

Transparent Anonymization of IP Based Network Traffic

Lexi Pimenidis and Tobias Kölsch*

RWTH-Aachen University,
Computer Science Department Informatik IV,
Ahornstr. 55, D-52074 Aachen, Germany

`{lexi,koelsch}@i4.informatik.rwth-aachen.de`

Abstract:

This paper presents an approach to enhance the usability of anonymizing networks by creating a virtual anonymous IP-network. An anonymization layer is hidden behind the operating system and transparently reroutes all network traffic. This enables the user to perform all IP based network access without adaption of the used programs. The provided IP can be used to provide services on the network anonymously. The generality of the approach also enables typical IP based services like DNS to be provided to the anonymous hosts.

1 Introduction

The distribution and vast popularity of global networks like the Internet allow end users to communicate world wide. Unfortunately most end users and even professionals do not pay enough attention to secure their systems and network traffic against intrusions, eavesdropping or data analysis. It is difficult to cause awareness to these problem, as the threats are often invisible. As a result sensible data is send carelessly through the network. However, not only the content of a message contains sensible information that has to be protected, e.g. by encryption. The address information that is included in most network transmissions can also contain sensitive information. So sending encrypted data in regular intervals to the web server of a certain political party leaks information of political affiliation.

To counter this leakage of information several anonymization techniques have been proposed, developed, and deployed. E.g. Mixminion [DDM03] anonymizes email traffic on different network layers, freenet [CSWH00] provides an anonymous storage and publication system, JAP [BFK00] and MorphMix [RP02] are anonymizing HTTP-proxies, and Tor [DMS04] provides a Socks proxy and anonymizes TCP/IP streams on the transport layer.

A problem common to all of these techniques is that they are developed for a special

*Thanks to Dr. Doğan Kesdoğan and George Danezis for their invaluable help

purpose. As a result they can not be used by all programs that use the Internet to perform day to day work.

Tor is the one notable exception. The general Socks4a interface it provides, gives all application that support this general proxying protocol the ability to anonymize their traffic. Applications not supporting it can be started using the `torify` command, which redirects calls to the networking functions of the operating system. However, this approach still bears some problems. First, the configuration overhead is significant, as it has to be figured out for every application if it supports the Socks4a protocol and the application has to be configured to use it. This is complex and error prone, especially as the user usually gets no feedback, whether the traffic is really anonymized or not. Second, the `torify` approach does not prevent the torified programs to issue dynamic name service (DNS) requests, that disclose the users target of interest.

Another problem with Tor is that the back channels it provides are not adequate for general usage. The setup of such channels has to be done in the Tor configuration file, and is as such bound to a restart of the anonymizer. Another deficiency of the present approach is that the anonymous domain names are represented by a cryptographically hashed URL that is usually difficult to remember and can not be accessed without using Tor.

The communication suite formerly known as “Tarzan”[FM02] would qualify for a piece of software that actually does provide the envisioned ease of communication. Alas, it has to be compiled by the user, is still in experimental status and primarily supports only FreeBSD. Since we’re looking for a solution that can be accomplished with software that is known to work, and does so on a variety of operating systems, we have to choose other components.

Since the configuration needed to communicate anonymously with existing techniques is for experts only, the set of users is small. This reduces the anonymity set provided by a technique. As a result the users are not as well protected as they could be and they are possibly prone to stigmatization for using anonymization techniques.

Our solution to this problem are *Virtual Anonymous Networks (VAN)*. They introduce an anonymization layer that is hidden behind the operating systems and allow transparent access to the global network through an anonymizing overlay network. The approach provides users with truly anonymous IP addresses, that can be used for sending and receiving IP packages. As a result all their traffic is hidden behind this temporary address. The user side configuration is by this reduced to setting up just one system instead of having to adapt every program.

We built the proposed architecture using publicly available tools and evaluated its functionality to ensure practicability and usability. The resulting network allowed easy, anonymous, TCP/IP based peer-to-peer communication for all applications without additional configuration.

In section 2 we will describe VANs in more details. Some informations about the proof-of-concept network we created to verify our ideas are given in section 3. The results will be discussed in detail in section 4 and an outlook will be given in section 5.

2 Virtual Anonymous Networks

We propose to deploy an IP based overlay network that is able to anonymize data streams, but looks like any other IP based network to the user. Clients can connect to this network, like they dial up to an Internet service provider. But the IP address they get, is not linkable to them. They can then send and receive data anonymously over this virtual network. Additionally it is possible to add typical service functionality to this network, like DNS. The IP communication itself is not restricted by the approach. However, the operators of the network may introduce restrictions, e.g. using a firewall, to control the traffic flow.

The overlay network consists of two additional network layers. An anonymizer on the lower layer hides the host's real IP address against the VAN servers. We chose Tor for this purpose. The second layer is provided by a virtual private network (VPN) that provides the anonymous IP addresses. OpenVPN [ope04] was chosen to provide this for our implementation. Both tools are installed on every host that should get an anonymous address.

Using the anonymization layer, the VPN clients are then able to connect to one of the VPN servers and receive an IP address from the servers private IP range. Consequently the client can communicate anonymously with other clients in the same virtual network. If different virtual networks are interconnected, all clients can communicate with each other, without knowing the other user's identity.

After our approach is set up every client computer will have at least two IP addresses. The first one is that of the real network device. All traffic that uses this address is not automatically protected and will be routed directly to its destination. The second IP is that provided by the VPN server. Messages that are sent or received by the host on this address can not be linked to its real IP address. It now depends upon the routing of the operating system, which traffic will be anonymized and which will be send directly into the Internet. One possible configuration could send local traffic directly, while sending traffic to remote hosts through the VAN.

In the following we describe the used tools and give a more concise description of the implementation of our method.

2.1 The Anonymizing Overlay Network

As the users host should be anonymous towards the VAN server, all traffic to him is proxied through an anonymizer. As pointed out earlier we chose Tor [DMS04]. Tor has the most general approach to anonymize network traffic and it is capable of relaying real time TCP traffic through untrusted networks. Another advantage of Tor is that it can be run on various operating systems, e.g. Windows and Linux. It also does a basic form of network load balancing because the clients choose their paths through the Tor network at random.

Tor in its current form is not designed to cope with global observers or similar threats often considered in actual research [KAP02, Dan03, KP04, MD04]. But it does a good job in hiding information from locally restricted attackers, as network administrators or other

local authorities.

The Tor network consists of Tor servers and clients. A client that wants to communicate, builds a tunnel that randomly passes through some of the servers. TCP/IP packages of a client application that are to be sent to a host in the Internet, are relayed through this tunnel. The last Tor server then proxies the connection to the final destination and sends the answers back through the tunnel. The way in which the tunnel is instantiated guarantees that each server only knows his predecessor and his successor. So only the first server knows the client and only the last server knows the final destination. Also the appearance of the sent packages is changed at every hop such that two non neighboring Tor servers cannot link individual packages to each other based on their appearance.

The Tor client provides a Socks4a interface for applications. This protocol is capable of handling unresolved URLs, such that IP addresses do not have to be retrieved before a TCP connection is set up. As presented in Section 1, applications that do not support proxying with Socks4a can be made to compatible using the `torify` command. However explicit DNS resolution undermines the gained anonymity.

2.2 The VPN Overlay Network

A VPN server allows multiple clients to connect. Each client receives an IP from a given range, and all traffic on this virtual overlay network is then either forwarded to the participating clients or masqueraded to outside IPs (masquerading is also known as *network address translation*). After the initialization of the VPN, the users are able to communicate with each other, as if they were on a single LAN sharing IPs from the same range, although they actually are not and their external IPs are from completely different networks.

Under normal conditions each client needs a unique certificate to authenticate himself to the server. Obviously we do not want to give away a client's identity this way, so we configure the server to accept multiple connections with the same certificate and publish a single client certificate that is needed to connect to the server. On the other hand, it is desirable for the servers to have each a unique certificate so no attacker can impersonate them.

Although a single VPN server would be sufficient to protect anonymity, we want multiple VPN servers for scalability reasons. This leads to a routing problem for traffic between different VPNs. A simple solution is to create a fully connected network between the VPN servers. This can be done by interconnecting the VPN servers to each other. Traffic to the other networks is then directly forwarded to the target VPN server, who then forwards the data to its destination. If the amount of VPNs increases some more dynamic routing method for packages between the private networks might be necessary.

We use OpenVPN 2.0 [ope04] as virtual private network solution. This choice was made because it runs on most major platforms, it is easy to set up, and it is easily adapted for our purpose.

To protect the users DNS queries, a DNS server can be installed on the VAN server. We

used the popular `bind` [bin] server for this.

2.3 Connecting and Using the Network

In order to use our *virtual anonymous network (VAN)*, the user needs to install a Tor-client and an OpenVPN-client. After he starts the Tor client, he connects the VPN client to a VAN server through the Tor network. The server provides him with an IP address that can be considered anonymous. The VPN client then replaces the default gateway on the user's computer with the VAN server. He also configures the computer to redirect all DNS-queries to the VAN's DNS server. As soon as the second connection is established, the user's Internet traffic is effectively anonymized without the need to further configure applications.

It is even possible to configure only the gateway of a local area network. This anonymizes the entire networks outgoing traffic without the need to configure each single computer. The gateway is set up as described above. Though this reduces the setup effort, this solution suffers from the typical effect of masquerading: inbound connections on the anonymous IP-address of the gateway can not easily be targeted to the single machines in the network. Additionally, local observers are able to completely compromise the system.

3 Implementation

To build the proposed architecture we used a closed network of nine networked computers. Five of them were running as Tor and VAN servers. The remaining four were used as clients as shown in Figure 1.

Setting up the Tor nodes in an closed environment requires us to replace the default directory server by one of our Tor nodes. This node also receives the other nodes' fingerprints to classify them as trusted. The nodes interconnect as soon as they are started.

Prior to starting the VPN-servers, it is necessary to create a certificate authority and create keys and certificates for the participants. We created one certificate per VPN-server and one for all clients. After starting the VAN-servers, they need to be interconnected pairwise.

On the four client computers the Tor client needs to be installed and configured to use the local Tor network, instead of the global one. After the Tor client successfully connects, the VPN-client can be started. The client connects to a random VAN-server to avoid linkability with prior IPs and for load balancing purposes. It then redirects non-local traffic to the VAN-server through the Tor network.

For testing purposes we pinged the clients pairwise and transmitted data by TCP and UDP without any problems.

Figure 1: Our experimental network consists of five interconnected VAN servers and four clients. The clients IPs are anonymized by the Tor network and their VAN IPs are used for inter communication. The picture does not show that the VAN servers also act as Tor servers. Also some connections between the VAN servers are left for clarity.

4 Discussion

Once set up, the presented method enables the user to communicate anonymously without any further adjustment of programs. The anonymizing network prevents the VAN-server or an observer from learning the identity of a user through his IP address, so the VAN IP address can not be linked to the individual.

One of the advantages given for free by using OpenVPN is that of being able to resume broken down network connections. This is especially interesting for mobile users.

Also, having regular IPs as addresses, it is possible to assign ordinary fully qualified domain names to anonymized IPs. Such, a name like *medical-forum.net* can be resolved to one of those IPs. The DNS entry has to be changed every time, the service changes its IP, but there are established solutions to this problem (e.g. *dyndns.org*).

Anonymity in VANs Security properties, like the degree of anonymity, have to be re-considered in compound architectures because they may be weaker than those of the single parts it consists of.

The anonymity of users in VANs is not better than that of ordinary Tor. Especially as the anonymous IP stays the same over a longer period of time and is the origin of requests from exactly one user, his different communications can be linked to each other. Also, if an attacker finds out a user's anonymous IP, he can attack this user although he does not know his real address. This is impossible against Tor users without VAN. Direct attacks against an anonymous IP include port scanning and fingerprinting. However, the countermeasures

known from regular IP networks can be applied here, e.g. fire-walling inbound traffic. If those techniques are correctly applied, active adversaries gain little more information than passive ones.

Problems and Issues In this section we are going to discuss open problems and issues related to the proposed architecture. Where possible we will suggest a solution.

One problem is, how to distribute the OpenVPN certificate for the client computers. This can be solved in different ways: VAN servers can run a HTTP server where they provide access to the needed software and certificates. Another solution is to package it directly with the VAN client software.

Another problem is that a central authority is needed to provide disjunct IP-ranges and certificates to the operators of the VAN-services. This authority might also support the routing between local networks of the VAN, as our approach is only suitable for very small VANs.

Another issue is that the anonymous IPs are not accessible to outside hosts. A solution to this would be to use globally assigned IPs and have the range's owner provide a gateway to the VAN. However it might not be a good idea to make the services available to outsiders, as this reduces their motivation to use the system. As a result they do not increase the anonymity set.

For a large scale system, the choice of the IP-range of the virtual network is also crucial. The private IP-ranges 10.0.0.0/8 and 192.168.0.0/16 are used too often in LANs, such that collisions are likely in a global environment. However mostly the lower values of these ranges are usually used such that sub ranges with higher values might be a fine choice. Also the private range 172.16.0.0/12 is seldomly used and might be a good choice in a global environment. A wide support of IPv6 will help here.

Although it looks convenient to apply DNS-services on anonymous IPs, this allows certain impersonation attacks on anonymous services. If a service recently changed its IP, an attacker can try to gain the old IP to impersonate the earlier owner. The use of SSL secured connections that use a certificate which refers to the DNS entry and is reliably certified by some root CA can solve this problem.

Last but not least, this architecture does not protect the user who identifies himself by his choice of software, the fingerprint of his operating system, or by sending their identity on the application layer. Application layer filters and proxies can provide some help at this stage, but there is no satisfying solution right now.

5 Conclusion and Outlook

We have presented an architecture based on Tor and OpenVPN that enables transparent anonymization of all IP based network traffic and provides users with anonymized IP addresses. This approach reduces the configuration overhead, as it only has to be performed at one single point instead for every single application that's traffic should be anonymized.

This decreased complexity also increases reliability, as the user only has to assure this one system is working correctly.

Since the degree of protection of anonymity networks grows with the number of active users, we consider it important for them to be highly usable and versatile. We showed that current techniques can be extended by existing software in a way that allows most convenient and secure usage of anonymity networks.

Furthermore we discussed arising problems and validated the practicability of our approach by setting up a prototype and evaluating it experimentally.

We look forward to see some of our ideas implemented in a single piece of software, possibly even as kind of a plug-in to Tor or another anonymity system. Alternatively some out of the box package of our prototype could lead to a larger popularity and result in the required anonymity set.

References

- [BFK00] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.
- [bin] ISC BIND, Berkeley Internet Name Domain. <http://www.isc.org/sw/bind/>.
- [CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66. Springer-Verlag, LNCS 2009, July 2000.
- [Dan03] George Danezis. Statistical Disclosure Attacks: Traffic Confirmation in Open Environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.
- [DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.
- [DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.
- [FM02] Michael J. Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.
- [KAP02] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of Anonymity in Open Environments. In *Proceedings of Information Hiding, 5th International Workshop*. Springer Verlag, 2002.
- [KP04] Dogan Kesdogan and Lexi Pimenidis. The Hitting Set Attack on Anonymity Protocols. In *Proceedings of Information Hiding, 7th International Workshop*. Springer Verlag, 2004.

- [MD04] Nick Mathewson and Roger Dingledine. Practical Traffic Analysis: Extending and Resisting Statistical Disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, LNCS, May 2004.
- [ope04] The OpenVPN-project. <http://openvpn.sourceforge.net/>, 2004.
- [RP02] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.