3

INTERNET ANONYMITY: PROBLEMS AND SOLUTIONS

Claudia Eckert and Alexander Pircher
Faculty of Mathematics and Computer Science,
University of Bremen, Germany
eckert@informatik.uni-bremen.de

Abstract Internet services like the World Wide Web or email programs are already widely in use for private and business work. Unfortunately, with every access a lot of user-specific information is leaked. Hence, using popular Internet-services results in threats against user privacy, as this data can be eavesdropped by attackers or collected by service providers in order to create user profiles. To defeat such threats anonymizer services have been introduced especially for anonymous email and net news. But the available anonymizing services lack a lot of deficiencies and do not provide the required degree of anonymity to Internet users. This is mainly because theses services have been implemented in a rather ad hoc manner lacking a systematic analysis.

The Anonymous-project aimed at revealing and overcoming the deficiencies of existing approaches by following a systematic methodology. Our paper summarizes the main results of the Anonymous-project. It explains the problems and limitations of current anonymizing services and presents our new services.

Keywords: Anonymity, Privacy, Internet Protocols

1. INTRODUCTION

Internet services like the World Wide Web or email programs are already widely in use for private and business work. Unfortunately, with every network access a lot of user-specific information is transmitted over public networks. Examples of such information are the user's IP address, the URL of the previously loaded Web page or date and time of the performed access. Hence, using popular Internet services results in threats against user privacy, as this data can be eavesdropped by attackers or collected by service providers in order to create user profiles. The problem is that the user normally does not know to which extent sensitive data concerning his privacy is collected and stored.

For instance, let us have a look at ordinary emails. Normally, a sender will be willing to give away his own email address, but he certainly does not want to reveal other information like the URL of his previously visited Web page.

Until now, several programs for anonymizing Internet services have been developed. They use different techniques with respect to data avoidance and/or data concealment. Usually, data avoidance is achieved by suppressing the relevant data. Data encryption provides appropriate means to conceal transferred data preventing unauthorized third parties from accessing plaintext information. But data encryption is not applicable for those data items that must be accessible by the authorized communication partner to be able to correctly execute the used transfer protocol (e.g. HTTP, SMTP). Data hiding in such scenarios can be achieved by replacing them with some uniform patterns. The problem is, that the existing anonymizing services just hide or anonymize parts of the sensitive data. The result is that Internet users are not as anonym as they could be. Furthermore, in some services the replacement of original data by uniform patterns results in faults during protocol execution.

All these deficiencies discovered in existing services can be prevented if service development is performed based on a systematic analysis of the Internet-services. Analyzing the protocol specifications of Internet standard protocols enables to classify the transfered data into the class of sensitive data which must be anonymized and all the rest which is not critical. In addition, a serious analysis can reveal to which extent the sensitive data is in fact required to perform the protocol successfully. This has been done within our Anonymous-project [5]. Our aim was to repair the discovered deficiencies by developing anonymizing services which are stronger and more flexible than the existing ones.

The rest of our paper is organized as follows. Section 1.1 presents the main results of our investigation of common Internet services. To be able to compare the strength of anonymizing services we have developed a simple metric which we will explain in section 1.2. Based on our analysis and the metric we have evaluated existing anonymizing services. The results are summarized in section 2. Based on our experiences we have developed our own suite of anonymizing services which will finally be presented in 3. In 4 we summarize the main contributions of our paper.

1.1. ANALYSIS OF RELEVANT INTERNET SERVICES

Popular Internet services like HTTP, FTP, SMTP, or NNTP are client-server applications running on top of the Internet protocol (IP). These services deliver data packets to the IP layer which appends an IP header to each packet. The IP header contains among other data the IP address of the packet's sender and receiver. Hence, an anonymizing service could simply try to conceal the sender's IP address. But this simple technique is not applicable if the protocol requires a bidirectional communication between sender and receiver or if the protocol must send messages back to the sender, for instance, in cases of faults. In addition, we should notice that IP addresses are often already appended to data packets on higher protocol layers by application protocols themselves. This is why address data should be filtered on a per application protocol basis in addition to the IP level filtering. As all the investigated services have in common that anonymity with respect to third parties can be achieved by incorporating encryption techniques, we omit this technique in our subsequent discussion.

1.1.1 HTTP (Hypertext Transfer Protocol).

HTTP [3] is a request/response protocol which establishes a bidirectional connection between client and server. A request message of a client consists of several headers and the message payload. Classifying the header data into privacy critical and uncritical data we observe that the *referer*-entry obviously belongs to the first class. This entry contains the URL of the page from where the server was called. Hence, the server is able to gain context information about his clients by simply inspecting the *referer*-field. Consider for example a client who sends a request in the context of a search engine. In this case, the *referer*-entry contains the whole hit list of the search previously performed by the client. From the server point of view this list might contain interesting information about competitors in the digital market. In addition, the searched key words transfered within the *referer*-entry reveals a lot of information about the client's intends and requirements.

The various *accept*-fields used in HTTP messages belong to the class of sensitive data as well. These fields normally contain the preferred character sets, the language and coding schemas etc. of the client. By carefully analyzing the *accept*-fields an adversary is able to derive a lot of critical information concerning the HTTP client. IP address information can be found in the fields *client-IP, X-Forwarded* and *cache control* which therefore belong to the class of sensitive information, too. Another sensitive field is the *user-agent*-field which identifies the client's user-agent and the optional *from*-entry specifies the email address of the user who is associated with the user-agent. Authentication information is transfered within the *authorization* and *proxy-authorization-* fields. They might contain the user name and user password which is usually just base64 coded. Besides all these headers that are fully specified by the protocol specification, a HTTP message might possess headers which can be defined by users in arbitrary manner, possibly containing lots of user-specific sensitive information.

In contrast to the header fields discussed above which are created by the client system the *cooky*-entry contains data that has originally been created by the server and is stored on the client side. The *cooky* is transfered automatically by the client-browser whenever the client re-connects to the server. The transfered information enables the server to re-identify the client though the underlying HTTP protocol is stateless. The problem is, that usually the client does not know what information is sent to the server encoded in the *cooky*-entry. As a client often reveals privacy related information about himself by filling in Web-forms delivered by the server, the server can extract user-specific information and encode them in cookies. Each time the client re-establishes a connection to the server, the server just decodes the cooky to reveal the identity of the client.

Anonymizing HTTP

We have analyzed 7648 Web requests. First, we observed that in 99% of all of these requests the *accept*-fields contained the above mentioned sensitive data but which was not used by the receivers at all. Hence, within an anonymizing service this data can be either completely avoided or substituted by other patterns without disturbing the server's functionality. In contrast, the information within the *user-agent-* field which was present in over 98% of the requests, actually was used by the servers to tailor the presentation of the required pages to the specific capabilities of the client's browsers. Hence, services which aim at anonymizing these data should be configurable in a flexible manner. This will allow to use the tailored services even in the context of anonymizing activities.

Information contained in the *referer*-field have been transfered in more than 95% of all analyzed requests. This is remarkable, because that information is not necessary to execute HTTP correctly. As we have pointed out previously, an adversary might infer a lot of sensitive information about the sender looking at the *referer*-field. Hence, we recommend to omit all data within *referer*-fields.

Our analysis revealed that the data in the *from*-fields are neither used by the protocol itself nor by service providers. As it might contain a lot of interesting information for an unauthorized third party, we strongly recommend to omit these fields as well, Cookies have been observed in at least 30% of all analyzed requests, though the requested services are usable correctly without them as well. Hence, leaving out cooky data by anonymizing services will not cause an unacceptable denial-of-service.

To avoid the unprotected transfer of IP address information within header fields of HTTP, we recommend to anonymize the fields *client-IP, X-forwarded-for* and *cache control.* This is feasible without disturbing the overall protocol functionality, because these fields are not required in order to execute the protocol correctly. Since the header field *proxy-authorization* is solely required to authenticate the browser with respect to the proxy server, we recommend that the proxy should anonymize this information before forwarding the modified message. Finally, we require that a HTTP-anonymizer should be configurable in such a way that all unknown headers will be anonymized by default. But the anonymizer should be flexible enough to allow selectively an unconcealed transfer of such header data.

1.1.2 FTP (File Transfer Protocol).

FTP [7] provides services to efficiently transfer data. It works session-based and differentiates between control and payload data. A separate control channel is established to transfe control data and this connection is held open during the whole session. In contrast, FTP establishes a new connection for every send or receive transaction for transferring payload data. Notice, that each time FTP establishes a connection either for control or data transfer, the IP address of the client is transmitted. In addition, some commands require parameters which might contain sensitive information. These commands are the *user*-command which identifies the user via an ASCII-string, the *pass*-command which contains the user-password as a parameter, and the *acct*-command requiring a parameter that identifies the user's account name. Using the anonymous FTP service, current Internet browsers normally provide a password that identifies the used browser type (e.g. mozilla@ in the Netscape Communicator). Besides, browsers usually possess the option to transfer the email address of the user which is registered in the browser configuration file.

Anonymizing FTP

Since non-anonymous FTP requires the user name, password as well as user account name to be able to authenticate the FTP-client anonymizing this data at the side of the communication partner (i.e. the FTP-server) is not possible. The same holds for the client's IP address which is required to establish data connections. As mentioned before, encrypting all these data is appropriate to thwart attacks from unauthorized third parties. With respect to anonymous FTP we recommend to anonymize the transfered email address. This can be accomplished for instance by substituting it with a fictive one concealing the true identity of the FTP user.

1.1.3 SMTP (Simple Mail Transfer Protocol).

As we are aware, that there are a lot of good arguments that doubt the appropriateness of anonymous emails we subsequently just focus on one special aspect which we think is an important one. That is, in our opinion an email sender should reveal his identity to his receiver(s), but besides this, he will not be willing to expose any other information concerning his privacy. Examples for such information that should be suppressed are data about the used execution environment like the operating system and hardware. Even more important is the suppression of data about the sending context like links to other messages the email refers to.

The transfer of emails is the main task of SMTP [6]. To this end, SMTP establishes a bidirectional connection between client and server. This connection is afterwards used to transfer several mails. Analogous to FTP SMTP requires the client's IP address for establishing this connection and further sensitive data can be transfered by calling specific SMTP commands. The *mail*-command specifies a parameter that contains the return path to the email sender. This path is used in cases of faults to resend the mail to its sender. The *data*-command is used to transfer the mail data. The data itself consists of a header part and a payload part. One in our sense important header item is the *return-path.* It specifies the sender's address as well as the return path to this address. This path is constructed as follows. Every sender who takes part in transferring the mail to the final destination, appends its own address as well as date and time information to the path. Hence, the path shows in detail the whole route over which the mail was routed to its receiver.

The sender's identity can be derived from other header items as well. Critical in our sense are the *from, sender* and the *reply-to* entries. The *from* field contains the identity of the sending agent (e.g. a machine or a person). The *sender* entry contains the identity of the sender in cases where the author of the message is not its sender. The *reply-to* field specifies the mail boxes to which answers can be send. Furthermore, the *in-reply-to* and *references* header fields contain problematic data as well. They identify the mail to which the current mail replies as well as all other messages that are referenced in the mail.

Besides the fields that are specified in the SMTP standard a mail might contain arbitrary user-defined as well as so called extended header (starting with X-) fields.

Anonymizing SMTP

Our analysis revealed that most of the header fields contained in mails are not required by SMTP. Hence, we recommend to anonymize all header items that carry information beyond direct sender identification. That is, at least the *in-reply-to, return-path* and *references* field should be anonymized to conceal critical data as far as possible not only with respect to third parties but with respect to authorized email receivers as well. As mentioned before, some of these headers are in fact useful in case of trouble shooting. Therefore, the anonymizing service should be flexible enough to allow for individual configurations.

Since extended headers characterize the sender quite good and, in addition, might contain arbitrary data we strongly recommend to suppress all these non-standardized but widely used headers. If complete suppression is not feasible the data should be replaced by random patterns. This will not disturb the execution of SMTP because the protocol does not require the extended headers. Obviously, unknown headers (user-defined) should be completely suppressed by anonymizer.

1.1.4 NNTP (Network News Transfer Protocol).

As before, we do not argue in favor for anonymous postings, but focus on the intension to restrict the transfer of sensitive data to a minimum. That is, we are especially interested in such data that is automatically appended to a posting (or mail, see previous section) without giving the user any opportunity to control or regulate this.

NNTP [4] offers services to read, post and distribute news articles. To this end, the client establishes a bidirectional connection to the news server. This connection is then usable for several actions like reading and posting articles. Like the SMTP protocol NNTP specifies header fields for news articles. These headers are comparable to those used in SMTP and need not be discussed again. In addition, a news header can contain a field called *organization.* This field identifies the organization to which the sender belongs. It therefore carries interesting data that might characterize the sender quite good. Analogous to HTTP and SMTP NNTP allows to use non-standardized headers which might contain arbitrary user-specific data.

Anonymizing NNTP

Because of the similarities between SMTP and NNTP we recommend similar anonymizing actions. That is, an anonymizer should conceal or suppress all automatically generated header data. Again it would be helpful, if the anonymizer is flexible enough to selectively de-anonymize data items. None of the non-standardized headers are required for the correct functionality of the protocol. Hence, all these potential dangerous headers should be anonymized by default. But since some of these data might be used by some servers it should be possible to selectively de-anonymize the required set of data.

1.2. LEVELS OF ANONYMITY

The subsection presents a simple metric to compare the strength of anonymizing services. First, we want to capture the notion of anonymity more precisely. *Anonymizing* in our sense means the modification of privacy-related data with the aim that single data carrying privacy-critical data can no longer be associated with a specific person except a huge amount of money, time and man power will be spent.

A weaker form is given by *pseudo-anonymity.* Here, privacy-related data is modified according to a specific assignment rule (usually by using pseudonyms) with the effect that single data carrying privacy-critical data can no longer be associated with a specific person without the knowledge of the assignment rule.

Now, we will define several levels of anonymity to be able to distinguish between anonymizing services of different strength. The different strengths of the levels result from the different scopes of the anonymizing measures. We distinguish between the following three scopes: anonymity (1) against the communication partner, (2) against third parties, and against (3) the anonymizer itself. The strongest form of anonymity is provided, if the anonymizing service covers all three scopes.

Level 1 Pseudo-anonymity with respect to the communication partner:

The communication partner is not able to associate single data items with a specific person. But all measures to achieve pseudo-anonymity are solely performed by the communication partner himself. That is, this kind of anonymizing can neither be influenced by the user nor is he able to control the success and the correctness of the service. On the other side, the user is completely relieved from coping with anonymizing actions. Services on this level do not provide anonymity with respect to third parties.

Level 2 Pseudo-anonymity with respect to the anonymizing service:
In contrast to level 1, privacy-critical data is only transfered to the anonymizing service and not to the communication partner, which provides a higher level of anonymity. The communication partner solely knows a pseudonym of the user without being able to associate it with a real person. But using such an anonymizing service is not fully transparent for users as they must explicitly call it. And, as before, services on this level do not provide anonymity with respect to third parties.

Level 3 Anonymity with respect to the communication partner:
Analogous to level 1 all anonymizing measures are performed by the communication partner without being controlled by the user. No anonymity with respect to third parties is provided.

Level 4 Anonymity with respect to the anonymizing service:
Analogous to level 2 privacy-critical data are anonymized by a third party before being delivered to the communication partner. Again we do not have a protection against third parties.

Level 5 Anonymity with respect to third parties:
The user is protected against the communication partner as well as against intermediate third parties.

Level 6 Anonymity with respect to the anonymizer:
Up to level 5 the anonymizer always possesses knowledge about its users as well as their communication partners. On this level we require that the anonymizer is not able to infer a connection between users and communication endpoints. In addition, the level requires anonymity against intermediate third parties.

2. ANONYMIZER – STATE-OF-THE-ART

Most of the activities in the area of Internet anonymity concentrate on email and news anonymity whereas anonymizing services for the WWW and FTP area are hardly available. Since we are interested in anonymity of client-related data, we do not investigate projects that concentrate on server anonymity like the JANUS-project [2].

2.1. WWW AND FTP

The most simple anonymizing services just anonymize the log file of WWW-server (e.g. http://www.media.mit.edu/~daniels/software/scramble.html). We call such anonymizer **log file anonymizer.** The log file records all the accesses to the server. If the log file anonymizer is integrated into the Web-server then the data can be recorded in the log file in an already anonymized form. Depending on the fact whether anonymity or pseudo-anonymity is offered, such anonymizers only provide level 3 or level 1 services with all the problems mentioned above. Even worse, the measures incorporated in existing approaches are faulty in such a sense that not all critical data is really anonymized. To be more concrete, these approaches just conceal the host name, IP addresss and user name. But our analysis (see the previous section) revealed that a lot of other data fields containing privacy-critical data still exist, like for example the *referer* or *user-agent* fields. This data is stored unconcealed in the log file of the server.

Proxy-Server

Other implementations of WWW or FTP anonymizer integrate their service into a proxy server. A popular example for this kind of anonymizing technologies is the *Junkbuster* (http::/www.junkbuster.com). Using the proxy approach allows to modify or conceal critical data before the messages are delivered to their destinations. For instance, the Junkbuster suppresses the forwarding of cookies, and of *from-* as well as *referer*-fields and it substitutes user-agent data uniformly by *Mozilla/3.01 Gold.* But, with Junkbuster all other fields containing critical data like the *accept*-fields are forwarded to the final destination without modification. Especially, all unknown header fields are forwarded without being anonymized. Hence, the anonymizing service offered by Junkbuster is incomplete. Since the proxy-integrated anonymizer do not encrypt the data transfer between browser and proxy-server, such anonymizer could only be classified to level 4 or just 2 in ease of pseudo anonymity.

Web-Anonymizer

Web-Anonymizer work quite similar to proxy-integrated anonymizer. One popular representative is the *Anonymizer* (http://www.Anonymizer.com). But in contrast to the proxy approach, the required services of Web-anonymizer are integrated into the Web-server. As a result, this service can be used behind a firewall and the data between browser and Web-server can be transfered in encrypted form. Unfortunately, these advantages are accompanied by an additional management overhead compared with proxy-approach. Additional anonymizing activities are required if a Web-page or a file that has been requested by a client does itself contain references. If the client clicks on these references directly no anonymizing services would be applied to these Web accesses. Hence, such references must be modified to ensure that each call (clicking on the reference) will be directly send to the Web-anonymizer. Obviously, this complicates the implementation of Web-anonymizer considerably which can lead to erroneous services. Furthermore, such anonymizer usually work much slower than proxy-based solutions.

We have carefully studied the above mentioned *Anonymizer* service (cf. [5]). Our analysis revealed that the *Anonymizer* actually anonymizes a lot of privacy-critical data such as the *from,* the *referer* and the *cooky* fields. But other critical data like, for instance, *client-IP* and *cache control* are forwarded unmodified. Furthermore, it is not possible to selectively de-activate the suppression of cookies. Hence, servers which require cookies are not accessible via the *Anonymizer* service. In addition, it should be noticed that the *Anonymizer* does not encrypt the transfered data. Since this restriction is not dictated by the inherent Web-server architecture the level of anonymity implemented by the *Anonymizer* is lower (just level 4) than the one that is reachable in principle. In fact, such anonymizers could provide level 5. Web-server anonymizer could even be improved by using them in a cascading manner which would result in level 6 anonymity.

Crowds

The Crowds-project [8] follows a completely other approach by hiding a user within a crowd. To this end, a user's message is sent encrypted to a randomly selected member of the crowd which selects another receiver among the crowd or sends the message to its final destination. Obviously, a small crowd is not sufficient to guarantee anonymity. The Crowds service suppresses a lot of critical data like *from, cache control, cooky, X-forwarded* and *referer* fields. In addition, it anonymizes the *accept* as well as the *user-agent* data by replacing the original data with default values. But this can cause problems, if a server who interprets this data runs into trouble by using the default values. For example, the value *zip* used to replace other data items in the field *accept-encoding* is not defined in the HTTP specification and might lead to a faulty server action. Furthermore, under Crowds unknown headers possibly containing critical information are transfered without modification. Another disadvantage of the Crowds service is that it is not usable in conjunction with firewalls on the client side. Despite of theses problems Crowds is applicable in a cascading manner. Hence, anonymity level 6 is reachable in principle. But it should be noticed that massage delivery is considerably delayed through cascades of Crowds servers. Note, that the client of a WWW request is not able to determine the crowd members which are involved in the delivery of his message. That is, unreliable members, untrusted members or nodes that are not online might be randomly selected. The service can be improved, if the clients can select a route depending on information about the current availability and load of crowd members.

2.2. EMAIL AND NEWS

Since email and news are asynchronous services it is not necessary that the messages are delivered to their final destination immediately. This is exploited by remailer services which are the most popular anonymizers in this area. The original idea goes back to D. Chaum [1] who proposed the mix approach to transform messages. A user of a remailer service must explicitly send his messages to the remailer. The remailer removes the header information and forwards the message. To thwart traffic flow attacks, the remailer usually delays message forwarding and puts dummy messages into the message stream. This kind of functionality characterizes the so called type 1 or cypherpunk remailers. A list of available remailers can be found under http://anon.efga.org/Remailers/TypeIList. The second class of remailers are the type 2 remailers or mixmaster (http://anon.efga.org/Remailers/TypeIIList/type2.list). But, using this class of email anonymizer requires a specific email client which encrypts the messages and padds them to a uniform lenght of usually 30kbyte.

To be able to use remailers in a remailer cascade, the sender must encode a chain of encrypted remailer addresses into his mail. Each remailer removes the entry of the chain which has been encrypted with its public key. The encrypted entry contains the address of the next mailer in the chain. After decrypting its entry a mailer is able to forward the mail correctly. This technique ensure that only the first remailer in the chain knows the identity of the original sender and solely the last remailer in the chain knows the final destination address. Though such a chaining is feasible, in practice remailing services just use one remailer. As a consequence, a remailer sees all the critical data contained within the header data fields as only the message payload is encrypted. Hence, current remailers just offer the anonymizing level 5 whereas the level 6 is achievable by cascading remailers.

3. NEW ANONYMIZING SERVICES

The previous sections showed deficiencies of existing anonymizers. They do not provide anonymizing services that sufficiently anonymize all critical data (in particular, IP-addresses and *accept* fields are omitted) and the step-wise deactivation of anonymizing measures are scarcely supported. In addition, most approaches implement an anonymizing level that is lower than the one that might be achievable. Therefore, in the Anonymous-project [5] we developed and implemented new anonymizing services to overcome these revealed deficiencies.

3.1. LOG-FILE ANONYMIZER

Our log-file anonymizer has been developed for the Apache Web-server. We anonymize log-entries which contain the following information: *hostname, identd-name, user-name, time, request line, status, amount of bytes being sent, referer, user-agent.* The Apache-server has been configured in such a way that it forwards all theses data to our log-file anonymizer. This data is anonymized as shown in table 1.

The resulting log-file is a compromise. Our anonymizer tries to offer an appropriate level of anonymity for its users (i.e. level 3) while preserving enough information for service providers, for instance, to, generate meaningful access statistics. Our anonymizer covers all privacy critical data. We are aware that substituting the hostname by the top-level name can lead to problems if the server requires more precise information to perform specific analysis (e.g. recognizing accesses originating from robots). Therefore, for this entry as well as for other ones we offer the option to configure the anonymizing measures according to individual server needs. Nevertheless, it should be clear that a log-file anonymizer is not our first-choice anonymizing technique, because the uses is not able to control and to configure it appropriately.

Data	**Modification**
hostname	.< top-level-domain> (e.g. de)
identd-name	–
user-name	–
time	[day/month/year:hours : 00:00 zone]
request	no modification
HTTP-version	HTTP/1.0
referer	–
user-agent	anonymizing browser, OS, language

Table 1 Anonymized log-file entries

Example:

The protocol entry

```
sunsystem.in.tum.de
[18/Apr/2000:13:13:27+02000]
"GET /images/logo.gif HTTP/1.1"
"http://www.in.tum.de/"
"Mozilla/4.5[en](Win98;I)"
```

is anonymized to:

```
.de
[18/Apr/2000:13:13:00:00+02000]
"GET /images/logo.gif HTTP/1.0"
-
"Netscape(Windows)"
```

3.2. PROXY ANONYMIZER

The anonymizing proxy receives requests and forwards them to the original proxy server after having concealed and modified the relevant data items (see figure 1). Hence, anonymizing is performed on the user side. But our proxy anonymizer is not bothered with cache management or other management tasks as this is still performed by the original proxy server.

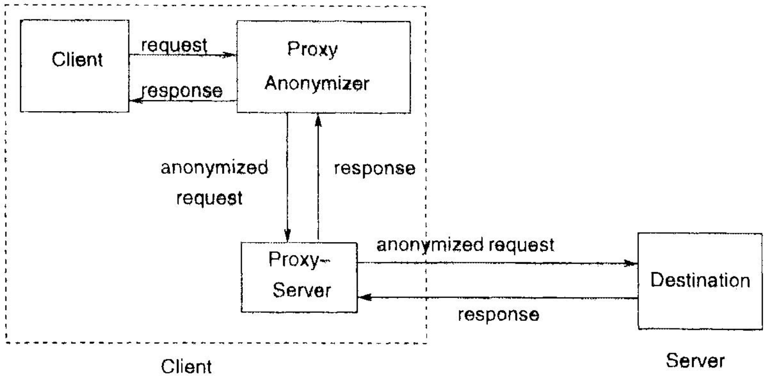

Figure 1 Proxy anonymizer

The anonymizing proxy is registered in the configuration file of the browser. A user can individually configure the anonymizer via a setup page. That is, the user can select the data to be placed into header fields and he can selectively deactivate anonymizing activities for those critical header fields we have previously explained. For instance, the user can determine data items which should be written into the *accept, from* or *user-agent* fields. Additionally, he can substitute the data within those fields that contain information about the user's operating system, his hardware platform etc. As a result, requests from different clients are anonymized in a non-uniform manner. Recognizing anonymous requests is therefore difficult for an adversary. Furthermore, our proxy anonymizer allows to configure the data values that the server appends to the fields *cache-info, client-IP, X-forwarded, via, forwarded.* Since several servers actually use or require the information contained in the fields *host, referer, cooky, cache-control* and *content-type,* our proxy anonymizer provides the option to selectively activate or deactivate the concealment of these fields. Because of the reasons mentioned in the previous sections, our anonymizer automatically suppresses all unknown header fields. The proxy anonymizer reaches anonymity level 4.

3.3. WEB-ANONYMIZER

To use our Web-anonymizer which is integrated into a Web-server the user must specify the file that should be anonymized and must explicitly call our service via an URL. The data will be anonymized and afterwards forwarded to the specified destination. We just forward the data that is actually required to execute the requested service. For instance, in case of the GET method we will transfer the *host*-header whereas in case of the POST method the *host-, content-length-* and *content-type*-headers are forwarded. All server-side answers are anonymized before being delivered to the client (see figure 2). Hence, subsequent accesses of the client on links contained within the answer pages of servers (e.g. encapsulated graphics) are automatically routed via our Web-anonymizer. Besides anonymizing links contained within HTML-data we also anonymize references within JavaScript programs. Furthermore, in contrast to the before mentioned *Anonymizer,* we also conceal FTP references and email as well as news references are automatically forwarded to our anonymizing services (see below). Our Web-anonymizer suppresses all unknown headers and conceals all the privacy critical data discussed in section 1.1. The data can be encrypted by the requesting client, but until now, no cascading anonymizing servers have been implemented yet. Hence, at the time being our anonymizer reaches level 5.

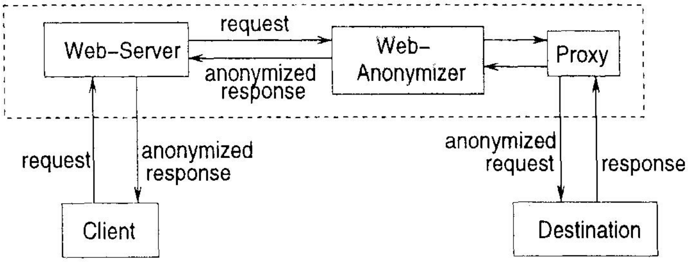

Figure 2 Web-anonymizer

3.4. EMAIL AND NEWS-GATEWAYS

Our email and news anonymizers aim at avoiding the transfer of privacy critical data. To this end, our anonymizing services ask the user to fill in a HTML form with the email data to be anonymized together with the receiver address. Our services just act as gateways as they forward the data to a remailer service within the Internet and send an acknowledgment about the successful forwarding back to the client. The gateways conceal all data except the receiver address, the subject line and the mail payload before sending it to the remailer. A remailer is dynamically selected from the list of remailers. Criteria to select an appropriate remailer are (1) the availability of the service and (2) the functionality that is supported by the service. For instance, our remailer selection takes into account whether the remailer supports the required message format or news support. The availability of remailers is evaluated based on a remailer statistics that is updated once per hour. The level of anonymity can be increased if the sender uses a SSL connection to our anonymizing service which guarantees anonymity with respect to third parties as well. This level is not achievable by just calling a remailer using an email encryption program like PGP, because these programs solely conceal the mail payload and leave the traffic information unmodified. Since no cascading of our service is provided yet, our anonymizing services reaches level 5.

Our Web-anonymizer, and the email as well as news gateways are available in the Internet see `http://anonymouse.home.pages.de/`.

4. CONCLUSION

With the increasing use of Internet services we are faced with severe threats against user's privacy. All widely-used Internet services and protocols transfer a lot of privacy critical data. By analyzing these data, an adversary is able to generate detailed user profiles. Anonymizing services have been proposed and implemented to thwart these threats against privacy. Log-file anonymizer offer very simple, user-transparent and efficient solutions, but the user can not control the anonymizing activities. Browser-supported proxy servers offer simple and efficient measures as well. Anonymizing services are performed on the client-side and are, hence, controllable. But they are not usable behind a firewall and they do not support anonymity against third parties. More elaborated features can be offered by Web-anonymizers. They are usable behind firewalls and support encrypted data transfer. But Web-anonymizer suffer from slow and complicated anonymizing measures. Emails and news are commonly anonymized by using remailers, but information about the sender is transferred unconcealed to the remailer and an overloaded or not available remailer might cause unacceptable message delays.

Our analysis of available anonymizing services revealed a lot of severe deficiencies. Within the Anonymous project we have implemented new services to overcome the existing problems.

References

[1] D.L. Chaum. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. *Communications of the ACM,* 24(2):84, 1981.

[2] T. Demuth and A. Rieke. Securing the anonymity of content providers in the world wide web. In *Security and Watermarking of Multimedia Contents,* pages 494–502, San Jose, 1999.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol — HTTP/1.1, January 1997.

[4] B. Kantor and P. Lapsley. RFC977: Network news transfer protocol, February 1986.

[5] A. Pircher. Anonymity in the Internet. Technical Report, TU-München, Master Thesis (in german), August 2000.

[6] J. Postel. RFC 821: Simple mail transfer protocol, August 1982.

[7] J. Postel and J. K. Reynolds. RFC 959: File transfer protocol, October 1985.

[8] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security,* 1(1):66–92, 1998.