_i, S_j) \geq \eta_C$, where η_C is the correlation threshold. The choice of η_C trades off true-correlation and false-correlation of the services; after deriving η_C value corresponding to significance of 0.05 and manually investigating the defined service correlation metric on a large number of correlated and non-correlated pairs of network services, we set η_C to 0.49726. Moreover, we found a bimodal distribution, where correlated services had high values of $C(\cdot)$, while the opposite was true for independent services. In Section 3.3, we derive η_C, based on the significance of correlation coefficients.

Using the computed correlation values, we build the correlation graph, which is a non-directed graph whose vertices are the extracted network services. Two vertices representing the services S_i and S_j are connected with an edge only if they are correlated, i.e., if $C(S_i, S_j) \geq \eta_C$.



(a) time-series analysis (b) sample extracted mission

Figure 1: A sample extracted mission.

3.2 Missions

PARIS finds the *maximal cliques*[2] in the correlation graph and considers them as candidate network missions. Figure 1 shows the process of candidate mission extraction and an example of an extracted mission that represents a part of a web mail mission.

Frequent and infrequent missions. Our system generates one correlation graph and a set of candidate missions for each one-hour period T_I. The candidate missions might suffer from two problems. The first problem is that some candidate missions can be infrequent missions, or they can be the result of coincidental synchronous activity of several services. An infrequent mission can show up as a result of a temporary service that is set up by an administrator and then removed. The second problem is that when a mission (clique of services) occurs in one analysis period it is possible that one of its subsets occurs in some other analysis periods (we often observed this in our experiments). Such a subset mission is redundant, and we want it removed because the superset mission contains all relevant information.

To solve the aforementioned problems, we apply two filtering steps to the candidate missions. First, we discard all infrequent missions appearing less than a threshold (η_M). The rationale behind this is that infrequent missions are coincidentally-correlated activities or temporary missions. We call the parameter η_M the *mission threshold*. The remaining missions are called *frequent missions*.

To address the second problem mentioned above, we check each frequent mission to see whether it is part (a subset) of a larger mission. If this is the case, we compute how many unique appearances each mission has. The number of unique appearances for a mission is the number of T_I periods in which it appears **and** no other supersets of this mission appear. We then discard all missions with unique appearances less than η_M. The reason for this is that some remaining candidate missions may be parts/subsets of other candidate missions, and, therefore, they do not provide any additional information.

3.3 Statistical Analysis for Service Dependency Detection

Given two activity distributions with zero real correlation, $D_1 = N(\mu_1, \sigma_1)$ and $D_2 = N(\mu_2, \sigma_2)$, and two samples of size n, $S^1 = \{s_1^1, s_2^1, \ldots, s_n^1\}$ and $S^1 = \{s_1^2, s_2^2, \ldots, s_n^2\}$, the following distribution (t) follows approximately t-distribution

[2]A maximal clique is a clique on the graph that cannot be extended by including one more adjacent vertex.

with $n-2$ degrees of freedom: $t = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}}$, where r is the Pearson correlation of two samples [11]. We use this property to compute a significance value for a correlation coefficient threshold. The null hypothesis (H_0) is that the two distributions have real correlation of less than or equal to zero. The alternative (H_1) is that the two distributions have positive correlation. Using this formula, it is easily shown that correlation threshold $\eta_C = 0.49726$ when the number of sample points is $n = \frac{T_I}{\Delta} = 12$ corresponds to the correlation coefficient with significance value of $p - value = 0.05$.

Moreover, we use mission threshold η_M to select dependencies that occur at least η_M times. Here we calculate the probability that a random set of three services will have Pearson correlation value higher than or equal η_C for η_M times or more. We define p as the probability of false detection of correlation between three services in a single period: $p = (p - value)^3$. $N = \frac{T_E}{T_I}$ is the number of time-series intervals in the analysis period.
$F(\eta_M; N, p) = \sum_{i=\eta_M}^{N} \binom{N}{i} \times p^i \times (1-p)^{N-i}$ is the probability that a random set of three non-correlated services appears as correlated in η_M or more time-series intervals. This probability for our thresholds, $\eta_M = 24$, $N = 24 \times 30$, and $p = 0.000125$, is $F = 8.02641 \times 10^{-50}$.

3.4 Clustering Network Services

To detect services of the same type, PARIS performs an analysis of the network traffic (flows) for each service. The basic intuition is that similar services will create flows with similar characteristics. That is, services of the same type will have almost the same number of connections of specific types. As an example, type A services have 50% long duration flows with a high number of packets and a small number of bytes and 25% of short lived burst flows (high number of packets with high number of bytes), and 25% of short lived duration flows (low number of packets and high number of bytes). One difficulty of clustering services in this way is assigning the flows to these meaningful (carrying information) classes. To solve this problem, we cluster the flows. To make our approach less sensitive to the clustering parameters, we use fuzzy clustering for flows. This way each flow will have a membership value in each flow class.

PARIS implements a three-step process. First, the system generates models for the traffic (flows) for each service. Next, these models are clustered. Finally, PARIS assigns a type to each service, based on the cluster(s) that its flows belong to. This three-step process is described below.

Modeling network flows. For each network connection, the system extracts the following five features:

- F_1: the number of packets sent by the client,
- F_2: the number of packets sent by the server,
- F_3: the number of bytes sent by the client,
- F_4: the number of bytes sent by the server, and
- F_5: the time duration of the flow.

The five features are represented as a *feature vector* $x \in \mathbb{R}^5$, such that $x = (a_1, ..., a_5)$ and $a_i \in \mathbb{R}$ is the value of feature F_i for that flow. This results in N feature vectors, where N is the number of flows during the analysis period T_E.

Clustering network flows. PARIS classifies network flows based on their feature vectors, by using a *fuzzy c-means* (FCM) algorithm [6]. Fuzzy clustering provides weighted membership for each network connection; i.e., each connection can be a member of multiple clusters with different weights. This makes the clustering less sensitive to the choice of clustering parameters, as compared to the standard *k-means clustering* algorithm [25]. As an example, two flows that belong to the same k-mean cluster might move to different clusters by a slight change in the parameters of the k-means algorithm. For fuzzy clustering, however, slightly changing the algorithm parameters modifies the membership weights only slightly, keeping the similarity of the flows. K-means clustering, on the other hand, provides single membership for each clustered flow, which is not required here since the clustered flows information is not used directly (it is used for clustering the services). Given that N is the total number of flows, FCM runs an optimization algorithm to cluster the N feature vectors into C clusters such that the following *fuzzy c-means cost* function is minimized:

$$J(U, V) = \sum_{j=1}^{C} \sum_{i=1}^{N} (\mu_{i,j})^m \|x_i - v_j\|^2 \quad (3)$$

In Equation 3, $V = \{v_i | i = 1, \ldots, C\}$ is the set of *cluster centers*, and $U = \{\mu_{i,j} | i = 1, \ldots, N, j = 1, \ldots, C\}$ is the *fuzzy partition matrix*. In other words, $\mu_{i,j}$ represents the degree of membership between the feature set x_i and the j^{th} cluster, subject to the constraints that $\mu_{i,j} \in [0, 1]$, and $\sum_{j=1}^{C} \mu_{i,j} = 1 \quad (\forall j \in \{1, \ldots, N\})$. Our choice of the parameter C does not significantly change our final results for service clustering (we chose $C = 100$). We particularly want C to be as small as possible because C is the number of dimensions of our next step in the clustering process. We also want it to be large enough to convey enough information about different connections. We have 5 features for each connection, and if each of these features could get 3 values (small, average, and big) we would have $3^5 = 243$ different types of connections. Similarly if each feature could get two values, we can have $2^5 = 32$ connection types. We selected 100 as a value that represents a compromise between these two numbers.

The parameter $m \in [0, \infty]$ is the *weighting exponent*, specifying the fuzziness of the clusters, and $d_{i,j} = \|x_i - v_j\|$ is the Euclidean distance between the feature set x_i and the cluster center v_i.

Finding network services of the same type. Finally, PARIS identifies the same type network services using a second clustering algorithm. This is done by grouping network services based on the similarity of their network flows. To this end, for each service S_i, the system analyzes the X^{S_i}, which is the set of all feature vectors x_{S_i} related to that service. More precisely, PARIS examines all flows related to a service to compute an *average fuzzy membership* (AFM) vector μ^{S_i} for S_i. Intuitively, the AFM vector captures how the flows associated with a single service are distributed over the different types of network traffic (clusters) computed in the previous step.

More formally, to derive μ^{S_i}, which is a length-C vector, we compute the average of the rows of U that correspond to the members of X^{S_i}:

$$\mu_i^S = \frac{\sum_{j_n | x_{j_n} \in X^{S_i}} \mu_{j_n}}{N_i} \quad (4)$$

where N_i is the size of X^{S_i}, and μ_i is the i^{th} row of the fuzzy partition matrix U.

Once the system has computed one AFM vector per service, we use a *k-means clustering* algorithm [25], with $k = 20$, to classify them. We selected k as the expected number of different services we would expect to see in the target network.

3.5 Meta-Missions

A meta-mission is an abstraction of the network mission. For instance, a web mail service composed of an IMAP service, an SMTP service, an LDAP service, and an HTTP service would be considered a meta-mission. This meta-mission can be instantiated as a (concrete) mission by specifying the actual (concrete) network services involved. In other words, a meta-mission represents a mission type. It is often useful to know how many different types of missions are run in a network.

To extract meta-missions from previously-identified concrete missions, PARIS identifies all sets of missions that are composed of the same number of services with the same types. PARIS identifies the service type by clustering the services, as described above.

Finding meta-missions. Using the information about types of services, PARIS can determine meta-missions. More precisely, PARIS considers two equal-sized missions M_1 and M_2 to be instances of the same meta-mission M if, for any service $S_i^1 \in M_1$, there is a service $S_j^2 \in M_2$ such that S_i^1 and S_j^2 are of the same type. Two services S_1 and S_2 are considered of the same type, if either they are in the same service cluster, or they use the same port number.

3.6 Detecting Backups

Backup services are frequently deployed to ensure the availability of critical network services. Given that there might be more than a single administrator responsible for a large network, and people might move on after having deployed backup services, it would be beneficial for a situation awareness system to automatically identify backups. Also, by automatically finding backup services, one can relieve administrators from having to specify their presence manually.

Knowledge about backup services is important, because these services might not produce a lot of traffic (since they are inactive for most of the time). Therefore, they might be discarded by tools that focus solely on traffic and activity volume. However, they are often critical components for network missions and should receive attention similar to the primary (currently active) services.

We distinguish two kinds of backup services: *active/passive* (A/P) backups and *active/active* (A/A) backups. In the case of A/P backups, one server acts as the main server, while the other(s), i.e., the backup(s), only becomes active when the main server experiences difficulties. In the case of an A/A backup scenario, two (or more) servers provide the service simultaneously, sharing the service load. Failing one of the A/A servers shifts the service load to the other backup(s).

3.6.1 Detection of A/P Backups

PARIS uses a combination of two approaches, negative correlation and failure correlation, to detect A/P backup services.

Negative activity correlation. *Negative correlation* indicates the degree to which two services activities are inversely related. This captures the activity behavior of the A/P backup services, because a backup service becomes active only when the main service is not functioning. We define the negative correlation of two services S_i and S_j as:

$$NC(S_i, S_j) = \max\{C^P(R_{S_i}, R_{S_j}), C^P(P_{S_i}, P_{S_j}), C^P(B_{S_i}, B_{S_j})\} \quad (5)$$

where $C^P(\cdot,\cdot)$ is the Pearson product-moment correlation, defined previously in Equation (2). As before, for a service S, the time sequences $R(S)$, $P(S)$ and $B(S)$ are the number of requests, the number of packets, and the number of bytes processed by S, respectively, evaluated for non-overlapping evaluation slots (Δ). PARIS declares two services to be A/P backups if their negative correlation is less than a threshold η_{NC} (we set η_{NC} to -0.49726. We looked into the correlation matrix from different time periods of normal network operation and verified that even though the correlation becomes negative in some cases, it does not exceed -0.49726).

3.6.2 Detection of A/A Backups

If two (concrete) missions are instances of the same meta-mission, and they share a large fraction of common services, the uncommon services are likely to be active/active backup services. Consider the case where PARIS detects two concrete instances of a meta-mission where most services are the same, and, in addition, the services that are different are of the same type. The fact that the additional services are of the same type and always appear together with the remaining services indicates that these services are an integral part of the mission. Yet, the multiple services do not appear to work together. Hence, we assume that they are replacements for each other; which is exactly the case with A/A backups.

PARIS examines every pair of missions M_1 and M_2. When these missions belong to the same meta-mission *and* they share at least 60% of the same services, the remaining services are checked for the presence of A/A backups. More formally, we define $S^{M_i - M_j}$ as the set of services that belong to the mission M_i but not to the mission M_j. PARIS declares two services $S_1 \in S^{M_1 - M_2}$ and $S_2 \in S^{M_2 - M_1}$ to be A/A backups if either they have the same type (they belong to the same service cluster) or they use the same port number.

4. RANKING NETWORK SERVICES

In this section, we discuss how our system ranks network services based on their importance. The ranking process operates in two steps. First, we compute initial importance scores for each service. These initial scores are then updated based on dependencies between services.

4.1 Initial Importance Scores

In the first phase, we compute an initial importance score for each network service. To this end, we use the following *service features*:

- F_1–F_4: the number of bytes and packets sent and received by the service,
- F_5: the number of requests handled by the service,
- F_6: the number of network missions the service has been involved in,
- F_7: the number of failures of the service (if available).
- F_8: the number of clients of the service (if available).

The features F_1–F_5 are directly taken from the corresponding service profiles. A service is considered more important when it is involved in more activity. Here, the activity is based on simple, network-level statistics. The sixth feature F_6 leverages information about relationships with other services. In particular, when a service is involved in more missions, we expect its importance score to increase. The last feature F_7 captures the number of failures that a certain service has experienced during the analysis period. To count the number of failures, we leverage the information from network health monitoring devices. If this data is not available, the value for this feature is zero for all services. We consider services that fail more often to be more "important." [3]

We normalize each feature value F_i into the range of $[0, 1]$. We then use a simple, weighted sum over all feature values for a service S to determine this service's initial importance score $\hat{I}^s$:

$$\hat{I}^s(S) = \sum_{m=1}^{8} \alpha_i F_i^N(S) \quad (6)$$

α_i is the weight for the i^{th} feature and $F_i^N(S)$ is the normalized value of F_i for the service S (we currently set all weights to 1, leaving the identification of possibly better weights to future work).

4.2 Final Importance Scores

In this step, we compute the final importance scores for network services. To this end, we use the detected service dependencies and backup relations to build a service *relationship graph* and we use it to update the initial importance scores. The nodes of the relationship graph are services. There is an edge from S_1 to S_2 in the relationship graph if there is a relationship between S_1 and S_2. In the following, we discuss how service dependencies and backups lead to edges in this graph.

Service dependency relationships. A service S_1 *depends* on another service S_2 if a failure in service S_2 disrupts the activity of S_1. We use two different ways to determine service dependency relationships.

When two services are part of the same mission, we introduce a corresponding edge into the relationship graph.

As a second mechanism, we use the simultaneous failure of two services as another indication of their dependency. In our current implementation, PARIS uses failure reports sent by a network monitoring device (Nagios) to detect service failures. More precisely, PARIS analyzes the failure logs collected during the analysis period T_E. We consider a service S_1 to depend on another service S_2 if at least $f = 50\%$ of S_2's failures *co-occur* with a failure of S_1. We consider two failure messages to co-occur if they are reported within a time window smaller that T_f (we set $T_f = 1min$, considering the time accuracy of the Nagios reports). If two services are found that have such a significant correlation of their failure reports, we introduce a corresponding edge into the relationship graph.

[3]In this case, a higher importance implies that the services is more fragile, and hence, requires more attention from a system administrator.

Table 1: The parameters of PARIS

Symbol	Description	Value
T_E	Analysis period	1 month
T_I	Length of a time series interval	1 hour
Δ	Length of time slot for time series	5 min
η_S	Service extraction threshold	30
η_C	Service correlation threshold	0.49726
η_M	Mission threshold	24
K	Number of service clusters	20
η_{NC}	Negative correlation threshold	-0.49726
γ	Weight factor of link analysis	5

Backup relationships. If service S_1 is a passive backup of service S_2, we add an edge from S_2 to S_1. The intuition behind this step is that if service S_2 has a backup, it loses some of its importance while the backup gains some of this importance.

Propagating scores. Once the network relationship graph is generated, PARIS uses Google's PageRank link analysis algorithm [33] to propagate initial importance scores. The intuition behind propagating importance scores in the relationship graphs is simple; a service that important services depend on is also important.

Once PARIS has performed the adjustments for service dependencies and backups, we use the final values of the nodes in the relationship graph as the final importance scores for the network services.

5. IMPLEMENTATION AND EVALUATION

We implemented PARIS as a system in Python and evaluated its performance by running it on the network monitoring data collected from a large university department network. Our dataset contains 38.5 GB of NetFlow records, gathered over a period of five months. Overall, the dataset captures more than 1.6 billion connections and 593 unique internal IP addresses. Out of the 1.6 billion connections, 1.25 billion connections were between an internal and an external host, while 350 million connections were between two internal hosts. We also had access to the firewall configuration files of the department, which contain 1,141 ACL rules. Moreover, we had access to approximately 120 thousand Nagios alerts that were gathered during the same one-month period.

5.1 Experimental Results

PARIS took about five hours to analyze one-month of network data; this could be further improved by utilizing a more powerful machine and optimizing the code. Table 1 summarizes the parameters used for our experiments, as discussed throughout the paper.

In the first step, PARIS extracted 156 network services. In the next step, PARIS checked for relationships between services. In our dataset, we found 4,049 candidate missions

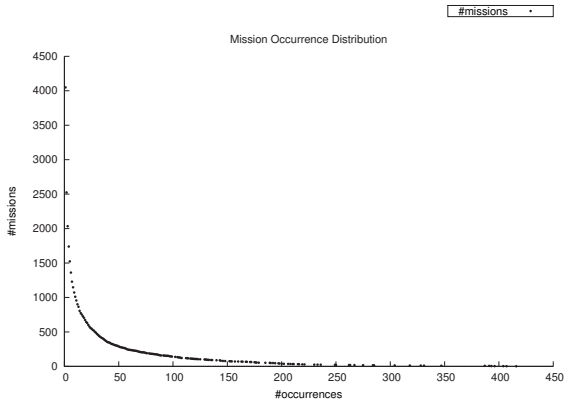


Figure 2: Mission frequency distribution.

(sets of correlated services). Figure 2 shows the distribution of the frequency of these candidate missions. The horizontal axis shows the number of occurrences x for missions, and the vertical axis shows the number of missions y that occur at least x times.

Looking at Figure 2, we see that the threshold $\eta_M = 24$, which is the cut-off between infrequent and frequent missions, is close to the inflection point of the graph. Using this threshold for the first filtering step, we remove 3,338 infrequent candidate missions. A large fraction of these 3,338 missions (2,609, or 78.2%) are supersets of more frequent missions (those on the right of the cut-off point). A mission X is a superset of another mission Y when X contains all services of Y and at least one additional service. Typically, such infrequent supersets are detected when an unrelated service S happens to be active at the same time that a "true" mission is operating. In this case, PARIS will correlate and incorrectly connect these services together. The first filtering step removes (most of) these spurious relationships.

Only 729 of the removed candidate missions in the first filtering step were not supersets of frequent candidates. In 531 of these 729 cases, the missions involved port 22 (`ssh`) and port 111 (`portmapper`). The infrequent `ssh` missions typically capture cases in which someone logged into a remote machine and initiated some actions or tunneled some traffic. The `portmapper` missions are introduced because portmapper is very frequently used, and hence, gets correlated with other, independent services that happen to be active at the same times. We only found two interesting cases that PARIS arguably missed. In one case, a set of machines was using a P2P protocol to exchange data for two hours. In the second case, a set of Hadoop machines were working together over a period of six hours. Notice that we found Hadoop machines also involved in more frequent activities, but these missions included additional (non-Hadoop) machines as well.

In the second filtering step, PARIS examines the remaining 711 missions and removes an additional 594 missions that are subsets of other frequent candidate missions. A mission X is a subset of Y if it contains a subset of Y's services. The second filtering step is useful to remove instances in which PARIS has detected most of the services that make up a mission, but, because of noise in the data, missed one that should have been included as well. We manually inspected the 594 missions. As expected, the removed missions were all parts of more complete missions that PARIS retained. Hence, no valuable information was lost. Finally, as its output, the mission extraction process produced 117 network missions.

Table 2 lists all the identified network operations, together with their corresponding numbers of constituent missions. This process was done manually, based on the knowledge about the domain and the application protocol semantics of different services. We performed this analysis to be able to present our results in a more succinct fashion. However, information about the missions themselves would already provide significant insights into the major tasks of the network.

Looking at Table 2, it can be seen that some missions map directly onto an operation. In other, more complex operations, we have identified multiple missions that are all related to a single operation. In the case of the "web operation", we found 63 individual missions. The individual missions were related to communication between the web server and storage servers (such as the NFS file services and Hadoop), authentication tasks (including LDAP), and DNS. Given that there were multiple services involved for each service type, we observed multiple combinations among individual groups of services. Most operations make immediate sense for a university network when looking at the services that are involved (web, mail, configurations with `cfengine`, ...). We discussed the extracted operations with the administrator of the network, and he verified the correctness of all but two extracted operations. These two operations, which the administrator was not aware of, were the malware analysis operation and the cloud operation.

We looked at the involved IP addresses and further tracked down these operations. We found that the malware analysis operation involved three machines, the actual analysis machine that was running malware samples, the MySQL server to store results, and a web server through which samples were submitted by external sources. The operation is important for the research group who runs this analysis infrastructure, and it was running for most of the month. The second mission involved four machines that were likely running cloud computing services (the machines were named `eucalyptus-*`, based on the popular, open-source cloud computing package). These machines worked together intensely for a total of 27 hours during the analysis period.

Our analysis confirmed that we found the key operations that the university runs. In addition, we found two interesting (and, for the involved parties, important) operations that the system administrators were unaware of.

Backup services. We also checked for backup services. Our system was able to identify four backup services in the organization. In particular, PARIS identified an NFS backup, an LDAP backup, a zookeeper server backup, and a main web server backup. The network administrator again verified that PARIS had detected *all* of the organization's backup servers and that no false positive backup was detected.

Ranking services and hosts. Using the information about the network services and their relationships, we computed the importance scores for each service and host.

In Table 3, we show the Top-10 services, given their final importance ranking. Along with their final rankings, the table also shows the initial rankings of these services (Column 3), the services that were in that location before computing final scores (Column 5), and the new ranking of those services based on the final scores (Column 6). For example,

Table 3: The importance ranking of the Top-10 services.

Final top 10			Initial top 10		
Final	Name	Init	Init	Name	Final
1	DNS	3	1	NFS1	2
2	NFS1	1	2	main web	4
3	cfengine	12	3	DNS	1
4	main web	2	4	Hadoop	9
5	LDAP30	9	5	LDAP36	6
6	LDAP36	5	6	LDAP12	15
7	fileserver1	15	7	web37	12
8	fileserver2	10	8	DHCP	58
9	Hadoop	4	9	LDAP30	5
10	NFS2	11	10	fileserver2	8

the first row means that the DNS server's final ranking is 1, while it was at location 3 before the final score adjustments. Also it shows that before the final adjustment, server NFS1 was at rank 1 and after score readjustment has moved to rank 2. This is to demonstrate the difference between initial and final rankings. In general, we observe that the importance scores of those services that depend on many other (important) services increase. For instance, the table shows the effect on the ranking of the *cfengine* service. It rises in importance because it was part of multiple missions and forms the foundation for the correct functioning of many servers. As another example, we can see that the DHCP service has dropped significantly in the ranking table, since not many other services depended on it. This is because the department's policy is to assign static IP addresses to all important network services.

We also attempted to validate the correctness of the service ranking with the network administrator of the organization. In particular, we asked the administrator's opinion about the Top-10 important services; all of the mentioned services appear in the Top-20 important services ranked by our system. Next, we presented our list of Top-10 services to the administrator and he verified that all of the extracted services and hosts are among the top important services of the organization.

PARIS can also compute importance scores for individual hosts. This score is based on the importance of the network services $\{S_1, ..., S_i\}$ that a host H provides, and it is computed as:

$$I^h(H) = \sum_{j=1}^{i} I^s(S_j) \quad (7)$$

Table 4 shows the importance rankings of the hosts. The results were verified with the network administrators as well.

6. DISCUSSION

In this section, we address possible complexity and generality issues of PARIS.

6.1 Complexity

Although computing the activity correlation for every pair of services has a quadratic complexity in terms of the number of services, this kind of analysis is necessary only when any possible pairing of services can be a valid one. Fortunately, in reality, the computer networks are structured hierarchically and mission implementation often times remains local in these clusters. Therefore, these clusters of services can be analyzed separately and independently, which makes our analysis linear in terms of the number of clusters, while quadratic to the number of services inside each cluster.

Table 4: The importance scores of the hosts. The host IPs are replaced with their services to anonymize the data.

Host's services	Final score	Rank
NFS, NetBIOS, SMTP, IMAP	3.760	1
NFS1	3.250	2
main web	1.741	3
DNS, DHCP	1.431	4
NFS2	1.386	5
NetBIOS	1.328	6
Hadoop	1.326	7
cfengine	0.971	8
LDAP1	0.930	9
print server	0.913	10

Another potential issue is that maximal clique listing for a dense graph can take exponential time in terms of the number of nodes of the graph. To reduce the complexity of the clique detection, we first delete all the non-frequent edges from all activity correlation graphs. An edge that is not frequent itself cannot be part of any frequent clique. Listing all maximal cliques of a sparse graph (i.e. with low degeneracy) is linear in number of its nodes [14]. In Section 3.3, we also showed that the probability of a random pair of services being recognized as a correlated pair (i.e. an edge in the graph) can be reduced to an arbitrarily small value by increasing the data points or by increasing the correlation threshold to the corresponding value. Therefore, maximal-clique listing complexity is not an intrinsic problem with our solution and can be avoided by parameter tuning.

6.2 Generality

We can divide our work into three layers of abstraction:
1- The underlying hypothesis: A mission is built of several services working together to achieve the same goal. This property leads to synchronized activity among services in the same mission. Missions can be detected by any statistical tool that detects synchronized activities, and they are recognizable as cliques in the synchronized-activity relation graph.
2- Synchronized-activity detection: Using the number of connections/packets/bytes as activity indicator and using Pearson correlation as synchronized-activity measure.
3- The thresholds and constants chosen to maximize the system accuracy: service-extraction threshold, mission threshold, etc.

The more abstract the design is the more general it becomes. We expect the underlying hypothesis to be true for new datasets from other networks, because it is an intrinsic property of the missions, which does not depend on the size of the network, types of services, and data gathering devices. In contrast, the thresholds and constants of the system have been selected to improve the accuracy of PARIS using a specific dataset, and can be tuned for other datasets to achieve better results.

7. RELATED WORK

Table 2: Network operations, together with the number of involved missions.

Operation name	Components	Number of Missions
email operation	Web, SMTP, DNS, and IMAP server	3
web operation	Web, DNS, LDAP, and Hadoop server	63
data center configuration operation	cfengine, LDAP server, and file server	14
roaming profile operation	NFS, LDAP, and DNS server	32
malware analysis operation	Analysis, Web and MySQL server	2
print operation	SNMP and print server	1
web backup operation	Web and LDAP server	1
cloud operation	cloud services	1

Situation awareness has been well studied in different research areas [12, 13, 27, 35, 4, 16, 32, 40, 23, 37, 17]. In particular, Salerno et al. discuss a situation awareness model for military applications and demonstrate its applicability to global monitoring and cyber awareness scenarios [23]. Tadda et al. refined the proposed models to apply them to cyber situation awareness [16]. In [37], Salerno reviews a number of metrics being used in information fusion frameworks and evaluates their applicability to the assessment of situations.

Holsopple et al. propose TANDI [21], a threat assessment mechanism for network data and information . Based on TANDI, Holsopple et al. also propose a situation and impact awareness system, FuSIA, which aims to enhance network situation awareness by providing plausible estimates of the future actions of the ongoing attacks [22]. FuSIA uses *threat projection algorithms* to derive *plausibility scores*, which analyze the capability and opportunity of ongoing attacks. Then, FuSIA combines the plausibility scores in order to provide a final fused estimate of the future situation. FuSIA is not fully automated, as it is not able to detect missions. In fact, a security analyst needs to manually provide FuSIA with mission information; this is in contrast to the automatic identification of mission information by Paris. Goodall et al. introduced Camus [18], which automatically maps cyber assets to missions and users by combining different data mining, inference, and fusion approaches. Camus, however, is based on the strong assumption that the asset and mission information is available in different formats and in different locations of the network. This requires frequent involvement of the network administrator, in order to ensure the availability and freshness of the data. Gomez et al. take a similar approach by automating the assignment of intelligence, surveillance, and reconnaissance assets to specific military applications [17]. Lewis et al. propose an alternative mission reference model in order to map the cyber assets to the missions using a mathematical constraint satisfaction approach [29].

Grimaila et al. study situation awareness by focusing on the information assets, instead of the cyber assets, and propose a cyber-damage assessment framework [19]. This, however, requires manual definition and prioritization of the operational processes and the information assets. Holsopple et al. survey the past efforts in situation assessment ranging from visualization to algorithmic threat projection, and they describe the need to associate situation assessment processes and models with some requirements needed to enhance the situation awareness [20]. The paper argues the need for effective automated processes that estimate and project the situation, taking into account the domain-specific concerns from the analysis. The paper also suggests the use of interactive visualization of the situation, threat, and impact for different application models.

Other research has targeted the classification of network applications based on NetFlow information [36, 28, 38, 8]. In particular, in order to classify and profile network applications, Lampinen et al. [28] propose a network application clustering algorithm that uses NetFlow information from the monitoring devices.

It should be mentioned that there is a relevant body of research trying to classify network applications based on header and payload full information of individual packets [26, 5]. Paris, however, only needs lower-level NetFlow data to perform its analysis, requiring much lower resources for its operation, e.g., lower CPU, storage, and bandwidth.

Some previous work [31, 24, 10, 3, 9, 34, 2, 15, 7, 30, 42] tried to address the problem of situation awareness by detecting hidden dependencies among network services.

8. CONCLUSIONS

In this paper, we developed a new method for providing automatic situation awareness for computer networks. We designed Paris which automatically identifies network services, finds their dependencies, and ranks them based on their importance towards network missions. This enables network administrators to make speculations about the future status of the network and to prioritize defensive and corrective actions. We implemented Paris and validated its performance by deploying it on a large real-world network.

9. ACKNOWLEDGMENTS

This work was supported by National Science Foundation (NSF) under grant CNS-1408632, and Army Research Office (ARO) under grant W911NF-09-1-0553.

10. REFERENCES

[1] NetFlow protocol. http://www.manageengine.com/products/netflow/help/cisco-netflow/netflow-ios-versions.html.

[2] P. Bahl, P. Barham, R. Black, R. Ch, M. Goldszmidt, R. Isaacs, S. K, L. Li, J. Maccormick, D. A. Maltz, R. Mortier, M. Wawrzoniak, and M. Zhang. Discovering dependencies for network management. In *In Proc. V HotNets Workshop*, 2006.

[3] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. *SIGCOMM Comput. Commun. Rev.*, 37, 2007.

[4] P. Barford, M. Dacier, T. G. Dietterich, M. Fredrikson, J. Giffin, S. Jajodia, S. Jha, J. Li, P. Liu, P. Ning, X. Ou, D. Song, L. Strater, V. Swarup, G. Tadda, C. Wang, and J. Yen. Cyber SA: Situational Awareness for Cyber Defense. In S. Jajodia, P. Liu, V. Swarup, and C. Wang,

editors, *Cyber Situational Awareness*, Advances in Information Security. Springer US, 2010.

[5] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06, pages 6:1–6:12, New York, NY, USA, 2006. ACM.

[6] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[7] A. Brown, G. Kar, G. Kar, and A. Keller. An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Environment. In *In Seventh IFIP/IEEE International Symposium on Integrated Network Management*, 2001.

[8] U. Chaudhary, I. Papapanagiotou, and M. Devetsikiotis. Flow classification using clustering and association rule mining. In *Computer Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD), 2010 15th IEEE International Workshop on*, 2010.

[9] M. Y. Chen, A. Accardi, E. KÄścÄśman, J. Lloyd, D. Patterson, A. Fox, and E. Brewer. Path-Based Failure and Evolution Management. In *In Proceedings of (NSDIâĂŹ04)*, 2004.

[10] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl. Automating network application dependency discovery: experiences, limitations, and new solutions. USENIX Association, 2008.

[11] T. Coladarci, C. Cobb, E. Minium, and R. Clarke. *Fundamentals of Statistical Reasoning in Education*. Wiley/Jossey-Bass Education. John Wiley & Sons, 2010.

[12] M. R. Endsley. Design and Evaluation for Situation Awareness Enhancement. In *Proceedings of the Human Factors Society 32nd Annual Meeting*, volume 1 of *Aerospace Systems: Situation Awareness in Aircraft Systems*, 1988.

[13] M. R. Endsley. Towards a Theory of Situation Awareness in Dynamic Systems. *Human Factors*, 37, 1995.

[14] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *ISAAC (1)*, pages 403–414, 2010.

[15] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica. X-trace: A pervasive network tracing framework. In *In NSDI*, 2007.

[16] G. Tadda, J.J. Salerno, D. Boulware, M. Hinman. and S. Gorton. Realizing situation awareness within a cyber environment. In *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2006*, volume 6242. SPIE, 2006.

[17] M. Gomez, A. D. Preece, M. P. Johnson, G. de Mel, W. W. Vasconcelos, C. Gibson, A. Bar-Noy, K. Borowiecki, T. F. L. Porta, D. Pizzocaro, H. Rowaihy, G. Pearson, and T. Pham. An ontology-centric approach to sensor-mission assignment. 2008.

[18] J. R. Goodall, A. D'Amico, and J. K. Kopylec. Camus: Automatically mapping Cyber Assets to Missions and Users. *MILCOM 2009 - 2009 IEEE Military Communications Conference*, 2009.

[19] M. Grimaila, R. Mills, and L. Fortson. Improving the Cyber Incident Mission Impact Assessment Processes. In *4th Annual Workshop on Cyber Security and Information Intelligence Research*, 2008.

[20] J. Holsopple, M. Sudit, M. Nusinov, D. Liu, H. Du, and S. Yang. Enhancing situation awareness via automated situation assessment. *IEEE Communications Magazine*, 48, 2010.

[21] J. Holsopple, J. Yang, , and M. Sudit. TANDI: Threat assessment of network data and information. In *SPIE, Defense and Security Symposium*, 2006.

[22] J. Holsopple and S. Yang. FuSIA: Future situation and impact awareness. In *Information Fusion, 2008 11th International Conference on*. IEEE, 2008.

[23] J.J. Salerno, M.L. Hinman, and D.M. Boulware. A situation awareness model applied to multiple domains. In *Proceedings of SPIE*, 2005.

[24] S. Kandula, R. Chandra, and D. Katabi. What's going on?: learning communication rules in edge networks. *SIGCOMM Comput. Commun. Rev.*, 2008.

[25] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[26] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. *SIGCOMM Comput. Commun. Rev.*, 35:229–240, August 2005.

[27] G. Klein and B. Crandall. Recognition-Primed Decision Strategies. ARI research note 96-36, 1996.

[28] T. Lampinen, H. Koivisto, and T. Honkanen. Profiling network applications with fuzzy c-means clustering and self-organizing map. In *FSKD'02*, 2002.

[29] L. Lewis, G. Jakobson, and J. Buford. Enabling cyber situation awareness, impact assessment, and situation projection. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, 2008.

[30] J.-G. Lou, Q. Fu, Y. Wang, and J. Li. Mining dependency in distributed systems through unstructured logs analysis. *SIGOPS Oper. Syst. Rev.*, 44:91–96, March 2010.

[31] A. Natarajan, P. Ning, Y. Liu, S. Jajodia, and S. E. Hutchinson. NSDMiner: Automated Discovery of Network Service Dependencies. In *In proceedings of IEEE International Conference on Computer Communications (INFOCOM '12)*.

[32] P. Porras, M. Fong, and A. Valdes. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In *Proceedings of the International Symposium on the Recent Advances in Intrusion Detection*, 2002.

[33] L. Page. PageRank: Bringing order to the web. Stanford Digital Libraries Working Paper 1997-0072, Stanford University, 1997.

[34] L. Popa, B. gon Chun, J. Chandrashekar, N. Taft, and I. Stoica. Macroscope: End-Point Approach to Networked Application Dependency Discovery, 2009.

[35] J. Rasmussen. Skills, Rules, and Knowledge; Signals, Signs and Symbols, and Other Distinctions in Humans Performance Models. *IEEE Transactions on Systems, Man and Cybernetics*, 13, 1983.

[36] D. Rossi and S. Valenti. Fine-grained traffic classification with netflow data. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, IWCMC '10, New York, NY, USA, 2010. ACM.

[37] J. Salerno. Measuring Situation Assessment Performance through the Activities of Interest Score. In *Information Fusion, 2008 11th International Conference on*, 2008.

[38] S. Song and Z. Chen. Adaptive Network Flow Clustering. *2007 IEEE International Conference on Networking, Sensing and Control*, 2007.

[39] Symantec. Internet security threat report. Technical report, April 2012.

[40] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing. Ranking attack graphs. In *Recent Advances in Intrusion Detection*, 2006.

[41] J. Yuan and S. Ding. An alerts correlation technology for large-scale network intrusion detection. In Z. Gong, X. Luo, J. Chen, J. Lei, and F. Wang, editors, *Web Information Systems and Mining*, volume 6987 of *Lecture Notes in Computer Science*, pages 352–359. Springer Berlin Heidelberg, 2011.

[42] A. Zand, G. Vigna, R. A. Kemmerer, and C. Kruegel. Rippler: Delay injection for service dependency detection. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, pages 2157–2165, 2014.