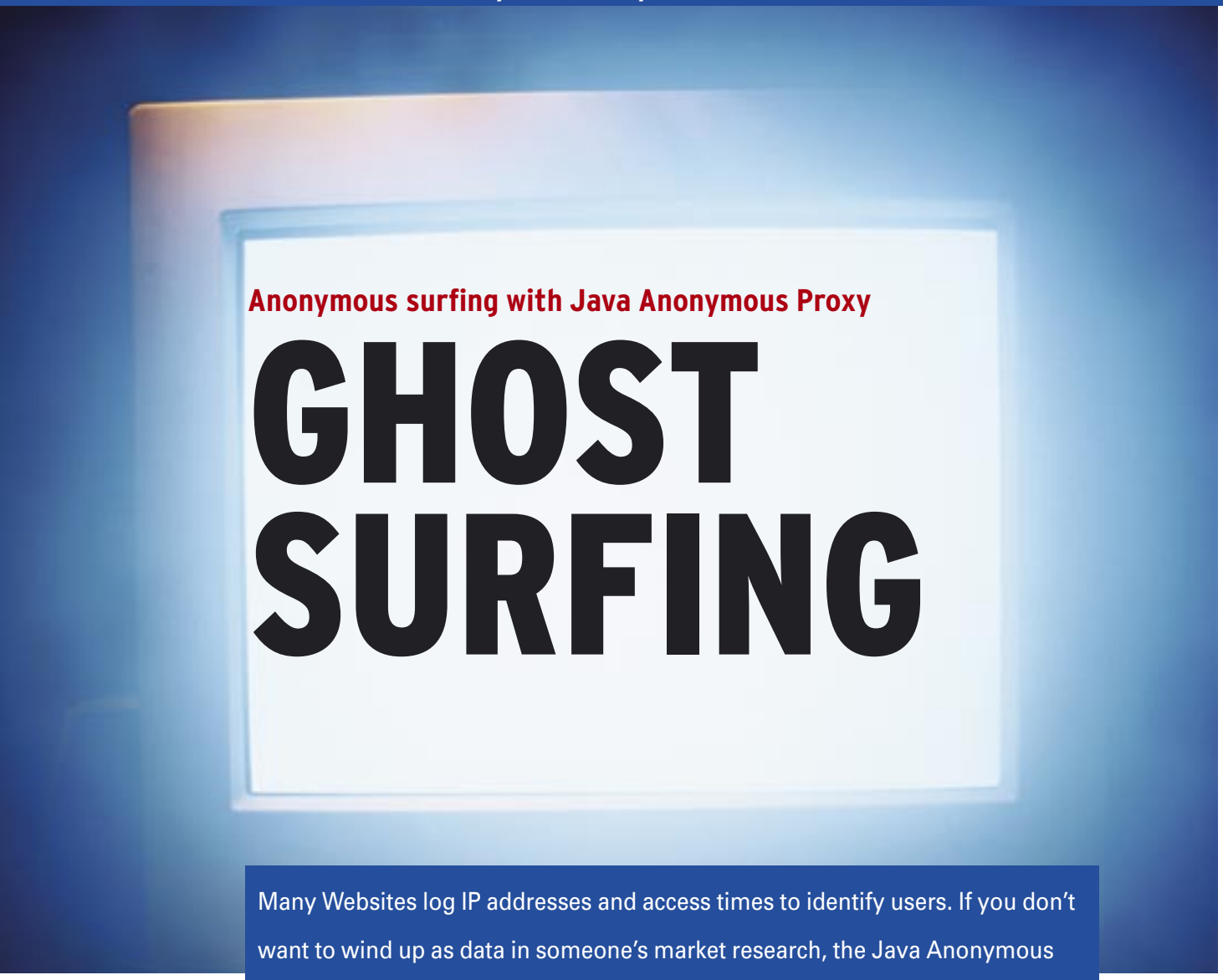

Anonymous surfing with Java Anonymous Proxy

GHOST SURFING

Many Websites log IP addresses and access times to identify users. If you don't want to wind up as data in someone's market research, the Java Anonymous Proxy will keep your surfing secret. **BY TOBIAS EGGENDORFER**

Unless you happen to be a VIP, you can drop into a baker's shop anywhere but in your own home town and pay cash for a loaf of bread without revealing your personal data. Nobody will record the way you move or attempt to discover your behavior patterns. In day-to-day life, anonymity is the rule. In contrast, the Internet allows seamless logging of visitor traffic on websites.

This continuous data collection allows for unique visitor identification; all it takes to identify a user is a short phone call to your provider, who is easily identified via a *whois* entry, to get your personal data. Governments have passed laws to prevent the most extreme misappropriation of personal data, but Internet users with static IP addresses are still easily identified on the web through a simple *whois* request.

To protect your privacy, you need to bring in heavy artillery in the form of an anonymization service. An anonymization service is a service that lets the user surf the web anonymously. The service obscures the user's true IP address, preventing snoopers from following the user around the web.

Users employ a variety of techniques for surfing on the Internet without detection (see the box titled "Private Surfing".) One popular privacy tool is Java Anonymous Proxy (JAP) [1], a portable proxy that supports anonymization in Linux.

Cookies and Anonymity

JAP gives you maximum anonymity, but imprudent handling of cookies can endanger your privacy. Many Websites set cookies, which they use to identify returning visitors. This is not a big issue assuming that cookies are set while you are surfing anonymously. But if you provide user data, or if cookies from previous sessions pre-exist in your browser, any protection JAP can give you is compromised. The remote Web server will still be able to identify you despite the anonymization service.

To help mitigate the risk, you could install two browsers, one for anonymous surfing and the other for "public" surfing. This would avoid mixing cookies and would reduce the risk of identification.

Many Websites require cookies to work properly, so you may not be able to get away with disabling them completely. This said, most browsers give you the option of restricting the validity of any cookies to the current browser session. When you close the browser, the cookies are deleted, thus removing the danger of inadvertent identification.

Solutions such as Cookiecooker [9] are suitable for mixing cookies from ad servers, but they can also lead to you being incorrectly identified. Although Cookiecooker provides some protection against this, there is always the danger of ending up hijacking somebody else's session.

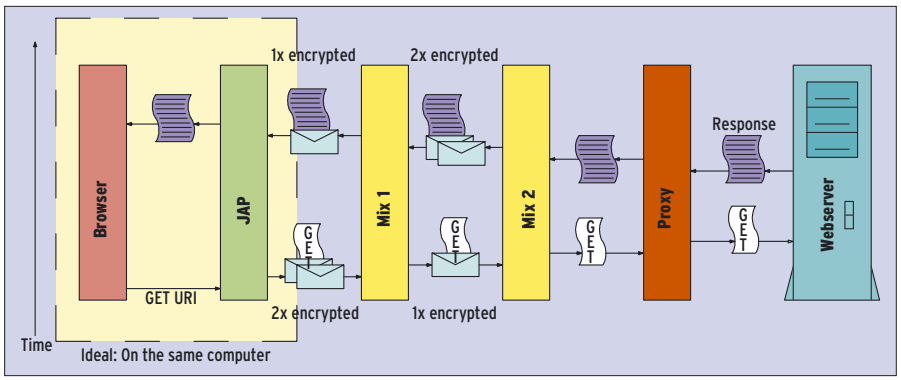

Figure 1: The message is encrypted separately for each mix, like a letter in multiple envelopes, and progresses through the mix cascade to improve anonymity.

JAP encrypts all requests and sends them to a *mix* – an intermediate system on the Internet that mixes user data from several sources. The data bounces through several mixes before finally reaching a proxy that sends the request to the web server. This article describes how you can surf secretly with JAP.

Understanding JAP

The underlying principle behind JAP is simple: data passes from the web client through a chain of several mixes before reaching a proxy server. At each step, the data is mixed with data from other users. The packets are also encrypted at each step. The proxy encrypts the message in a way that the last mix can decrypt it. The last mix takes the cypher text and encrypts it to allow the last-but-one mix to read it. The results are then re-encrypted for the last-but-two mix, and so on.

If a mix in this cascade decrypts a message sent to it, it can only see the cypher text for the next mix. As every mix uses a different key, a high level of privacy is assured. It is like putting a letter in a number of opaque envelopes. Each mix can only open one envelope (Figure 1).

This design provides an additional benefit: as outgoing messages always look different from incoming messages, there is no way of mapping incoming packets to outgoing packets. Even if an attacker were to sniff a mix's traffic stream or take control of a mix somewhere in midstream, there is nothing the attacker can do with the sniffed data. As multiple users route their data packets through each cascade, each cascade contains a nicely stirred cocktail of packets from different users, preventing any conclusions about the packet order.

Do you have neighbors who take delivery of a parcel in your absence and then hand it over to you saying "Hey, big parcel you got there" before going on to speculate about what it could be? JAP puts an end to this kind of speculation by chopping the data into 998-byte chunks and padding smaller chunks with random data.

Just like PGP or SSL, JAP relies on a mixture of symmetric and asymmetric

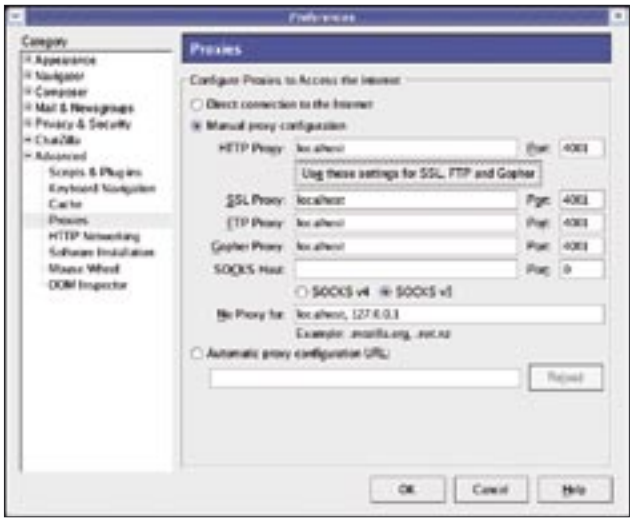

Figure 2: The proxy is easy to set up on the browser-side. You can reconfigure the default port of 4001 in JAP.

JAP Transmission

The data exchange between JAP and the mix, or between mixes in the chain, uses a standard format. Each packet has a size of 998 bytes, and header flags are used to specify important information.

The 32-bit channel ID is the most important bit of information; the ID assigns the mix packet to a mix channel. The channel ID always occupies the first 4 bytes of a mix packet. This is followed by 2 bytes, 5 of which are used for flags. The flags describe the makeup of the data packet. The remaining 11 bits are reserved for possible future extensions. The channel ID uniquely identifies the link between JAP and the mix, or between two mixes.

The open flag (bit 45) is set on establishing a connection to specify a new, random channel ID. The new channel relies on a symmetric key, the asymmetric, RSA encrypted version of which is stored in the first 128 bytes of the data packet. The rest of the packet is encrypted symmetrically using AES.

The payload has a 3-byte header that specifies the exact length (2 bytes) and the user data type (1 byte). This data type can be FTP or HTTP. The last mix in a cascade identifies a suitable proxy based on the data type.

The asymmetric part is the interesting bit here. The first 16 bytes contain the 128-bit key used for symmetric encryption by each mix. To allow the next mix to learn its key, all the data in the packet are left shifted by 16 bytes, and the data packet is padded with 16 random bytes to take it up to the standard size. The mix again encrypts the first 128 bytes with the next mix's key. If the open flag is not set, the key exchange does not take place. This increases the size of the packet's payload component.

Mixes set the close flag in the header to close a connection and pad the data component with 992 random bytes. Whenever a mix receives a packet of this kind, it drops the channel ID and the matching symmetric key. The packet also tells JAP that all data have been transmitted and closes the TCP connection to the browser. To prevent attackers from deducing the packet type from the header data, mixes also encrypt the header.

encryption for communication between mixes and the local proxy. JAP encrypts messages to the mixes with random symmetric session keys. On first contact, the symmetric key is then encrypted asymmetrically using the mix's public key.

Installation

Despite its apparent complexity, JAP is simple to install and use. Just download the JAP version that matches your own Java version from [1]. To discover your Java version type:

```
java -version
```

Then launch JAP by giving the following command:

```
java -jar JAP.jar &
```

This command should work fine, assuming you have a GUI and assuming that the *DISPLAY* variable is set appropriately.

All you have to do now is modify the proxy configuration (Figure 2). By default, JAP listens on port 4001 and only allows queries by *localhost*. If you are looking to install a single JAP for your local network, you will need to enable the *Forwarder* to give other machines access.

Choosing a Cascade

The other settings, which you can access via the *Settings* or *Details* buttons, are self-explanatory. The only question JAP newcomers then face is the choice of the best available cascades.

Cascades define your degree of anonymity: the more users that populate a cascade, and the bigger the cocktail of packets, the more difficult it becomes to sniff the cascade and identify individual surfing behavior.

On the other hand, the more users there are accessing the service, the slower Internet access will be. JAP distributes this information via its information service, which gives you a list of currently available mix cascades and their performance.

For security reasons, the JAP client checks if the list is correctly signed. This precaution prevents a would-be attacker from inserting a rogue cascade.

Conclusions

Java Anonymous Proxy (JAP) is a portable anonymous proxy application for Linux. Compared with the relatively complex technology on which the JAP is based, JAP is very easy to install and use. Once you have installed JAP on your own system, you can test your configuration using the testing service provided at [8]. ■

INFO

[1] Java Anonymous Proxy (JAP): *http://anon.inf.tu-dresden.de/index_en.html*

[2] The Cloak: *http://www.the-cloak.com*

[3] Guardster: *http://www.guardster.com*

[4] Anonymization.net: *http://www.anonymization.net*

[5] US Senate proxy: *http://online.securityfocus.com/news/1780*

[6] JAP and crime prevention: *http://anon.inf.tu-dresden.de/strafverfolgung/index_en.html*

[7] JAP at the command line: *http://anon.inf.tu-dresden.de/develop/commandline_jap_en.html*

[8] Anonymity test: *http://anon.inf.tu-dresden.de/anontest/test_en.html*

[9] Cookiecooker: *http://cookie.inf.tu-dresden.de*

Private Surfing

Web-based anonymization services such as The Cloak [2], Guardster [3], or Anonymization.net [4] provide a simple form of anonymization. When you type in a URL, the anonymization service requests the page from the target Web server, often analyzing the HTML and replacing links so that the links also use the anonymization service. Finally, the service serves up the requested page (Figure 3).

Web-based anonymization has a few drawbacks, one of which is the question of trust. The user does not know what the service logs or how anonymous the service really is. For another thing, it may be impossible to rewrite links that generate Javascript, and that would certainly break the surfer's cover. (To counter this, The Cloak has the option of removing all Javascript content from any pages you visit.)

Finally, the path between your own browser and the anonymization service is not encrypted, so any host, including your own provider's proxies, could actively log your surfing activities. This problem has prompted many web-based anonymization service providers to offer encrypted services via HTTPS, although these services may not come for free.

A proxy provides an elegant solution to the problem of parsing HTML pages and replacing links. This involves configuring the browser to transmit every single HTTP request to the proxy. The proxy then talks to the target server. Unfortunately, a conventional proxy gives you nothing in the line of anonymity. For example, the AOL proxy knows its users. So-called open proxies are better suited to the goal of anonymity; open proxies are proxy servers that anyone can use. Even the US Senate has provided an open proxy, although unwittingly [5]. To identify a surfer using an open proxy, you first need to check the open proxy's logfiles and then contact the provider. However, few web servers actually save this header content in their logfiles. (Note that some proxies do reveal the user behind the requests by adding an HTTP header that contains the IP address of the requesting machine.)

An open proxy is no solution to the issue of non-encrypted traffic. If an attacker can sniff the communication between the proxy and the web user, that user is no longer anonymous.

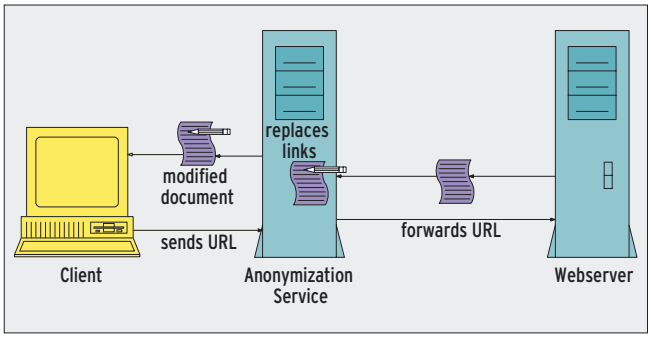

Figure 3: A Web-based anonymization service separates the user from the Web server, but the intermediate anonymization server could still track user information.