# Enhancing Graph-based Automated DoS Attack Response [1]

Gabriel KLEIN, Marko JAHNKE, Jens TÖLLE [a,2]
Peter MARTINI [b]
[a] *Research Institute for Communication, Information Processing and Ergonomics (FGAN-FKIE), Germany*
[b] *Insitute of Computer Science IV, University of Bonn, Germany*

**Abstract.** Timely and appropriate reactions to detected denial-of-service attacks against computer networks are crucial in both civilian and military settings. GrADAR is an intuitive graph-based approach for assessing the effects of DoS attacks against computer networks so that response measures can be automatically selected without human intervention. However, GrADAR has limitations insofar as implicit effects of countermeasures are only taken into account by propagation towards user nodes. Possible effects in the other direction are only considered if they are explicitly specified. For this, they need to be exactly known in advance which is often infeasible. This contribution presents an extension to GrADAR, in which we consider resource workload and processing capabilities and their effects on resource availability. We incorporate workload measurements into the GrADAR model which are done by passive analysis of network traffic. We further augment the active availability probes with passive measurements. This ensures more accurate availability values because additional measurement traffic that might falsify results only needs to be injected when resources are currently not accessed.

**Keywords.** denial-of-service attacks, automated response, response evaluation, passive availability measurement

## Introduction

In recent years, the number of attacks against computer networks has steadily increased. They cannot only be observed in civilian scenarios (e. g. e-commerce or online banking) but also in military settings. Among these attacks, denial-of-service attacks are the most prevalent, often resulting in inaccessibility of services and/or entire networks. This can result in enormous financial losses, in case of commercial applications, and negatively impact battle readiness where military networks are concerned.

In typical wired networks, such attacks can be detected with a high degree of accuracy by incorporating intrusion detection systems into the networks' perimeter defence. Once detected, security personnel can then suitably react to these attacks. However, be-

[1] Published in: Proceedings of the Cyber Warfare Conference, Cooperative Cyber Defence Centre of Excellence (CCD-CoE), Tallinn, Estonia, 2009

[2] Corresponding Author: Gabriel Klein, FGAN-FKIE, Neuenahrer Str. 20, 53343 Wachtberg, Germany; E-mail: g.klein@fgan.de

cause of the increased speed and reliability, it is desirable to react to detected attacks in an automatic fashion. For the automatic selection of response measures, it is necessary to quickly and accurately estimate the effects of the respective countermeasure on the network resources.

In previous work, we have proposed GrADAR [1,2], an intuitive approach to create and maintain a model of a computer network and the availability of its resources from the observations of deployed monitoring systems. The graph-based model is able to express both the effects of DoS attacks and the results of available response measures prior to their application in the real-world network. Thus, the approach provides a methodology for automatically selecting response measures to detected attacks. The most appropriate response is chosen based on metrics which are well-known from the pragmatic view of network security officers.

This contribution proposes an extension to our previous GrADAR approach that seeks to incorporate the effects of network and resource workload into the availability estimation. This will permit a more detailed modelling of the current network state. Further, it will allow the specification of the effects of more complex DoS countermeasures. To further improve the availability measurements and reduce the workload placed on the network and resources, passive measuring is employed instead of active probing.

The rest of this paper is structured as follows: section 1 introduces related work done in the area of passive measurements. Subsequently, section 2 gives an overview of the GrADAR approach as well as some of its limitations. Section 3 presents the proposed GrADAR extensions, describing workload and the measurement framework in more detail. Section 4 portrays first results and, following that, section 5 presents a summary and an outlook on future work.

## 1. Related Work

The area of passive network analysis has been a research area of interest for a number of years. In the context of this contribution, approaches concerned with the inference of server workload and the identification of network flows are of particular interest.

In [3], Barford and Crovella describe an architecture for actively and passively measuring the effects of Web server and network workload on the quality of client connections. They show that increasing network load causes a deterioration of connection quality. However, they observe a positive effect of server load on traffic characteristics and attribute this to a reduced burstiness of traffic.

Eriksson et al. [4] reiterate a methodology for determining the network structure by passive measuring of hop counts. They address the issues of missing hop count data and how to extract topological information from large hop count databases.

Passive measurement is also a popular method for determining the characteristics of network connections such as bandwidth, latencies or packet loss statistics. Seshan et al. [5] propose SPAND, a system in which sensors distributed throughout the network perform passive measurements and report to central performance servers where the data is collated. In [6], Lowekamp presents a report on the Wren project which deals with the development of network performance monitoring solutions. Here active and passive measurements of traffic statistics are combined to reduce the amount of artificial traffic wherever possible.

To passively determine availability and workload, observed traffic needs to be associated with the resources between which it flows. Distinct flows can, for example, be identified using NetFlow [7], but correlations between within the same or different flows also need to be ascertained subsequently. This is often done by simple payload inspection, although this potentially requires large amounts of memory. However, recent approaches for the identification of upper layer protocols include machine learning [8,9] and multi-scale gamma models [10].

## 2. GrADAR Overview

In the GrADAR approach, a simplified model of the real-world network is created in order to predict the effects of available response measures against denial-of-service attacks. Figure 1 shows an overview of the approach.

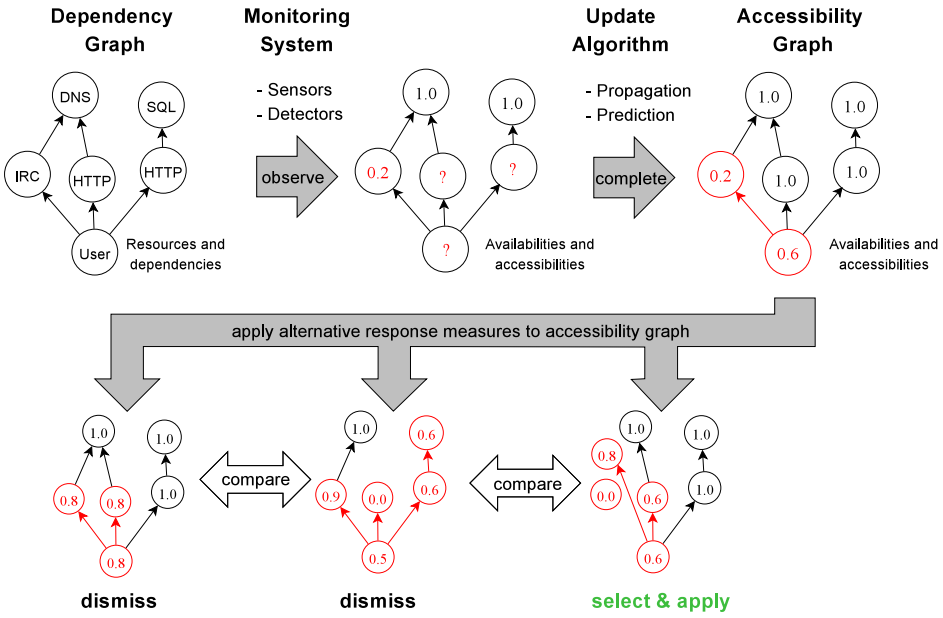



**Figure 1.** Schematic overview of the GrADAR approach.

### *2.1. Nomenclature*

The core concept of GrADAR is based on the *availability* of *resources*. Resources (as suggested in [11]) can be either services provided by hardware or software components (denoted as $\mathcal{S}$), or users (denoted as $\mathcal{U}$). Therefore, the set of resources is $\mathcal{R} = \mathcal{S} \cup \mathcal{U}$.

Each resource $r$ has an associated value $A(r) \in [0, 1]$, signifying the extent to which it is available to other resources. In [12] and [13], the concept of resource-typical transactions was proposed. We adopt this concept and define a resource's availability as the time needed for a transaction with the resource. Since a resource typically requires interaction

with other resources to function correctly, we assume that its availability is the result of two independent factors, an internal state (the *intrinsic* availability) and the values of other associated resources (the *propagated* availability). Thus, a resource's availability is defined as

$$A(r) = A_I(r) \cdot A_P(r) \tag{1}$$

for each $r \in \mathcal{R}$.

A resource $r$ may be dependent on other resources $s_1, \ldots, s_n$ (denoted as $r \triangleright s_1, \ldots, r \triangleright s_n$). In this case, the degree to which $r$ depends on each of these may vary [14,15] and can be specified by weighting the respective dependency. This can be formalised as

$$A_P(r) = D_r\left(w_{r,s_1}\left(A\left(s_1\right)\right), w_{r,s_2}\left(A\left(s_2\right)\right), \ldots, w_{r,s_n}\left(A\left(s_n\right)\right)\right), \tag{2}$$

where $D_r : [0,1]^n \rightarrow [0,1]$ is a dependency function and $w_{r,s_i} : [0,1] \rightarrow [0,1]$ are corresponding dependency weighting functions. In optimal conditions,

$$D_r\left(w_{r,s_1}\left(1\right), \ldots, w_{r,s_n}\left(1\right)\right) = 1.$$

A more detailed discussion of this can be found in [2].

### *2.2. Dependency Graph*

To represent the availability dependency relationships between the set of resources $\mathcal{R}$, the resources in the real-world network are modelled as a directed acyclic graph $\hat{G} = (\hat{V}, \hat{E})$ with $\hat{V} \subseteq \mathcal{R}$ and $\hat{E} \subseteq ((\mathcal{S} \cup \mathcal{U}) \times \mathcal{S})$. Its vertices correspond to the resources and the edges correspond to the dependency between the respective resources. These resource dependencies need to be determined beforehand, either analytically or experimentally. $\hat{G}$ contains an edge $(r, s)$ iff $r \triangleright s$. These edges are labelled with the corresponding weighting function $w_{r,s}$. This graph is called the *dependency graph* and reflects the ideal state of the network. Let $\hat{\mathcal{G}}$ be the set of all possible dependency graphs.

### *2.3. Accessibility Graph and Overall Availability*

A DoS attack typically affects the availability of resources. Thus, there is the possibility that some resources might no longer be accessible to others. Therefore, a second graph is required, the so-called *accessibility graph*. Mutual accessibility of a set of resources $\mathcal{R}$ is expressed by a graph $G = (V, E)$ with $V \subseteq \mathcal{R}$ and $E = ((\mathcal{S} \cup \mathcal{U}) \times \mathcal{S})$, in which an edge $(r, s)$ exists when a resource $s$ is directly accessible from $r$. The vertices $r \in \mathcal{R}$ of the accessibility graph are labelled with the corresponding resource's availability $A(r)$.

The availability of user nodes is interpreted as the user-perceived availability of the network. Since the network supports one or more users or groups of users in conducting a common mission, we define the overall availability of the network as the weighted average of all user nodes' availability values:

$$A(G) := \sum_{u \in \mathcal{U}} m(u) \cdot A(u),$$

where $m(u)$ is the *relative importance* of user $u$ to the common mission which needs to be determined beforehand or adaptively, and $\sum_{u \in \mathcal{U}} m(u) = 1$. Let $\mathcal{G}$ be the set accessibility graphs.

Usually, monitoring systems deployed in the network will only be able to observe availability values for some of the network's resources. This is especially true for users, for which an availability cannot be objectively measured. Thus, the availability of resources for which values cannot be observed need to be estimated. For a resource $r$ with $r \triangleright s_1, \ldots, r \triangleright s_n$, this estimation is done by propagating the availability values of the resources $s_1, \ldots, s_n$ in the inverse direction of the corresponding dependency relationship expressed in the dependency graph, i. e. along the edges $(s_1, r), \ldots, (s_n, r)$, and then calculating $A(r)$ according to equations (1) and (2) with $A_I(r) = 1.0$. This can be efficiently done, for example, with a depth-first search algorithm, starting from the user vertices and terminating at vertices with $deg_{\text{out}}(r) = 0$.

As opposed to the dependency graph, the accessibility graph shows the actual current state of the network.

### *2.4. Response Selection*

Once an attack has been detected, an appropriate reaction should be selected automatically. With the current dependency graph $\hat{\mathcal{G}}$ and the current availability graph $\mathcal{G}$, we define a *countermeasure* or *response measure* as a transformation

$$\theta : \hat{\mathcal{G}} \times \mathcal{G} \to \hat{\mathcal{G}} \times \mathcal{G}.$$

The dependency graph $\hat{G}' = (\hat{V}', \hat{E}')$ is obtained from $\hat{G}$ by adding or removing vertices or edges, and the accessibility graph $G' = (V', E')$ is derived from $G$ by also adding or removing vertices or edges but, additionally, vertex availability values can be changed. Let the set of available countermeasures be denoted as $\Theta$.

As response measures may be arbitrarily complex in nature, we assume that a single response measure $\theta$ can be divided into $N_\theta \in \mathbb{N}^+$ successive atomic *response steps* $\theta^{(i)}$:

$$\theta = \theta^{(1)} \circ \ldots \circ \theta^{(N_\theta)}.$$

Each of these real-world response steps $\theta^{(i)}$ corresponds to one of the graph transformation primitives mentioned above (adding/removing vertices or edges, setting availability). The effects of such a response step can be either directly associated with the action (*explicit impact*) or a result of changes in the environment due to the response step application (*implicit impact*).

For the automatic response measure determination, each available countermeasure $\theta \in \Theta$ is now applied in parallel to the current accessibility graph. For each change in the accessibility graph, the propagation algorithm mentioned above needs to be executed. The resulting graphs (so-called *response graphs*) are then compared with respect to different metrics, e. g. *expected response success* or *expected response costs* [2], and the most appropriate response is then applied to the real-world network.

The resulting dependency graph of one such GrADAR cycle is used as the input for the subsequent iteration. Thus, the process constitutes a so-called *closed-loop control system*.

### *2.5. Current Limitations*

During the validation of the GrADAR approach it became apparent that correctness and robustness could only be achieved if the precise effects of real-world countermeasures in the graph space could be accurately predicted. Because of the closed control loop structure of the approach, subsequent iterations of the loop would operate on incorrect input.

So far, the response measures for which effects were specified consisted only of blocking access to specific resources (e. g. closing a firewall port) or migrating resources to different locations (e. g. installing a new server to replace another). The effects of these operations on the graph could be specified fairly easily in terms of changed availability values or changing edges in the graph.

However, more complex behaviour of resources, such as the interactions between workload placed on a resource and its processing capacities and their effect on that resource's availability, are only expressible if fairly well known and specified in advance. This is because the effects are taken into account by propagating them through the accessibility graph according to the update algorithm. This imposes constraints in that effects can only be propagated in the reverse direction of the dependency relationship. Effects on resources in the other direction cannot currently be expressed. For example, blocking a DoS attack against a Web server at a firewall port would have the explicit impact of a 0.0 availability at the firewall port. This would be propagated to the (dependent) user node. However, a possible implicit impact could be an increased availability of the Web server. This is currently expressible only if the changes in availability are known beforehand and "hard-coded" into the graph transformation, something which is often impossible. It is desirable to predict such effects on resource availability dependent on the current network situation.

Furthermore, availability measurements are currently performed only through active probing, i. e. by sending requests to the respective resources and measuring and normalising the time required for an answer. This poses two problems. First, the measuring process itself produces workload for the network and the target resource, and second, it requires the measuring process to produce a traffic pattern which is representative for the specific resource.

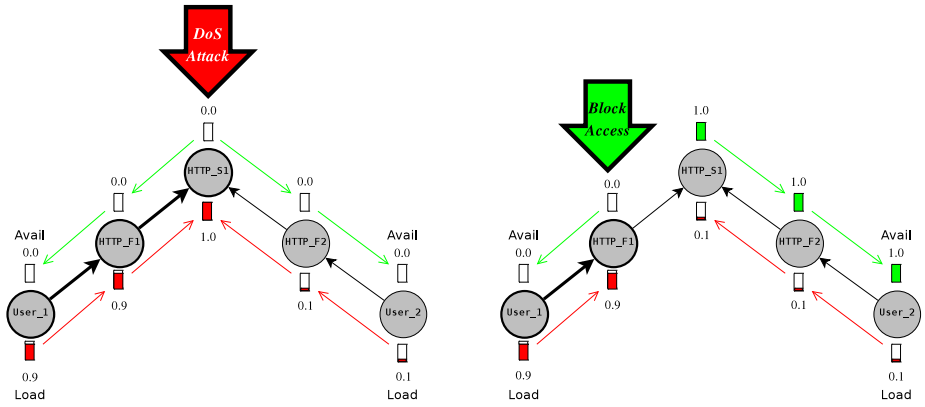
## 3. Beyond GrADAR: Improving Availability Estimation

Due to the limitations recounted in section 2.5, we propose to enhance the GrADAR approach by incorporating the effects of workload and resource capacity into the graph-based model and avoiding active probing in favour of passive availability measurements. We believe that through the enhancements described in this section the correctness and robustness of the approach can be significantly improved.

### *3.1. Solution Idea: Propagation of Workload*

To more accurately represent the graph-based equivalents of DoS attack response measures, we propose to incorporate the workload placed on the various resources into the GrADAR model. For this, the relationships between resource workload and its availabil-

ity need to be determined, especially in the area of a resource's processing capacity. Additionally, dependency relations between the workload of different resources should be investigated. If such relationships exist, this could allow the inference of workload values for resources of which workload cannot be directly measured. Similar to availability estimation, this could be done by propagation according to the workload relationships; see figure 2 for an example. Here, two users (or user groups), `User_1` and `User_2`, communicate with an HTTP server, `HTTP_S1`, via two separate firewall ports, `HTTP_F1` and `HTTP_F2`. In figure 2(a), the workload generated during a DoS attack by `User_1` is propagated (in the direction of the dependency relationship) to the firewall port and the HTTP server. Because of this workload, the HTTP server has an availability of 0.0 which, in turn, is propagated (against the dependency relationships) to the resources dependent on it. An example response to such an attack, namely blocking the attacker at the firewall, is depicted in figure 2(b). As a result, the workload generated by `User_1` is no longer propagated to `HTTP_S1` which causes an increase in the server's availability. This, in turn, is propagated to `User_2`.




(a) Workload-based DoS attack.

(b) Example response to DoS attack.

**Figure 2.** Example of workload propagation in the GrADAR model.

### *3.2. Workload Definition*

Before the workload $L(r)$ of a resource $r \in \mathcal{R}$ can be effectively measured, it needs to be defined in a suitable fashion. The dictionary defines workload as "the amount of work assigned to, or done by, a worker or unit of workers in a given time period" (The American Heritage Dictionary, 2nd Edition). Similar to the definition of availability as the normalised duration of a resource-typical transaction (see e. g. [12,13]), a resource's workload can thus be defined as the number of typical transactions a resource needs to process per unit of time. Since a networked application scenario contains multiple types of resources, different types of "work" need to be considered as each resource has transactions that are typical for it; for example, the number of concurrent transactions a Web server needs to process. Table 1 contains a listing of possible resources along with the availability and workload definition for each of them.

**Table 1.** Workload definitions for selected resources.

| Resource | Availability definition | Workload definition |
|---|---|---|
| IP stack | ICMP ping response time | IP packets/time |
| CMS | Delay for receiving backend content | Current active transactions |
| IRC server | Delay for connection, joining channel and sending a message | Current active transactions |
| DB server | Delay of query from Web server backend | Current no. of transactions |
| DNS server | Delay for result of lookup query | Requests/time |
| MAC layer | Interface up/down | Frames/time |
| CPU | Execution delay for application requiring CPU/memory/HDD | Average CPU load |
| Memory | | Average memory consumption |
| HDD | | Average consumed HDD capacity |

To adequately compare the workload of different types of resources, workload values need to be normalised:

$$\tilde{L}(r) = \frac{L(r)}{L_{max}(r)}, \tag{3}$$

where $L_{max}(r)$ is the maximum workload which a resource can adequately process within a certain time frame. This is closely related to the definition of availability (c. f. [1,2]) where request-response delays are normalised with respect to a maximum acceptable time from a user's perspective.

### *3.3. Measurement Framework*

The measurement of availability and workload is performed according to the framework outlined in the SDL [16] diagram depicted in figure 3.

Passive sensors at appropriate locations in the network (e. g. at central switches or a firewall) constantly observe passing traffic. Different *conversations* between consumers and providers of a service, and the *transactions* they comprise are identified by analysing packet headers and correlating certain fields, e. g. sequence numbers, IP addresses or port numbers. For each recognised transaction, various properties of the traffic such as average packet loss, transaction duration, average round-trip time, jitter, etc. can be used to evaluate the availability of a resource; c. f. [12] and [13] for details of how traffic properties can be used to make quantitative statements about the quality of a service. Thus, a resource's availability value can be updated after each completed transaction.

By logging the number of transactions for different services and/or protocols, these sensors can also establish a current workload for the observed resources. As already mentioned (c. f. table 1), the number of typical transactions within a specified period of time (or possibly the ratio of initiated vs. completed transactions) can serve as a workload metric.

The goal of our work is the identification of current threats to a network and the ability to react in near-real-time. Bearing this in mind, it seems advisable to consider the development of resource availability and workload over a customisable period of time rather than only the currently measured values. Using this as a base for decisions may reduce the likelihood of overreactions or false positives.

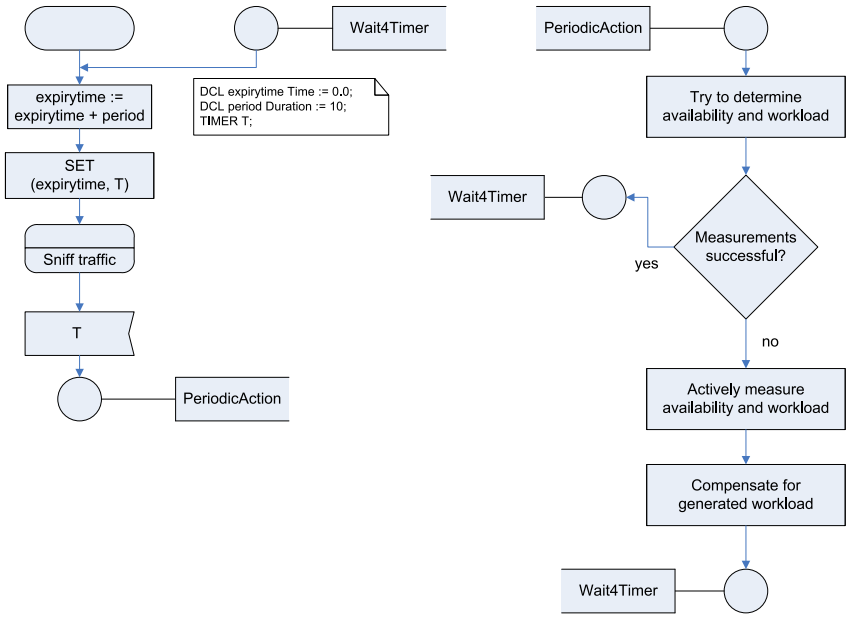



**Figure 3.** SDL diagram of the measurement framework.

The aforementioned description of passive measurements assumes that representative traffic for all relevant resources can always be observed. This may not be the case in real-world configurations, e. g. if clients are under EMCON in military settings. Thus, to always retain an overview of the current network status, active measurements need to be performed if no traffic was observed for a certain amount of time. In the case of availability, this is done via an active probe which consists of a (representative) request to the respective resource. The resulting availability is the normalised duration of this request. The resources' workload needs to be queried directly via appropriate interfaces at the resources themselves (e. g. via SNMP [17]), although this introduces a certain degree of dependency on specific applications and protocols. Note that, when performing active measurements, the load generated by these artificial requests needs to be taken into account and compensated when determining resource workload.

We are aware that both active and passive measurements have disadvantages. In the case of active probing this is the injection of additional traffic into the network resulting in increased workload for the targeted resources. On the other hand, when observing traffic passively, the volume of traffic to be dealt with is potentially prohibitively large. We try to avoid this by restricting ourselves to the analysis of only packet headers instead of entire packets.

## 4. Preliminary Results

We have performed first simple workload and availability measurements in a network topology depicted in figure 4. A client accesses a content management system (CMS) which, in turn, retrieves its data from a database server on a different host (scenario 1). In a second measurement scenario (scenario 2), both the CMS and the database are located

on the same host. In both cases, the servers are separated from the client by a firewall host. The servers hosting the Web and database servers are older-model single core machines and are thus not able to process requests entirely in parallel.

The values shown in figures 5 and 6 were obtained by generating an increasing number of concurrent requests per time frame. For each number of concurrent requests, the measurement process was repeated 300 times. The mean request-response durations were used as the basis for the availability calculations.

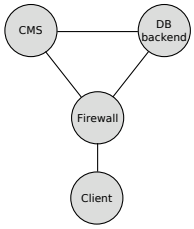



**Figure 4.** Server setup during measurements.

Figure 5 shows the measurement results for scenario 1. The normalised CMS workload and availability are plotted against an increasing number of concurrent requests directed at the Web server. At first, the CMS availability degrades linearly with the increase in workload. Beyond around 15 concurrent requests, an overload situation is entered, in which the availability remains at zero (shown as triangle-shaped data points in the diagram). At this point, the server cannot process requests within an acceptable time frame.

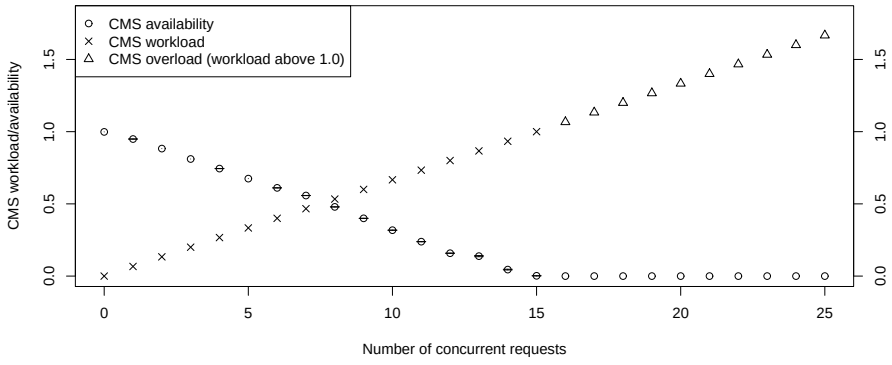



**Figure 5.** CMS workload and availability plotted against increasing number of concurrent requests; CMS and DB server on separate hosts.

In the second scenario, where the DB server is on the same host as the CMS server, the CMS availability degrades slightly more quickly (depicted in figure 6). This is most probably because both server processes share the same system's resources. Load processed by the CMS is partially transferred to the database, which, in turn, reduces the

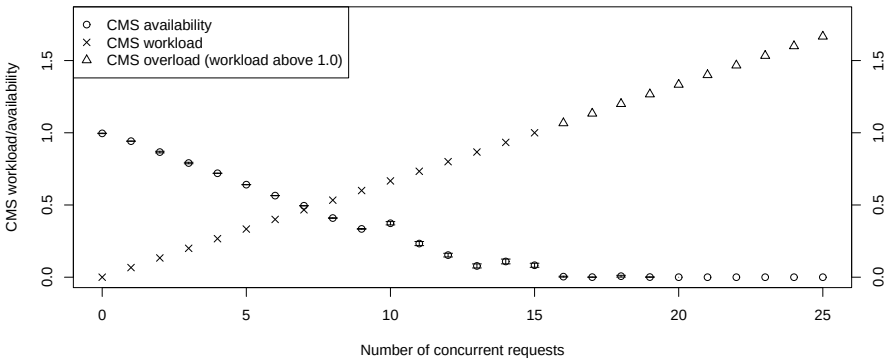
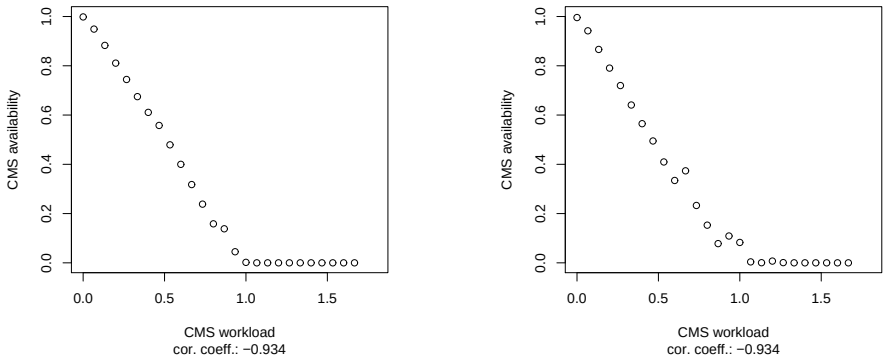



**Figure 6.** CMS workload and availability plotted against increasing number of concurrent requests; CMS and DB server on the same host.

available resources for the CMS. Also, the availability curve is less smooth. The small confidence intervals suggest that the reasons for the outliers are systematic. They could, for example, be caused by changes in scheduling policies. This is also true for the outlier observed in scenario 1.




(a) Separate CMS and DB servers (b) Common CMS and DB server

**Figure 7.** Relationship between resource workload and availability.

Figure 7 shows the relationship between workload and availability for both scenarios. In both cases, the correlation coefficient is very close to $-1$. This suggests the existence of a functional dependency between resource workload and availability. In this simple first example, we observe a linear dependency between the two values. However, more complex dependencies may exist, e. g. in the case of a multi-core system which is able to process multiple requests in parallel. Here, up to a certain workload, the availability should not be markedly impaired at all. Also, when considering other types of resources (e. g. the operating system kernel), binary relationships are possible, where a

resource remains fully available up to a certain workload, after which it immediately drops to zero.

## 5. Summary and Further Work

This contribution has discussed an extension to GrADAR, an approach for automatically assessing the effects of denial-of-service countermeasures. Including the effects of resource workload into the GrADAR model permits the specification of complex countermeasure effects.

Workload and availability measurements were performed in a simple DMZ-like setup which included machines representing a client, a firewall host and two servers. They were conducted using a passive monitoring solution capable of calculating resource workload and availability from observed network traffic. The results of these measurements indicate a possible functional dependency between resource workload and availability which justifies the incorporation of workload measurements into the GrADAR approach.

The work done regarding the GrADAR extension is of a preliminary nature. There are numerous aspects which need to be considered in future work. So far, the generated traffic used for measuring workload and availability consisted only of a steadily increasing number of concurrent clients requesting the index page of an e-commerce Web site. Representative traffic for such a scenario needs to be generated for a more detailed evaluation; e. g. according to a formal customer state model as depicted in figure 8 with different states for each type of viewed page and appropriate state transition probabilities. Also, possible dependencies between the workload of different resources needs to be investigated, e. g. workload placed on the CMS and its backend database.

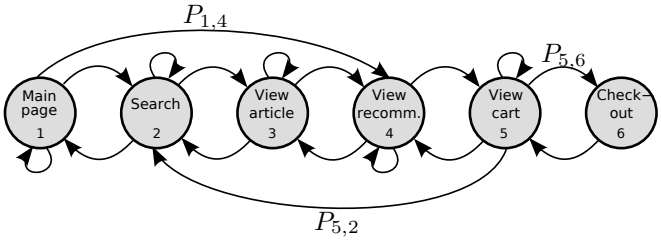



**Figure 8.** Possible state model underlying browsing by Web shop customers.

Where the passive analysis of traffic is concerned, problems may arise when only parts of conversations between resources can be observed, e. g. due to node movement in mobile ad hoc networks. In this case it might become necessary to harmonise the observations of multiple sensors distributed throughout the network.

## Acknowledgements

The authors wish to thank Christian Thul for his excellent development work and his valuable feedback.

## References


[1] M. Jahnke, C. Thul, and P. Martini. Graph based metrics for intrusion response measures in computer networks. In *Proc. of the 3rd LCN Workshop on Network Security. Held in conjunction with the 32nd IEEE Conference on Local Computer Networks*, Dublin, Ireland, October 2007.

[2] M. Jahnke, C. Thul, and P. Martini. Comparison and improvement of metrics for selecting intrusion response measures against DoS attacks. In A. Alkassar, editor, *Proc. of the Sicherheit2008 Conference*, Saarbrücken, Germany, April 2008.

[3] P. Barford and M. Crovella. Measuring Web performance in the wide area. *SIGMETRICS Performance Evaluation Review*, 27(2):37–48, September 1999.

[4] B. Eriksson, P. Barford, R. Nowak, and M. Crovella. Learning network structure from passive measurements. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pages 209–214, New York, NY, USA, 2007. ACM.

[5] S. Seshan, M. Stemm, and R. H. Katz. SPAND: Shared passive network performance discovery. Technical Report UCB/CSD-97-967, EECS Department, University of California, Berkeley, August 1997.

[6] B. B. Lowekamp. Combining active and passive network measurements to build scalable monitoring systems on the grid. *SIGMETRICS Performance Evaluation Review*, 30(4):19–26, 2003.

[7] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), October 2004.

[8] N. Williams, S. Zander, and G. Armitage. Evaluating machine learning algorithms for automated network application identification. Technical Report 060410B, Centre for Advanced Internet Architectures (CAIA), March 2006.

[9] P. Barlet-Ros, V. Carela-Español, E. Codina, and J. Solé-Pareta. Identification of network applications based on machine learning techniques. In *TNC 2008: Proc. of the Terena Networking Conference*, 2008.

[10] Y. Himura, K. Fukuda, K. Cho, and H. Esaki. Characterization of host-based traffic with multi-scale gamma model. In *Proc. of the 2nd CAIDA/WIDE/CASFI Workshop*, Seoul, South Korea, April 2009.

[11] T. Toth and C. Kruegel. Evaluating the impact of automated intrusion response mechanisms. In *ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference*, page 301, Washington, DC, USA, 2002. IEEE Computer Society.

[12] J. Mirkovic, P. Reiher, and A. Hussain. Measuring denial of service. In *Proc. of the ACM Workshop on Quality of Protection (QoP)*, pages 53–58. ACM Press, 2006.

[13] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W. Yao, and S. Schwab. Towards user-centric metrics for denial-of-service measurement. In *Proc. of the Workshop on Experimental Computer Science*, 2007.

[14] S. Bagchi, G. Kar, and J. Hellerstein. Dependency analysis in distributed systems using fault injection: Application to problem determination in an e-commerce environment. In *Proc. of the 12th Intl. Workshop on Distributed Systems: Operations & Management*, 2001.

[15] P. Bahl, P. Barham, R. Black, R. Chandra, M. Goldszmidt, R. Isaacs, S. Kandula, L. Li, J. MacCormick, D. Maltz, R. Mortier, M. Wawrzoniak, and M. Zhang. Discovering dependencies for network management. In *Proc. of the V HotNets Workshop*, 2006.

[16] ITU. *Specification and Description Language (SDL) (ITU-T Recommendation Z.100)*. International Telecommunications Union, August 2002.

[17] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin. Simple network management protocol (SNMP). RFC 1157 (Historic), May 1990.