

# Availability Monitoring

Fredrik Skarderud  
`fredrik@skarderud.net`

Ole Kasper Olsen  
`mail@olekasper.no`

Torkjel Søndrol  
`mail@torkjel.com`

Anders Wiehe  
`anders@wiehe.org`

Gjøvik University College / NISlab

20th November 2004

## **Abstract**

This report will look at availability related problems from a holistic point of view, i.e. we will look at how the availability is for the user, since a service which is running still can be unavailable for the user due to disturbing elements like spam, bad configurations, poor code and others. We will identify availability indicators, look at ways to get data used for measuring the indicators, look at ways to present the indicators and discuss several considerations one should have in mind before putting the metrics into action. We will last, but not least, describe a technical solution we implemented for gathering data and calculating some of the indicators defined.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Limitations</b>	<b>3</b>
<b>3</b>	<b>Availability Indicators</b>	<b>3</b>
3.1	General Reliability and Validity Considerations . . . . .	4
3.2	General Indicators . . . . .	4
3.2.1	Server Uptime . . . . .	4
3.2.2	Server Response Time . . . . .	6
3.2.3	Resistance Against DoS Attacks . . . . .	8
3.2.4	Password Length and Complexity . . . . .	8
3.3	Specific Web Indicators . . . . .	8
3.3.1	Retrieve a Web Site . . . . .	8
3.3.2	Web Site Accessibility . . . . .	10
3.3.3	Site Usability . . . . .	12
3.4	Specific E-Mail Indicators . . . . .	13
3.4.1	E-Mail Retrieval . . . . .	13
3.4.2	Spams Received . . . . .	14
3.5	Raw Data Harvesting . . . . .	15
<b>4</b>	<b>Implemented Availability Indicators</b>	<b>16</b>
<b>5</b>	<b>The Availability Metrics</b>	<b>17</b>
5.1	Introduction . . . . .	17
5.2	Calculation of the Metrics . . . . .	17
5.2.1	The Web Metric . . . . .	17
5.2.2	The Mail Account Metric . . . . .	18
5.3	Presentation of the Metrics . . . . .	18
5.4	Technical Solution and Implementation . . . . .	20
5.4.1	Data Gathering . . . . .	20
5.4.2	Result Display . . . . .	22
<b>6</b>	<b>Security &amp; Ethical Considerations of the Solution</b>	<b>23</b>
<b>7</b>	<b>Future Work</b>	<b>23</b>
<b>8</b>	<b>Conclusion</b>	<b>24</b>

## 1 Introduction

CIA<sup>1</sup>—the foundation of information security. Being one of the ground pillars of security, availability seems to be rather overlooked especially when it comes to figuring out a way to measure it properly.

First, a short definition of the term availability is in its place. In an information security context, we mean *availability of information*, not availability of a given server or host as is often used in computer science. The requirement of availability is that someone who are authorized to receive the information, will be able to get it in a timely manner whenever he or she requires it.

In other words, the classical computer science aspect of availability plays a large role in availability as seen from an information security point of view, but it is only part of the whole. A server may be running, yet an authorized user may not always be able to get his or hers data in a timely manner. For example, an e-mail server may be up even though it is misconfigured, thus leading to problems when the users try to download their e-mail. A different scenario may be that a Web site is too poorly coded for a person to be able to access it with his or her mobile phone, making the information completely inaccessible, even though the Web server is up and running.

In this report we will try to establish indicators which may be used for measuring the information security aspect availability. We will then take some of these indicators and combine them into two availability metrics. The idea here is that the indicators each represent a small part of the puzzle, and that they correctly combined give a good overall picture of availability from an information security point of view.

## 2 Limitations

There are an infinite number of aspects which influence the availability of different services. For example, not knowing how to turn on your own computer will severely affect the availability of any service requiring the use of a computer, but this is obviously out of scope for this report. Due to time constraints, we have mainly focused on indicators which are in some way connected to the server, within a server administrator's control, so that he or she may improve on the monitored service based on the measurements provided by the availability monitor. As a consequence of this, we have focused on indicators which are relatively easy to implement as measures.

## 3 Availability Indicators

Indicators for availability may be divided into separate categories. We have decided to separate indicators into *low level indicators*, *mid level indicators* and *high level indicators*. Low level indicators are typically indicators which can be physically measured by networking tools and the like, whereas higher level indicators are more based on user interaction, and the user's cognitive processes. Mid level indicators are indicators based on a somewhat more thorough check

---

<sup>1</sup>Confidentiality, Integrity and Availability

than the low level indicators. For example, that you are in fact able to access and use a given service rather than just checking if it's up or not.

As we are focusing on writing an availability monitor in software, many of the higher level indicators will be difficult to measure, but we acknowledge their importance in creating an information security related availability metric. We have therefore described several higher level indicators in this section, alongside low level indicators.

We have concentrated on two different systems where availability is important. The now ubiquitous World Wide Web is being used extensively for e-commerce and other important information exchanges and is still growing, and is therefore one of the systems we will attempt to monitor. The other system we have chosen is that of electronic mail, more specifically the POP3 service, which is the service responsible for serving the e-mails from the inbox to the user's e-mail client.

In this section we define both general indicators, and specific indicators for each system. The general indicators are indicators which can be applied to any system running a service one wish to monitor, and they are generally more low level than the more system specific indicators.

The indicators are summarised in a table format derived from suggestions by the US National Institute of Standards and Technology [1].

Some of the indicators in this section uses a scale ranging from 0 to 33, where 33 is the best score available and 0 is the lowest. The maximum score of 33 is something we have defined based on how our final availability metric is measured, as these indicators are a part of the metrics we have implemented.

### 3.1 General Reliability and Validity Considerations

Many indicators may very easily be tainted by load on both network and routers between the availability monitor and the server which is being monitored. However, as long as the monitor is running on the same network and uses the same route to the destination as the clients using the server, the monitor will give accurate results. If the availability monitor and the clients are running on different networks and using different routes to the server, the validity of some indicators will suffer greatly, as the increase in response time will probably only be evident for the availability monitor and not the client, or vice versa. It is in any cases important that the availability monitor is running on a server with good enough network bandwidth not to be a bottle neck.

### 3.2 General Indicators

#### 3.2.1 Server Uptime

In this case we define the term *uptime* as *amount of time a given server is up and running*.

Uptime of servers have long been a good indication of availability in system administrator circles. However, in the context of information security, uptime is not nearly as indicative of availability as one might think.

To illustrate why, consider a researcher trying to access some information via a given Uniform Resource Locator (URL). When the researcher tries to access the URL via his Web browser, he is given only an error message telling him

that access is “**Forbidden**”. What could conceivably have happened behind the scene in this case, could be that someone has been working on the server and accidentally altered the file permissions so that the Web server no longer has access to read the file. In this case, the server is up and running, and may have flawless uptime rates, yet information is not available when the researcher needs it. Thus, availability suffers from an information security point of view.

Another example may be the difference in server load during different time of the day. A server which has 95% uptime, but is down for a short period of time every day at 5PM when everyone at the workplace is about to check their day’s work of changes into a CVS tree, has extremely poor availability, even though the uptime seems to indicate differently. Similarly, if a server has 100% uptime, but occasionally extremely heavy loads, information may be totally or almost inaccessible due to extreme response times. Thus, availability is again poor. See section 3.2.2 for a response time indicator.

Performance Goal	Measure network and server uptime.
Performance Objective	Determine if the server is available over a longer period of time.
Metric	Percentage of uptime.
Purpose	To measure whether the server accepts a connection.
Implementation Evidence	Try to establish a TCP connection to the server on a service’s port.
Frequency	Once every 30 seconds.
Formula	$\frac{Uptime}{Total\ Time} \times 100$ <p>additionally we have</p> $\beta = \begin{cases} 1 & \text{if successfully connected} \\ 0 & \text{if unsuccessfully connected} \end{cases}$
Data source	Both successful and unsuccessful TCP connections.
Indicator Level	Low Level.

### Scale

This indicator will use a percentage scale going from 0 to 100 percent.

- A server that is up in 97 of 100 measurements, will have a 97% uptime.
- A server that is up in 10 of 100 measurements will have a 10% uptime.

As each poll is done with an interval of 30 seconds, mapping polls to uptime is elementary. If a measurement reports that the server is down, it is assumed to be down for the next 30 seconds. Equally, if a measurement reports that the server is up, it is assumed to be up for the next 30 seconds.

### Reliability and Validity

As shown above, the validity of this indicator is not extremely good. However,

uptime is a very important part of the availability of the system, and it is important to measure it in any system.

To improve the uptime measurement with regards to validity, we will implement measurement of this indicator by using TCP packets directed directly at a service's port rather than generic ICMP echo requests which are often blocked by a firewall between a potential client and the server running the service.

As for the reliability of this measurement, it is excellent. The method of determining whether or not the server is alive and listening on a certain port is reliable and robust. The result of the repeated polling can only be either up or down, so that random variations will be non-existent except when dealing with extremely heavy server loads, in which the uptime measurement may time out unreliably and randomly.

### **3.2.2 Server Response Time**

As with uptime, the response time of a server is a very important aspect of availability, yet it may suffer from the same shortcomings with regards to validity as the uptime measurements. However, as the response time of a server is a more granular measurement than uptime (which could be either one of two possible values), and therefore it is easier to give meaning to the measurement when measuring availability in an information security setting.

The connection between availability and a server's response time is that a server with high response times will lead to more waiting on the user's part, and thus limit availability as defined in section 1.

Performance Goal	Measure the response time of a service.
Performance Objective	Determine if the server's response time is within acceptable values from an availability point of view.
Metric	Response time measured in millisecond the resolution from the client makes the initial request until a connection is made.
Purpose	A faster server response will be more appreciated by the user. A quick response is indicative of a more available server.
Implementation Evidence	Try to establish a TCP connection to the server on a service's port and measure the time it takes to make the connection.
Frequency	Once every 30 seconds.
Formula	<p>Given <math>\delta = \text{response time in milliseconds}</math> and <math>\delta &gt; 0</math></p> $\alpha = \begin{cases} 0 & \text{for } \delta \geq 2000 \\ 33 - \frac{33\delta}{2000} & \text{for } \delta \in \langle 0, 2000 \rangle \end{cases}$
Data source	Successful connections.
Indicator Level	Low Level.

### Scale

This indicator is measured on a ratio scale ranging from 0 to 33. In the formula for this indicator we define a score  $\alpha$  to indicate how good the response time  $\delta$  is from a server. The higher score received, the better is the response time from the server. The maximum response time we handle in this indicator is 2000 ms. If the response time rises beyond 2000 ms, we declare the server dead or not responding, giving the score of 0 from this indicator.

- A server with 10ms response time during a measurement will get a score of 32.835.
- A server with 1500ms response time during a measurement will get a score of 8.25.
- A server with 2500ms response time during a measurement will get a score of 0.

### Reliability and Validity

This is a good indicator for measuring the response time of a server, hence the validity is good. There are a lot of factors affecting the response time, like the route, server and router traffic loads between the client and the server and so on. Therefore the reliability is not so good, but the average of many measurements is expected to give a more accurate view of the server's response time.

### 3.2.3 Resistance Against DoS Attacks

If an attacker wants to limit or completely nullify the availability of a given resource on the Internet, a *Denial of Service* (DoS) attack is often the weapon of choice as attacks are often easily mounted (several tools exist [2]) and hard to defend against [3], although a well-configured firewall may stop some attempts. A DoS attack will, if successful, exhaust the bandwidth of the server, and thereby deny service for legitimate users.

Measuring a service's resistance against such attacks is virtually impossible without actually performing such an attack, thus using this as an indicator on unknowing sites is out of the question. Other ways of measuring resistance against DoS attacks may be by measuring a firewall's efficiency regarding filtering and throwing away unwanted network packets, the bandwidth of the service and a lot of other indicators. None of these will give a thorough picture of the situation, although they may—in certain situations—be adequate.

### 3.2.4 Password Length and Complexity

The most common user authentication schemes on the Internet are implemented with passwords. Often, this may create an availability problem, as people may forget their passwords, and thus be denied access to information they usually are authorised to access. A weaker password is easier to remember, and gives a higher degree of availability, but has poor level of security, while a stronger password is harder to remember, and gives a lower degree of availability, but a higher level of security.

Evidently, a trade-off between availability and the other fundamental parts of information security—confidentiality and integrity—have to be made. Should users be forced to use a password which matches a special template (e.g., minimum 8 characters in which uppercase letters, lowercase letters and numbers each must appear at least once), or should the users be able to decide for themselves?

Measuring how good a person is at remembering passwords is difficult, as different persons have different capabilities of remembering. Therefore we cannot simply calculate a score based on the password's structure. Instead we should check how often a person have requested a lost password. This number can be seen in proportion to the total number of times the password was used, so that a score can be calculated of how password protection affects the availability of a given service.

## 3.3 Specific Web Indicators

### 3.3.1 Retrieve a Web Site

Even though indicators such as uptime (section 3.2.1) and response time (section 3.2.2) seems to be enough to paint a good picture of a Web server's availability, they do not tell the entire story. This is especially the case with dynamically created Web sites which rely on database backends and server side scripts.

Much previous work has been made in discovering how the download speed of a Web page affects the economy and the Quality of Service (QoS) of the Web service. Zona Research, Inc. performed a research in 1999 [4] on how the economy of e-commerce was affected by download time. They claim that as much as \$4.35 billion in e-commerce sales is lost each year in the US due to

unacceptable download speeds. An experiment performed by HP Labs in 2000 [5] found that 6-8 seconds was the average maximum latency accepted by the participants when using e-commerce. These two reports tell us that the latency of a Web page has a major affect on its availability. There is a rule saying that “*Any homepage which need more than eight seconds to download is unacceptable*”<sup>2</sup>. A user has little patience when waiting for a Web page to load, and will in most situations abort the download process—even though the service is available—if the download time is much longer than eight seconds.

Performance Goal	Download a page of a Web site.
Performance Objective	Determine if the Web site’s data rate is within acceptable rates regarding availability.
Metric	Downloaded data measured in kilobytes (1024 bytes, KiB) per second.
Purpose	A faster server response will be more appreciated by the user. Thus, a quick response is indicative of a more available server.
Implementation Evidence	Try to establish a connection to the Web server and measure the time it takes to download the front page.
Frequency	Once every 2 hours.
Formula	<p>Given <math>\gamma = \frac{\text{fetch size KiB}}{\text{fetch time s}}</math></p> $\alpha = \begin{cases} 0 & \text{for } \gamma \leq 1 \\ \frac{33\gamma}{128} & \text{for } \gamma \in \langle 1, 128 \rangle \\ 33 & \text{for } \gamma \geq 128 \end{cases}$
Data source	Successful downloads.
Indicator Level	Medium Level.

### Scale

This indicator uses a ratio scale ranging from 0 to 33 calculated from the time it takes to download a web page and the size of the page. In the formula for this indicator we define a score  $\alpha$  to indicate how good the data rate  $\gamma$  is from a server. The higher score received, the better is the data rate is from the server. We consider a data rate of 128 KiB per second or above to be very good with regards to availability (there will be very few or no noticable delays), and any data rate from and above 128 KiB per second is awarded maximum score (33). Data rates from 1 KiB per second and less, is considered unusable. The user is likely to abort the loading of the page and try to find information elsewhere.

- If the measurement uses 1 second to download 100 KiB, it will get a score of 25.78.

<sup>2</sup>Eight Second Rule: <http://xarch.tu-graz.ac.at/autocad/wiki/EightSecondRule>

- If the measurement uses 2 seconds to download 50 KiB, it will get a score of 6.45.
- If the measurement uses 20 seconds to download 10 KiB, it will get a score of 0.

### Reliability and Validity

This indicator will measure the time it takes to connect to a Web site and retrieve one predefined page, including all its pictures (both on-site and off-site). The result will be calculated based on the cumulative size of the page with images and the time it took to download it. The fact that we only download one particular page of a site will impact the validity of the indicator, however by specifying the downloaded page manually, we may limit the negative impact on validity.

The measurements made using this indicator will be reproducible over time. They may vary somewhat during the time of day as the server's load will increase and decrease based on the visitors' usage patterns.

It is to be noted that this measurement is unreliable for documents less than roughly 1KiB, as the overhead for establishing a connection to the server would take much time in relation to the time it takes to download the data.

### 3.3.2 Web Site Accessibility

When Tim Berners-Lee first invented the World Wide Web in 1990, his vision was an interconnected web of information, marked up by semantic guidelines and accessible by a wide array of devices (known as user agents) [6]. However, with the advent of mainstream graphical user agents such as NCSA's Mosaic, Netscape's Navigator and their myriad of descendants (including Microsoft Internet Explorer, Opera, Safari and the various browsers based on Mozilla), accessibility and semantic markup went out the window as these new and fancy user agents introduced HTML tags which were designed to change only the visual representation of the information without carrying any semantic meaning.

In later years, the focus on a more semantic Web has increased and the World Wide Web Consortium (W3C) is on the right track with new XHTML coding standards [7, 8] which aims to strictly separate style from content, leaving the styling to Cascading Style Sheets (CSS) [9, 10]. Developing sites by the standards is integral in creating and maintaining an accessible Web site. In addition to strict coding standards which separates style from structure and information, the W3C have also released guidelines for Web content accessibility [11] through the Web Accessibility Initiative<sup>3</sup> (WAI). These guidelines give advice to Web developers which expand on the accessibility guidelines already given in the coding standards to help create a more accessible Web, making the Web available for everyone—with or without disabilities or handicaps, and with any user agent.

This means that sites coded by the standards and complying with the advice given from WAI have much greater chance of being available to people using limited and downscaled visual user agents. Such user agents may be found on devices such as mobile phones and PDAs which have a very limited visual space and also equally limited in their CPU capacities.

---

<sup>3</sup><http://www.w3.org/WAI/>

The following table summarises the indicator.

Performance Goal	Measure the accessibility of a Web site.
Performance Objective	Determine if the Web site is available by measuring it's compliance with current standards and guidelines.
Metric	Compliances with the WCAG AAA, AA and A guidelines, compliance with reported document type (HTML, XHTML, etc.).
Purpose	An available Web site need to be accessible to the widest possible audience.
Implementation Evidence	Acknowledged Web services such as Bobby and the W3C Validator will be used to measure a Web site's compliance with standards and guidelines.
Frequency	Once every 2 hours.
Formula	<p>Given <math>\omega = \text{specified document type}</math> and <math>\eta = \text{number of validation errors}</math></p> $\lambda = \begin{cases} 15 & \text{if } \omega = \text{XHTML Strict} \\ 12 & \text{if } \omega = \text{HTML Strict} \\ 9 & \text{if } \omega = \text{XHTML Transitional} \\ 6 & \text{if } \omega = \text{HTML Transitional} \\ 5 & \text{if } \omega = \text{Frameset} \\ 0 & \text{if } \omega = \emptyset \text{ or unknown} \end{cases}$ <p>and given that <math>\{\mu, \nu, \xi\} = \sum\{A, AA, AAA\}</math> errors and <math>\epsilon = 10\mu + 5\nu + 3\xi</math> we have that</p> $\alpha = \frac{\lambda}{\eta + 1} + 18 - \begin{cases} \frac{\epsilon}{18} & \text{for } \epsilon \leq 18^2 \\ 18 & \text{for } \epsilon > 18^2 \end{cases}$
Data source	The W3C Validator and Bobby.
Indicator Level	High Level.

### Scale

A ratio scale ranging from 0 to 33 is used also here, based on how well a web page validates using different validation standards. A document's compliance with standards and the standard chosen is given a maximum of 15 points. The remaining 18 points are awarded to sites which comply with the accessibility guidelines [11].

- A web page that is XHTML strict with zero W3C validator errors and 5 AAA errors will get a score of 32.167.
- A web page that is HTML transitional with 10 W3C validator errors, 1 A, 2 AA and 4 AAA errors will get a score of 17.767.

- A web page with unknown doctype with 42 W3C validator errors, 6 A, 38 AA and 9 AAA errors will get a score of 0.

### **Reliability and Validity**

As accessibility and availability is so closely related, the validity of this indicator is very good. As measurements of compliance to standards and guidelines would have to be implemented in software, the results would be reproducible as long as the content of the site doesn't change. Hence, the reliability is good in that regard, even though using different software may easily provide a different result.

### **3.3.3 Site Usability**

A cluttered and poorly designed Web site will be a significant hindrance with regards to the availability of the site in question. There are many things which can cause poor site usability, but most of them can be summarised by the fact that users will spend a considerable amount of time trying to find information, rather than absorbing it.

Here are some examples of common usability mistakes:

**Poor information structuring:** A Web site with much information should have well thought out menu structures, where width and depth of menu hierarchies need special considerations, so that the structure is intuitive to most people.

**Excessive use of Macromedia Flash or similar technologies:** Some Web sites are built totally in Macromedia Flash, a vector animation format. While this makes for some fancy sites, usability usually takes the back seat to fancy graphics and non-intuitive, complicated menus. Also, the Web browser's built-in functionality like back and forward navigation, bookmarking and more cannot be applied to the site.

**Excessive use of frames:** Sites using the age-old technology known as "frames", suffer from many of the usability drawbacks as sites which overuse Flash or Flash-like technologies. For example, much of a browser's built-in functionality such as bookmarking will be rendered unusable.

**Poor colour contrasts:** Some sites use very poor contrast between background colours and colours on the text. The result is poor usability, especially for visually impaired persons.

**Garish text effects:** Many sites use blinking and scrolling texts. Such effects severely hamper the usability of the site for a lot of people, as they are extremely disturbing while reading text on other parts of the site. Also, with blinking text, you rarely have time to read the entire text before it vanishes.

**Unclear mission statement:** Sites often lack any sort of introductory text, which new users can read to ascertain the usefulness of the site. As a consequence, people either waste time looking for information on other sites because they didn't think the site was relevant, or they waste time looking for information which isn't on the site at all.

There are numerous other usability problems regarding Web sites today. Usability “guru” Jakob Nielsen lists many in his book [12], which we’ll refer the reader to for more information regarding Web site usability.

### 3.4 Specific E-Mail Indicators

#### 3.4.1 E-Mail Retrieval

One good way of measuring the availability of a e-mail service is by trying to connect to the e-mail server and download a user’s mail from the server. This way, we not only determine whether or not the server is up and running, but also perform a holistic measurement of whether or not it is possible for a user to retrieve his mail.

Performance Goal	Download e-mail from an e-mail server.
Performance Objective	Determine if the e-mail server’s data rate is within acceptable limits regarding availability.
Metric	Downloaded data measured in kilobytes (1024 bytes, KiB) per second.
Purpose	A faster server response will be more appreciated by the user. Thus, a quick response is indicative of an available server.
Implementation Evidence	Try to establish a connection to the e-mail server and measure the time it takes to download the mail currently in the inbox.
Frequency	Once every second minute.
Formula	<p>Given <math>\gamma = \frac{\text{fetch size } KiB}{\text{fetch time } s}</math></p> $\alpha = \begin{cases} 0 & \text{for } \gamma \leq 1 \\ \frac{33\gamma}{64} & \text{for } \gamma \in (1, 64) \\ 33 & \text{for } \gamma \geq 64 \end{cases}$
Data source	Successful e-mail downloads.
Indicator Level	Medium Level.

#### Scale

This indicator uses a ratio scale going from 0 to 33 calculated from from the time it takes to connect to a mail server and download all the e-mails from the server, and the size of the mails transfered. For everyday use, with few extremely large e-mail attachments (of 1MiB and above), a data rate of 64 KiB per second is sufficient for transparent use of e-mail services for most persons. Thus, a data rate of 64 KiB per second will be awarded 33 points.

- A measurement that uses 1 second to download 5 KiB of e-mail will get a score of 2.57.

- A measurement that uses 2 seconds to download 100 KiB of e-mail will get a score of 25.78.

### Reliability and Validity

The retrieval of e-mail is closely related to the service availability indicator described in section 3.3.1, but they are separated in the report, since it was convenient to group the indicators by web and mail. Another reason for separating the two indicators is that the mail has a higher degree of validity than the availability of Web. For instance, the availability of e-mail is not affected by which mail client used to retrieve the mail, and it detects the cases when the mail server is up and running, but not within reach for the user. A factor that may affect the validity of this indicator is that we are measuring the connection time and the download time, but this will also include the overhead while connecting and downloading messages. In other words; fewer messages to download from the mail server will be more affected by overhead and have a lower degree of validity than when there are more data to download.

Since mail stored on a mail-server under normal circumstances doesn't change its content, this indicator is reproducible, and the reliability is good.

### 3.4.2 Spams Received

Spam is the single largest source of annoyance on the Internet today. A report from the independent Nucleus Research [13] shows that average spam cost per employee is \$1934 each year, and that average lost productivity per employee is 3.1% among Fortune 500 companies.

We define spam as unwanted e-mail received to the mail account in question. This can be virus mails, advertisement mails, scams etc. It is the mail account holder who must determine what spam-rate is tolerant for his account, since what mail is unwanted differs from one person to another.

This indicator uses the spam filter *SpamAssassin*<sup>4</sup> to detect how much spam received on a mail account. This is a heuristic spam filter that also uses a list of pre-defined tests<sup>5</sup> to determine what's spam and what's not.

---

<sup>4</sup><http://spamassassin.apache.org>

<sup>5</sup><http://spamassassin.apache.org/tests.3.0.x.html>

Performance Goal	Measure email accessibility.
Performance Objective	Measure amount of spam received on an e-mail account.
Metric	Number of spam received compared to genuine e-mail.
Purpose	The purpose is to measure the accessibility on an email account based on the spam received.
Implementation Evidence	Using <i>SpamAssassin</i> to detect the amount of spam received to a e-mail account.
Frequency	Once every 5 minutes.
Formula	<p>Given <math>v = \text{number of spam mails}</math> and <math>\sigma = \text{total amount of mails}</math></p> $\alpha = 33 - \frac{33v}{\sigma}$
Data source	E-mails received to a specific e-mail account.
Indicator Level	High Level.

### Scale

This indicator uses a ratio scale going from 0 to 33, calculated by how much of the mail in a mail box that is spam mail.

- A measurement detecting that 90 of 100 mails on a mail server is spam will get a score of 3.3.
- A measurement detecting that 5 of 100 mails on a mail server is spam will get a score of 31.35.

### Reliability and Validity

The reliability and validity of this metric could be very good, but it depends on that neither the user nor the mail server is equipped with a spam filter. It also depends on the spam filter used by the indicator to detect spam. The best spam filters on the market today uses *heuristic* and learning methods (such as Bayesian filters) along with a large array of preset filters to combat spam. The reliability and validity of the indicator highly depends on the spam filter's success at identifying spam (false positives and false negatives included).

## 3.5 Raw Data Harvesting

Creating a good metric for availability is difficult, both from a data gathering point of view and from a calculation point of view.

Some types of raw data are easy to harvest, most particularly aspects of availability which are often related to classical system administration, like checking if a server is alive or not and similar networking checks. We aim to enhance these basic measures in such a way that they are more adapted towards the information security aspect of availability. For example, rather than merely checking if

a machine is up and running or not, we try to create a TCP connection directly to the running service (i.e., the service's port) we are interested in monitoring.

Additionally we have also defined some higher level indicators which can easily be measured, like the accessibility of a Web site (section 3.3.2) and the actual operational status of a running service (sections 3.3.1 and 3.4.1).

Data gathering for other higher level availability indicators, such as measuring how difficult a site is to navigate for a user (section 3.3.3) is difficult to obtain. Software based solutions may be possible to implement, like for example an always-visible button which the user can click if he or she is lost. We consider this approach to be less than optimal, as it is difficult, or even infeasible to draw any conclusions from the results without doing an experiment in a heavily controlled environment. Another solution may be to parse the Web server log files looking at how many requests a user had to make before he or she gained access to a specific resource on the server, and also the time which was spent. However, given the stateless nature of the HTTP protocol and the limitations on data to log, the information which is derived from such an indicator is limited, unless you perform an experiment where the subjects are told to try to find a specific resource in the least amount of time and with the least amount of clicks. Combined with a survey afterwards, such an experiment is probably the best solution regarding obtaining such data. However, as we are focusing on indicators which can be applied generally on many different servers and configurations, conducting such surveys and experiments is unfortunately infeasible, yet we acknowledge it as a very important aspect of measuring the availability of the service.

There are also other indicators which are difficult to measure, even though not always from an implementation point of view. E.g., measuring a service's resistance towards a (Distributed) DoS attack is not too intricate to implement, but from an ethical and legal point of view such measurements will be difficult to carry out.

With some indicators we had to make some compromises. Polling a site for download speed once every two hours is too seldom to be able to properly catch general tendencies (like an increase in load each evening at 5PM, etc). However, we didn't want to put too much strain on both the servers in question, and our system (thus creating skewed results).

## 4 Implemented Availability Indicators

Here we have implemented three different indicators or measures to give a low level availability presentation. The measures displayed are uptime, average downtime per day, and response time. These are not representative as a holistic information security view of availability, but nevertheless interesting measurements. As we already have the data, we have implemented some presentations of them.

### Uptime

This graph is based on TCP requests sent to the server in question every 30 seconds in the chosen time interval. The logic behind the graph is as simple as if we get a reply from the server we say that the server is up, and if the TCP request times out (after 2000ms), we say that the server is down. Based on

those results, the graph has the value 1 if the server gives a reply, and the value 0 if the TCP request times out. In the top right corner of the graph image, the total downtime during the chosen time interval is displayed as a percentage of the total time displayed in the graph.

#### **Average downtime per day**

Another interesting way to look at the TCP request data is on a daily basis. This graph displays the time interval along the x-axis, and percentage downtime along the y-axis. Again we find the average downtime for the period in the top right corner of the graph image.

#### **Response time**

The third representation method of low level availability in this graphical presentation is to make use of the response time from the TCP request. The theory is as follows: The faster a TCP request reply is from a server, the better the availability of the server is. A slow server increases the time spent for a client communicating with it. A server's response time is displayed in millisecond along the y-axis, and the time interval is still along the x-axis. Among things that might be read from this graph is that when the response time is high, it might be much traffic to the server. One aspect to be considered using this measure is that servers in question which has a shorter route to the server running the data gathering scripts, should have a shorter response time than servers with a long route.

## **5 The Availability Metrics**

### **5.1 Introduction**

Based on the indicators for availability defined in section 3, we have chosen some of them to form part of a metric for measuring the availability of a Web site, and a metric for measuring the availability of a mail account.

We acknowledge and realise that these metrics are far from complete, however given the time available to us, we have tried to pick some indicators from different levels, and combined them into two metrics, which we have implemented in an availability monitor prototype. The monitor harvests raw data from some pre-chosen hosts and mail accounts, and presents the data graphically on a Web page.

### **5.2 Calculation of the Metrics**

The two metrics implemented in the availability monitor are mostly based on the same indicators, but there are differences, specially the mid and high level indicators used. The metrics are both based on repeated polling of the indicators, thus one will be able to see how the availability of the service changes over time.

#### **5.2.1 The Web Metric**

The complete web availability metric is calculated based on the different indicators defined earlier in this report. Low level, mid level and high level indicators

are used and combined to calculate the complete availability metric. The metric is given a scale ranging from 0 to 99, where 0 is the lowest score possible, and 99 the highest. The score is calculated based on the scores received from the low, mid and high availability indicators. Each level of indicators can give the maximum score of 33 adding up to a maximum of 99.

Indicators implemented in this metric are the follows:

**Low level indicators:** Server Uptime (section 3.2.1), Server Response time (section 3.2.2).

**Mid level indicator:** Retrieve a Web Site (section 3.3.1)

**High level indicator:** Web site Accessibility (section 3.3.2)

Combining the different indicators, the formula for the metric is then:

$$\phi = (\alpha_{low} + \alpha_{mid} + \alpha_{high}) \times \beta$$

where  $\alpha_{[level]}$  is defined by the individual indicators and  $\beta$  is given by the server uptime indicator. Note that if  $\beta = 0$ , the entire metric returns the score of 0, because the other measures or indicators would not be possible to perform. The availability of the service in question is then finally ranked based on the score achieved from the metric.

### 5.2.2 The Mail Account Metric

The mail account availability metric is based on the same low level indicators as the web metric. However the mid and high level indicators are different:

**Mid level indicator:** E-mail Retrieval (section 3.4.1)

**High level indicator:** Spams Received (section 3.4.2)

The formula for the metric is the same as for the web availability metric, as is the scale for the metric, and the scores received from the low, mid and high level indicators.

## 5.3 Presentation of the Metrics

We have chosen to display the metrics using three graphs, each of the graphs representing the low, mid or high level indicator.

The resulting graph is the cumulative result of the individual levels of measurements. Each level's value (ranging from 0 to 33) is represented by its own graph where low level measurements are indicated by a red graph, mid level by an yellow graph and high level by a green graph. The total availability of a system is indicated by the top-most graph as each level is added to the previous, giving a minimum availability value of 0 and a maximum value of 99. On the x-axis we find the time line, and on the y-axis we find the scale of the metrics.

When it comes down to analyse the total availability for the service in question, the total availability graph is used. On the y-axis three smilies are plotted to illustrate the users satisfaction. A sad smiley is to be found where the y-axis value is low. A semi-happy smiley is plotted where the y-axis value is in the mid range of the scale, and a happy smiley is found on the top end of the scale.

Figure 1: Total Availability graph - web

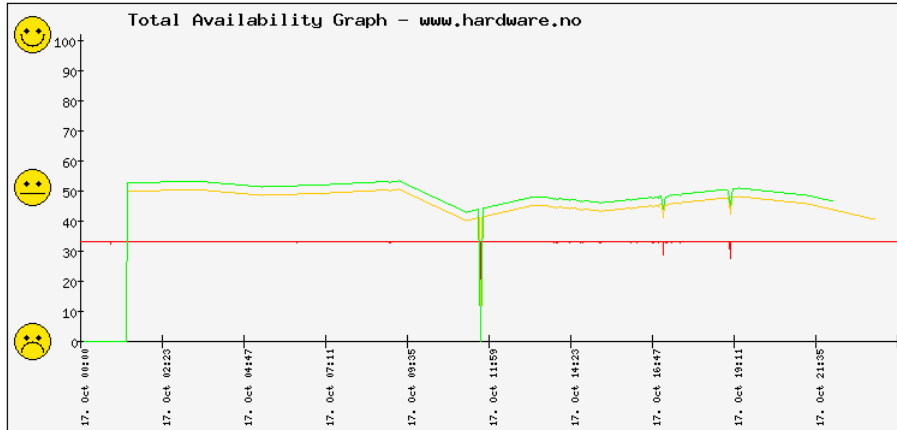
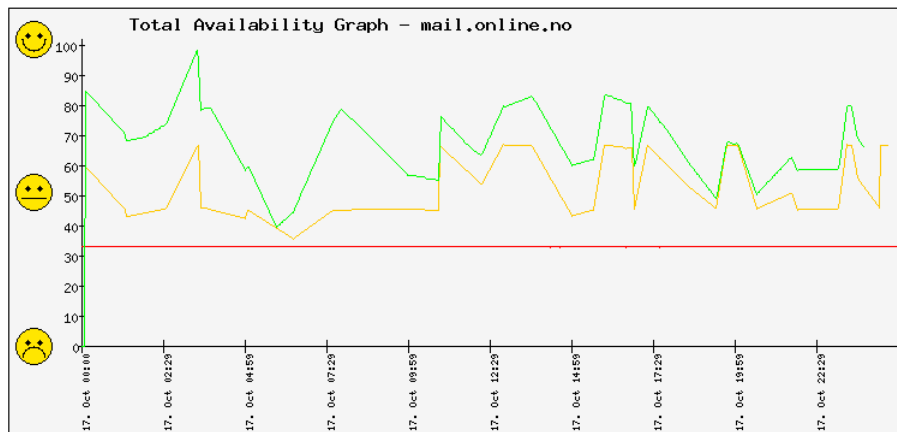


Figure 2: Total Availability graph - mail



The higher value the top graph (the green one) gets the better availability score the service in question gets. A good score gives a happy user.

Another way to look at the results displayed in the total availability graph is study the gap between the different coloured graphs. For example if the gap between the red and the yellow graph is small, it indicates that the service in question has poor availability or scores low regarding to the mid level indicator. If there is a big gap between the yellow and the green graph, it indicates that the service in question scores high regarding to the high level indicator. Figure 1 is a screen shot displaying the total Web availability metric for the domain **www.hardware.no**. Conclusions drawn from the graph are that the server has a very good and stable response time, decent download speed and low accessibility.

Figure 2 shows the total e-mail availability graph for a specific mail account on the mailservers **mail.online.no**. The yellow and green graph varies much in value. The yellow graph will vary depending on how much mails received on a

given time and how fast they are received, and the green graphs indicates that the spam ratio is sometimes high (low value) and the spam ratio is sometimes low (resulting in a high value).

## 5.4 Technical Solution and Implementation

Implementation-wise, the availability monitor consists of two modules. One data gathering module, and one result displaying module.

### 5.4.1 Data Gathering

Due to its great flexibility and availability of many pre-made networking packages, Perl was chosen to do the grunt work of gathering raw data from various sources, based on some of the indicators we have defined. The raw data are stored in a MySQL database for easy access by other applications.

The sampling of the data was implemented in such a way that each server was sampled at the correct interval, independent of how long time the previous server's sample took. Listing 1 shows an example of the general algorithm which is being used in the availability monitor.

Listing 1: Polling Example (5 second interval)

```
my %hosts = %{MetricsDB::getHosts(MONITOR.ID)};

#main loop
my $i = 0;
my $j = 3600*2; # do a download once every two hours
my $interval = 5; # wait 5 seconds between each host
while(1) {
    foreach my $h (keys %hosts) {
        my $starttime = gettimeofday();
        doIndexGrab($h, $hosts{$h}{"subp"}, $hosts{$h}{"prot"});
        my $endtime = gettimeofday();
        sleep($interval);
        $i += ($interval + sprintf("%.0f", $endtime-$starttime));
    }
    sleep(($j-$i < 0) ? 0 : $j-$i); $i = 0;
}
```

The Perl part of the availability monitor is largely based on efficient, publicly available packages. For example, for retrieving data from Web servers, the *libwww-perl*<sup>6</sup> (LWP) library was used. Listing 2 shows how we use the LWP user agent package in a generic subroutine for downloading Web content. The routine returns the number of bytes downloaded.

Listing 2: LWP Example (code excerpt)

```
use LWP::UserAgent;

# setting up the LWP user agent
my $ua = LWP::UserAgent->new();
# UA string
$ua->agent("NISlab_IndexGrabber/0.00001-ole.olsen\@hig.no");
```

<sup>6</sup><http://lwp.linpro.no/lwp/>

```

my $fetch_size = fetchslave("http://www.hig.no/image.png");

# subroutine for fetching data from URLs
sub fetchslave {
    my $url = shift;
    my $referrer = shift || undef;

    my $req = HTTP::Request->new("GET" => $url);
    $req->header("Referer" => $referrer) if defined $referrer;

    my $res = $ua->request($req);

    return $res->is_success ? length($res->content) : 0;
}

```

In section 3.3.2 we describe an indicator regarding the accessibility of a Web site based on standards and guideline compliance. Measuring compliance with standards and guidelines using computer software is extremely difficult. A check of many of the WCAG guidelines [11] are in fact impossible to implement in software, as they are based on the contextual meaning of text (actual content, captions, alternate texts for images, etc.) and other structural elements (tables, images, etc.) in Web pages. These are things only a human being will be able to discern. However, there are services available on the Web which are able to check for compliance fairly well.

The authors of the (X)HTML standards have an excellent validation service<sup>7</sup>. The service is able to check a site according to the document type the site is using.

For checking the WAI WCAG guidelines, Bobby<sup>8</sup> is an excellent resource, which we have utilised when implementing our accessibility indicator.

As both the W3C Validator and Bobby are Web based services, we have used the LWP user agent package as introduced in listing 2 when querying them. Listing 3 shows how we query the W3C Validator and parse the results.

Listing 3: Querying and Parsing

```

my $url_to_validate = "http://www.hig.no";
# URL encode the URL:
$url_to_validate =~
    s/([A-Za-z0-9])/sprintf("%%%02X", ord($1))/seg;

my $req = HTTP::Request->new("GET" =>
    "http://validator.w3.org/check?uri=$url_to_validate");

my $res = $ua->request($req);
if($res->is_success) {
    my $data = $res->content;
    my $host_id = MetricsDB::getHost($host);

    if($data =~ /This Page Is( <strong>NOT</strong>)?
        Valid <a href=" [^\"]*" >(.*)</a> (\w+)!/i) {

```

<sup>7</sup><http://validator.w3.org/>

<sup>8</sup><http://bobby.watchfire.net/>

```

my $valid = (defined $1) ? 0 : 1;
my $doctype = $2."_".$3;
if ($valid) {
    # store doctype and zero errors
    MetricsDB::doPreparedStatement($sth, $host_id,
                                   $doctype, 0);
}
else {
    # store attempted doctype and number of errors
    $data =~ /<th>Errors: <\th>\s+<td>(\d+)<\td>/;
    MetricsDB::doPreparedStatement($sth, $host_id,
                                   $doctype, $1);
}
}
}

```

#### 5.4.2 Result Display

The results are displayed on the web using XHTML 1.0 and PNG<sup>9</sup>. The reason for this is because the Web is platform independent, thus the results can be viewed from almost any computer or mobile device.

PHP<sup>10</sup>'s ease of use for real time image creation using the GD library<sup>11</sup> made it the natural choice for calculating and displaying the results of the data gathering.

We have made a Web page where the user can specify the results she wants to see. Options include metric selection, host selection and a time period to show results from. When the user has filled out the form and clicked on the "plot graph" button, the Web page is reloaded with a link to the image creation script along with the user's selections.

The image creation script makes a PNG image which is directly sent to the user's browser. The image is constructed from the user's input data. Each metric has three functions for calculating points for the low, mid and high level. The three levels are described in detail in section 5.2. In addition to metric specific level calculation functions, there's also generic graph plot functions. These functions plots the axes, text to describe axes intervals and the values returned by the metric calculation functions.

Listing 4 shows an example of a graph plot function, the low level plot. The results of the low level calculations are stored in the lowarray. Every result is plotted and a line is drawn from the last plotted line.

Listing 4: Graph plot example

```

/* Low level plot */
$lp_y = 0;
foreach ($lowarray as $key => $line) {
    $axis_point_x = $axis_start_x +
        floor((mktime(ltrim(substr($key, 8, 2), "0"),
                    ltrim(substr($key, 10, 2), "0"),
                    ltrim(substr($key, 12, 2), "0"),

```

<sup>9</sup>Portable Network Graphics

<sup>10</sup><http://www.php.net>

<sup>11</sup><http://www.boutell.com/gd/>

```

        ltrim(substr($key, 4, 2), "0"),
        ltrim(substr($key, 6, 2), "0"),
        substr($key, 0, 4)
    ) - $startdate) * $pixelsbetweentopoints);
$axis_point_y = $axis_stop_y -
    floor($line * $ypixelsbetweentopoints);

if($lp_y == 0) {
    $lp_y = $axis_point_y;
    $lp_x = $axis_point_x;
}

imageline($im, (($lp_x == -1) ? $axis_point_x : $lp_x),
    (($lp_y == -1) ? $axis_point_y : $lp_y),
    $axis_point_x, $axis_point_y, $graph_color);
$lp_x = $axis_point_x;
$lp_y = $axis_point_y;
}

```

## 6 Security & Ethical Considerations of the Solution

There are some considerations one must have in mind before starting an availability survey. First of all, measuring the uptime of others servers may not be in the best interest of the servers' owners. It may give results they won't agree with as they may give the business a bad reputation, or the measuring itself may generate a heavy traffic load on the current server if it is done too frequently.

Another aspect we must consider is what sort of data we store during an availability survey. Personal information (i.e. information that can directly or indirectly be linked to a physical person) can in most cases not be stored without the person's approval. Therefore it may be unethical or even illegal to store information such as IP addresses, MAC addresses and e-mail addresses without the person's knowledge. Storing such personal information would also place requirements on the security measures on the database [14] to properly protect the data.

We should be careful with how we presents the metrics data. Graphs and other visual and numerical statistical information can easily be wrongly treated to give a false impression of high or low availability. Also using technical terms the reader of the data is unfamiliar with may give him or her a wrong impression of the data.

## 7 Future Work

We do not claim our solution to be perfect or complete. Measuring the availability of a service from a holistic information security point of view is an extremely complex undertaking.

Technically, the most obvious improvements can be made to the part of the availability monitor which collects data for the Web site retrieval indicator (section 3.3.1). Currently, the time measure is a bit coarse, as it starts the

timer when the connection is being made, not when the data actually starts going over the network. Thus, the overhead of connecting and disconnecting to the server is included in the measure. As pointed out earlier, this skews the results when the site in question is very small in size. This also applies to the e-mail retrieval indicator (section 3.4.1) where a lot of time is used to establish the connection to the e-mail server, provide log in credentials, and then log out after the e-mail is retrieved. Additionally, it can be further improved with the ability to download and parse CSS (which again may link to images), JavaScripts and Java applets and also to recursively parse frame sets. Currently it only measures the time it takes to download a page and all images linked into that page with the traditional `img` HTML tag.

The mail metric can be extended to also check whether a mailbox can receive mail from various sources. IMAP, an alternative mail fetching protocol, can also easily be implemented into the already implemented mail metric.

A file server availability metric can be implemented relatively easily into the existing system. Indicators may include the availability of files and transfer speed both reading and writing.

Our implementation deals with *relatively* low level indicators. An interesting extension to our work would be to include some very high level indicators which may include doing some extensive testing and experimenting. For example a high level advanced web system availability indicator can be made. For example a system for measuring Class Fronter's<sup>12</sup> availability. Indicators may include logging in, entering rooms and downloading documents.

As mentioned earlier in the report, there is an almost infinite amount of indicators. Some can be easily implemented, others not.

With regards to our scales and the weighting of results, more research and experimentation needs to be conducted to properly calibrate the results to get meaningful results for a wide array of different sites and services. For example, is a data rate of 128 KiB per second from a site enough to be awarded maximum score? Perhaps it should be set higher or lower?

Further, each of our groups of low, mid and high indicators are weighted equally when calculating the metrics. Some of the indicator groups should perhaps be weighted less or more.

There can also be more effective ways to display the availability results.

## 8 Conclusion

This project have given us valuable insight in the world of availability. We have focused on measuring holistic availability from a information security point of view, and have described indicators which can be used to measure these availability aspects. Further, we have also merged some of the indicators into two metrics used to monitor the availability of web and mail servers. Last, but not least, we have created software that uses these metrics to monitor the availability of several chosen Web and mail servers, and we have created a Web interface for calculating and displaying the results from the gathered data.

Due to the limited time we had available for this project the availability monitor is somewhat limited in functionality, but we think this could be improved

---

<sup>12</sup>Virtual learning environments used by Norwegian colleges, universities and the like. The URL is <http://www.fronter.com/>.

to create a powerful tool for availability monitoring of different services.

The prototype of the availability monitor can be found on the intranet of Gjøvik University College at <http://128.39.80.251/>. Measurement data is available from October 1st. It is to be noted, however, that the development of the availability monitor software has been an evolving process and the ways data have been collected may not have been static throughout the measured period.

Complete source codes for the availability monitor is available at [http://128.39.80.251/avail\\_mon-prot](http://128.39.80.251/avail_mon-prot)

## References

- [1] Marianne Swanson et al. Security Metrics Guide for Information Technology Systems. *NIST 800-55*, 2003.
- [2] Felix Lau, Stuart H. Rubin, Michael H. Smith, and Ljiljana Trajovic. Distributed denial of service attacks. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2275–2280, Nashville, TN, USA, October 2000.
- [3] Mixer. Protecting against the unknown, January 2000. Specification, <http://packetstormsecurity.org/papers/contest/Mixer.doc>.
- [4] The economic impacts of unacceptable web-site download speeds. Zona Research, Inc., April 1999. Specification, [www.webperf.net/info/wp\\_downloadspeed.pdf](http://www.webperf.net/info/wp_downloadspeed.pdf).
- [5] Anna et al. Bouch. Quality in the eye of the beholder: Meeting users' requirements for internet quality of service, April 2000. Specification, <http://www.hp1.hp.com/techreports/2000/HPL-2000-4.pdf>.
- [6] Tim Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco, 1999.
- [7] Murray Altheim and Shane McCarron. XHTML™ 1.1 - Module-based XHTML. Specification, <http://www.w3.org/TR/xhtml11/>, W3C HTML Working Group, W3C Recommendation, 31 May 02001.
- [8] Johnny Axelsson, Beth Epperson, Ishikawa Masayasu, Shane McCarron, Ann Navarro, and Steven Pemberton. XHTML™ 2.0 (working draft). Specification, <http://www.w3.org/TR/xhtml12/>, W3C HTML Working Group, W3C Working Draft, 22 July 2004.
- [9] Håkon Wium Lie and Bert Bos. Cascading Style Sheets, level 1. Specification, <http://www.w3.org/TR/REC-CSS1>, W3C CSS Working Group, W3C Recommendation 17 Dec 1996, revised 11 Jan 1999.
- [10] Bert Bos, Tantek Çelik, Ian Hickson, and Håkon Wium Lie. Cascading Style Sheets, level 2 revision 1. Specification, <http://www.w3.org/TR/CSS21/>, W3C CSS Working Group, W3C Candidate Recommendation 25 February 2004.
- [11] Wendy Chisholm, Gregg Vanderheiden, and Ian Jacobs. Web Content Accessibility Guidelines 1.0. Guidelines, <http://www.w3.org/TR/WCAG10/>, Web Accessibility Initiative, W3C Recommendation 5 May 1999.
- [12] Jakob Nielsen and Marie Tahir. *Homepage Usability: 50 Websites Deconstructed*. New Riders Press, 2001.
- [13] Inc. Nucleus Research. Spam: The serial roi killer, June 2004.
- [14] Lov om behandling av personopplysninger (personopplysningsloven). I 2000 hefte 8, April 2000.