

OnionCat – An Anonymous Internet Overlay

Application and Usage

Bernhard R. Fischer

St. Pölten University of Applied Sciences

Matthias Corvinus-Straße 15, 3100 St. Pölten, Austria

2048R/5C5FFD47 <bf@abenteuerland.at>

Abstract

OnionCat is an anonymous Internet overlay. It allows users to share any kind of IP-based services with the advantage of anonymity. This greatly improves users' privacy and defeats surveillance.

IP-based sharing of services is exactly what the Internet does – web services, email, chat rooms, and many more. But unlike the traditional Internet with OnionCat users' locations cannot be ascertained using their "IP-footprints" they leave within every logfile on the net. OnionCat gains its anonymity by using anonymizing networks like Tor or I2P¹ as its transport. It is available on various operating systems including Windows.

This paper explains how OnionCat works and gives instructions about its application and usage. With a continuously growing community the OnionCat network could evolve into a feature and information rich network like we know the standard Internet today.

1 Introduction

OnionCat [2] may be used in a wide range of different applications. It provides a kernel interface, i.e. a network device, to which an IPv6 address is assigned. Thus it provides one of the most compatible interfaces possible. Of course an IPv4 interface would be even more compatible, but it does not fulfill the requirement for the most important development goal of OnionCat.

OnionCat is based on anonymizing transport layers like Tor,² [8] thus it may be used by various user groups for the same reasons as they use, for example, Tor. OnionCat is an extension of anonymizers. It adds features but it also extends the group of different users and use cases.

The most important goal of OnionCat is to enable users to transport raw IP data across an anonymizing network together with automatic IP address configuration. This has two considerations.

1. It makes it easier to use.
2. It creates a single logical virtual network segment so that all users can share it. Thus they are automatically connected virtually together.

The second item is achieved by every VPN, but different from any other VPN the OnionCat network is an open network. Every user can take part without any restriction or limitation in

¹<http://www.i2p2.de/>

²Currently it works just with Tor. Development of adapting to I2P is in progress.

respect to network addressing. A user can decide to change his address at any time³ and he can also leave the network again without leaving traceable footprints. Without further requirements one can use OnionCat to achieve the following use cases.

- Usage as an open anonymous network (This is described above).
- Usage as a real VPN for a privately set up closed user group.

Both are fitted to bypass surveillance or supervised networks of any kind.

In the following sections I will discuss the concept of OnionCat as well as the setup to act as a client in the open anonymous network.

2 Behind the Scenes

OnionCat basically is a *virtual private network* (VPN) from a computer science point of view. One of the most generic definitions is found in [4]: “A Virtual Private Network is a network of virtual circuits for carrying private traffic.”. I will refine and explain these terms more specifically. Virtual circuits are connections between nodes. Those connections do not exist physically, but virtually. TCP sessions, for example, could be seen as virtual circuits. While browsing the web a connection between the local computer and the webserver seems to exist but this is just a virtual connection. Of course those two are not connected physically, but still the connection carries some information, as is the case for web pages.

What Kosiur ([4]) further says is that those circuits carry *private* traffic. This does not necessarily mean that the traffic is always personal or secret to somebody. This might be the case but it is not a must. In some cases those connections could carry both types of information and very often it is only a matter of definition what is private and what is not.

Many VPNs share a similar concept, which is carrying traffic of a specific type across a network of the same type. In most cases VPN applications, like the Microsoft VPN, Cisco’s VPN and OpenVPN carry IP packets. Usually they achieve the same goal, which is to access some kind of “private” network. A prime example would be a company’s internal network. It’s designed to work for people who have Internet access. Obviously Internet is based on the *Internet Protocol* (IP). Thus follows that those types of VPNs carry IP packets within IP packets. Expressed in a different way, IP packets get encapsulated within IP packets. Figure 1 shows a simple diagram of how VPNs fit into the OSI layer model.⁴ This model defines that each upper layer depends on its lower layer and every network protocol can be assigned into a specific layer.⁵

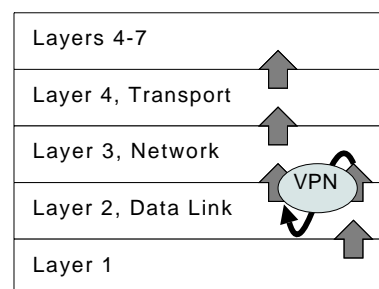


Figure 1: VPN intermediate layer.

The figure highlights three layers. Ethernet is contained within the *data link layer*. Usually we carry IP within Ethernet, hence IP is one layer above and is called *network layer*. On top of IP we have protocols such as, TCP and UDP, and categorize them into layer 4 – the *transport layer*. If a VPN is in use, IP is encapsulated into IP and not, for example, TCP into IP as the layer model suggests. Thus, from an architectural view, it inserts a second IP layer. That is what

³This is the case only if the anonymizing transport allows this. But Tor does as well as I2P.

⁴The OSI model discriminates between seven different layers for classification of network protocols. It also discriminates them based on their dependencies. A detailed explanation of the model can be found in [7].

⁵That is not entirely true because all models are simplified pictures of reality, but within this context that is not of importance.

Figure 1 depicts. Above layer two the VPN layer follows (which actually also is IP). On top of that layer an IP layer follows.

If a VPN is implemented there is always some kind of VPN layer. The VPN layer creates the virtual circuits. The difference between various VPNs is where they insert the VPN layer in respect to the OSI model. Figure 1 gives just an example of encapsulating IP within IP.

2.1 OnionCat VPN

As already mentioned, OnionCat is a VPN. As has been previously explained, VPNs consist of two fundamental parts. The virtual circuits and the traffic they carry. Both can be fitted into the OSI model and both may not be of the same layer.

OnionCat does not create a completely new type of virtual circuit. It uses circuits which are created by anonymizing networks upon request. For now OnionCat supports Tor. I2P is in development.

Tor's virtual circuits, which are relevant for OnionCat, exist only within the Tor network and connect two Tor nodes. As it is the case for most virtual circuits, one end initiates the connection and the other end accepts it. The latter usually is referred to as *server*. Within Tor nomenclature this server is called *hidden service*. The circuits are based on TCP. As such they are above layer 4 in respect to the OSI model. Part of the nature of anonymizing networks calls for the capability of two nodes (connected by a virtual circuit) to open up communication channels, but not know who or where the other node is. After circuit setup Tor does not care about data carried within it. It just manages that bytes piped into it at one end and drop out at the other end and vice versa.

For all virtual circuits addressing is required to designate a connection to a specific server. For TCP sessions, addressing is achieved by an IP address⁶ and a port number. Tor uses *onion-URLs* to address a specific hidden service. Onion-URLs are unique to a hidden service like IP addresses are unique for servers within the Internet. Onion-URLs are human readable 80 bits of address information based on some cryptography as described in [6].

OnionCat requests Tor to build such virtual circuits and sends raw IP data across. In this application the virtual circuits carry IP data as it is true for most VPNs. Figure 2 shows how OnionCat fits into the architecture of network protocols as defined by the OSI model. In the upper right corner it shows Tor's virtual hidden service circuits. They are based on TCP, hence, they are located above transport layer within the model. OnionCat (the cat's paw) inserts the VPN layer, it's not just an insertion as it was shown in example Figure 1. OnionCat actually makes a bridge from above the transport layer down to the network layer.

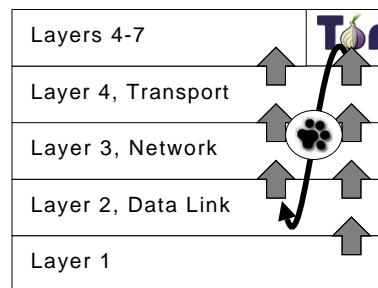


Figure 2: OnionCat in layer model.

2.2 OnionCat Addressing

A typical configuration for most kinds of VPNs is that they are setup in a static way. An example of this would be how their virtual circuits are addressed. It is common for organizations to run a centralized VPN entrance point to which all VPN participants connect. This setup is easy and usually matches all requirements for such a private VPN. But it is not suitable for an open anonymous network for several reasons.

⁶Actually IP addresses are property of IP and not TCP but that makes no difference within this context.

1. The person or organization that runs the entrance point probably will not stay anonymous. Even if they never appear in the public, such an entrance point might be revealed due to the fact that it is a traffic sink.
2. A centralized service is always a single point of failure.
3. The service provider might enjoy unlimited trust of its users which obviously would never be the case in today's world.
4. That kind of service could attract certain interest of various organizations like intelligence services.

Hence, the approach is to distribute it. To connect a Tor client to a hidden service, an example being establishing a virtual circuit within Tor, it is required to use Tor's addressing method of choice for hidden services. As explained above in Section 2.1, Tor uses onion-URLs which are 80 bit long addresses. If we assume that every client runs his own hidden service, then all of them also get a unique hidden service address – an onion-URL. This leads to the interdependency that every client can connect to every other client since every client now also is a uniquely identifiable server.⁷

The difficulty now arises from layer discrepancy. OnionCat lies between Tor on one end and the operating system on the other end. In respect to the layer model (see Figure 2) Tor (the hidden service) operates above layer 4. The other end of the VPN layer which OnionCat creates is at layer 3, which is the IP layer. Every layer has its own addressing method. Hidden services use the 80 bit long onion-URL and the IP layer obviously use IP addresses. A static configuration, one example is a configuration file, would solve that problem but not in respect to the requirement from above of *not* being static.

We looked for a complete dynamic solution which does automatically exclude some kind of "configuration file update service". The solution lies within the IPv6 protocol. IPv6 uses 128 bit long addresses. This is a huge address space and obviously greater than 80 bits. Because OnionCat should act as a private network with public access, we chose a network prefix of the *unique local IPv6 unicast addresses* according to [3]. It perfectly meets the requirements for OnionCat. The smallest possible prefix length as defined in the standard document is 48 bits which leaves another 80 bits for addressing hosts. Using this configuration, OnionCat can translate IPv6 addresses to onion-URLs and vice versa. If an IPv6 packet arrives from the operating system OnionCat extracts the lowest 80 bits from the packet's destination IPv6 address, translates it into an onion-URL, and requests Tor to open a virtual circuit to the desired destination. After the connection is setup OnionCat starts forwarding all packets through this virtual circuit. On the other end of the virtual circuit, OnionCat receives the packets from Tor and forwards them to the operating system. The operating system then in turn does with IP packets what it always does. From the operation system's point of view, there is no difference if a packet arrived on a physical Ethernet interface or from OnionCat's virtual tunnel interface. More details about OnionCat's addressing and forwarding mechanism can be found in [1].

This method perfectly distributes the VPN entrance point. In this configuration every client is an entrance point. Summarized for network users, all one needs to know about is the destination IP address.

⁷Specifically for Tor it is true that running a hidden service does not require it to be a transit node which eliminates the headache of attracting huge amounts of traffic. This is of high importance for users with low bandwidth Internet connectivity.

3 Installation And Configuration

The default application for OnionCat is to create an anonymous VPN which is publicly accessible. It enables users to take part in the anonymous network. Once being a participant one could either use the network's services or provide ones own services or both. A prerequisite is to have a network address. The addressing method basically was discussed in 2.2. Thus, we first need to install and configure the anonymizer and run a hidden service. The following explanations refer to Tor as an anonymizing transport network, because OnionCat was originally developed for Tor and it is known to run stable with it.

3.1 Install and Configure Tor

To install Tor there are basically two ways: install it with a package manager or compile and install it from source. The first solution is probably the easiest way and usually suits most users' requirements. To build from source gives a little bit more flexibility in fine tuning several build options. Details on package installations and package mirrors should be looked up on the operating system's or distribution's main sites.

For Windows and MacOS X you should follow the links on the download page of the Tor project: <http://www.torproject.org/easy-download.html.en>. For both OSes *Vidalia* (www.vidalia-project.org) is installed together with Tor. Vidalia is a configuration and control GUI for Tor which is also available for Linux.

After successful installation we need to add a *hidden service*. This is done by either editing the Tor configuration file `torrc`, or by adding it with Vidalia. The latter results in Vidalia editing the configuration file of Tor. The configuration file is usually located in `/etc/tor/torrc` or `/usr/local/etc/tor/torrc`. On Windows it is located in `C:\Documents and Settings\\Vidalia\torrc`. Add the following two lines to the configuration file:

```
HiddenServiceDir /var/lib/tor/hidden_service/  
HiddenServicePort 8060 127.0.0.1:8060
```

On Windows the full path may be omitted. The directory will be created in `C:\Documents and Settings\`. The `HiddenServiceDir` directive specifies the directory where to locate the private key for the hidden service. `HiddenServicePort` specifies that all TCP connections, which are dedicated to virtual destination port 8060 from within Tor, are forwarded to the local host (127.0.0.1) on TCP port 8060. This is the port that OnionCat listens to by default. The port numbers should not be changed unless you know exactly what you're doing.

Now start Tor but, **make sure that the system clock is correct** beforehand. Tor will then create a directory at the location specified by `HiddenServiceDir`, as well as put two files into it: `private_key` and `hostname`. The first one contains the private key associated with the local hidden service. If running a service for other users, a web service for example, it is a good idea to backup this key to a safe place. It is with this key that a specific hidden service is uniquely identified. If the machine crashes and all data is lost, the hidden service can be recovered by copying the backed up key to the hidden service directory on the new machine.

The file `hostname` contains the hostname which is used by the Tor network to lookup and connect to this hidden service. It is the onion-URL. Look into the file. It contains a string like `a5ccbdkubbr2jlcq.onion`. Vidalia will display the hostname in "Provided Hidden Services" field in the "Services" settings window. You will need this hostname for OnionCat setup as explained below. Have a look at the log file to see if Tor is working. If using Vidalia, then just click on "Message Log".

If Tor works correctly it will say “Tor has successfully opened a circuit. Looks like client functionality is working.”. Note that Tor may need some time (a few minutes) to boot.

3.2 Installing OnionCat

Now, after successful installation of Tor, we can run OnionCat. As long as there are no packages⁸ OnionCat must be built from source. The steps are

1. Prepare build environment.
2. Build and install OnionCat.
3. Configure and test OnionCat.

On Unix-like OSes step 1 is not very difficult and it is most likely to be already setup. All that OnionCat needs is a C compiler, usually GNU gcc and the GNU make utility. On Windows we also need those two programs. Before this can be done, it is necessary to create a POSIX-like environment. This is done with Cygwin (www.cygwin.com). Using Cygwin is more difficult, hence, I will explain it in a separate Section 3.3. If you are going to install OnionCat on Windows read Section 3.3 before.

Download an OnionCat source tarball from www.cypherpunk.at/ocat/download/. Untar it. Change to the directory and configure it as described on that page. Now you should be able to run OnionCat by typing `ocat`.

3.3 Installing Cygwin on Windows

To run OnionCat on Windows create a POSIX-like environment. Go to www.cygwin.com and download and run the Cygwin installer. It will ask some questions, but click continue until you reach the package selection menu and select “gcc: C-Compiler” and “make: GNU 'make' utility”. Both are found in the “devel” section of the package selection window. Continue with the installation process until finished.

OnionCat is IPv6-based but Cygwin unfortunately does not support IPv6 at the current stage of development. But luckily there is an IPv6 patch available at win6.jp/Cygwin/ [9] by Jun-ya Kato. Download and install it as described on that page.

The next step is to install the TAP driver. This is a virtual network interface, usually called a *tunnel device*. This is the virtual layer 3 interface for Windows. Go to www.openvpn.org and download the OpenVPN Windows installer. To run OnionCat, OpenVPN itself is not necessary but the installer contains the TAP driver which was developed by the OpenVPN project. It is licensed under GPL version 2 with some additions. After downloading, execute the installer. It will display an options menu where you can un-select everything except the TAP driver. Continue with installation.

After successful installation click on the Cygwin icon on the desktop and continue reading at Section 3.2.

3.4 Configuring OnionCat

To configure OnionCat for its primary intention as client for the open anonymous network, we need the onion-URL which is located in the `hostname` file in the hidden service's directory

⁸The package building process can be very time consuming. There are already packages in preparation for FreeBSD, OpenBSD, Debian and Ubuntu Linux and MacOS X.

(see Section 3.1). When running OnionCat the first time you probably should run OnionCat in foreground to make sure that everything works correctly.⁹

In the shell, run OnionCat as root with the command `ocat -B <your_onion_url>`. OnionCat will produce some output. There might be errors like “select encountered error: "Interrupted system call", restarting”. As long as this just happens during startup it can safely be ignored. There might also be the warning “can't get information for user "tor": "user not found", defaulting to uid 65534” which can also be ignored.

Now check if everything is configured correctly. Issue the command `ifconfig`. It lists several stanzas. There should be a stanza for every registered network device. One should read `tun0` and have an IPv6 address assign. If the `tun0` stanza exists but has no IPv6 address assigned OnionCat may have failed assigning the address. In some very rare cases this may happen for a currently unknown reason. In that case assign the address manually. In order to do this, issue the command `ocat -i <your_onion_url>`. It should return the IPv6 address associated with your onion-URL. Now configure the address with `ifconfig`. Lookup the correct syntax of `ifconfig` in the appropriate man page.

Now check the IPv6 routing table: `netstat -nr6`.¹⁰ It lists all entries of the kernel's IPv6 routing table and it should contain at least one entry: the OnionCat IPv6 prefix `fd87:d87e:eb43::/48` pointing to the tunnel device. If there is no such entry, which could happen in some very rare cases, add the route manually. Lookup the correct syntax in `route` man page.

4 Using The Global Anonymous Network

If everything is setup correctly as described in the previous Sections you should now be able to use the OnionCat global anonymous network. First try to ping one of the existing hidden OnionCat services. Currently there are a few services known to be permanently online.

- `dot.aio`¹¹ (`fd87:d87e:eb43:f683:64ac:73f9:61ac:9a00`) is a web-based service registration directory. It's intended to let OnionCat service providers register their service in order to be found by others. It's not required for a service to be registered but it enhances usability for new users. They can browse this page and lookup existing OnionCat services.
- `irc.onion.aio` (`fd87:d87e:eb43:2243:5f84:5b12:7bb5:bbc2`) basically is a *Internet Relay Chat* (IRC) server. IRC is based on the protocol definition of RFC1459 [5]. For a quick introduction have a look at Wikipedia at en.wikipedia.org/wiki/Internet_Relay_Chat. There is also a web-based audio stream (“OnionCat Radio”) available on port 1337 at this same address and a new, web-based community platform called “*Whose Space?*”.
- `ping.onion.aio` (`fd87:d87e:eb43:f947:ad24:ec81:8abe:753e`) currently does nothing than just respond to echo requests.
- `mail.onion.aio` (`fd87:d87e:eb43:744:208d:5408:63a4:ac4f`) is a combined SMTP/POP3 server. It accepts mails on port 25 for recipients of domain `onion.aio` (e.g. `eagle@onion.aio` which is my email address). Users can fetch mail using the POP3 protocol on port 110. Mailboxes need to be registered in advance. Unfortunately there is currently no automatic registration service available. Post an email to `onionmail@onion.aio` on this server in order to get an account. Note that this is completely anonymous as long as you don't send personal information across with your email.

⁹At the time of writing this document (March 8, 2009) OnionCat will fail on Windows if not run in the foreground.

¹⁰The digit '6' might be omitted on some OSes.

¹¹The term “aio” refers to *anonymous Internet overlay*.

Try to ping one of those hosts by issuing the `ping6` command which is ping for IPv6. After some time it will respond and list the *round trip time* (RTT) in the right most column. Be patient, Tor may need up to one minute for the first time it connects to a hidden service. After the connection is setup the RTT will be between 0.5 and 10 seconds. Currently there are many efforts within the Tor project to improve connection setup time and RTT in respect to hidden services.

If everything worked until now, you can start using the network as you do with Internet with the sole exception that there is no DNS. It requires the use of plain IP addresses instead of domain names. As long as there's no feasible DNS solution, the host names can be registered locally. On all Unix-like OSes this is easily done by just putting IP address hostname pairs into the file `/etc/hosts`. This is possible even on Windows. The file is usually located at `C:\WINDOWS\SYSTEM32\drivers\etc\hosts`.

5 Conclusion

OnionCat is an add-on for anonymizing networks like Tor. It interfaces with the IP routing process of the kernel and creates a VPN on top of anonymizing networks. Within this document I promoted the primary development idea of OnionCat, described the basic concepts, and gave a brief installation, configuration, and usage guide for OnionCat. Additionally there are already some services available which were presented.

The OnionCat software is still in heavy development and may not work on every system without further intervention, but we have managed to port it to major operating systems like Windows XP and MacOS X.

One problem still not solved sufficiently is the DNS problem, specifically how to resolve hostnames to OnionCat IPv6 addresses. As a matter of course they could be stored within Internet DNS but this most likely will leak information. With that in mind it is not a good idea. A feasible solution might be to setup a private DNS within OnionCat. That is basically no problem but it would require a user to have some kind of Split-DNS service running locally. This scenario would lead to additional installation effort.

References

- [1] Bernhard R. Fischer. OnionCat - A Tor-based Anonymous VPN. In *Proceedings of the 25th Chaos Communication Congress*. Chaos Computer Club, December 2008.
- [2] Bernhard R. Fischer. OnionCat Project Site. <http://www.cypherpunk.at/onioncat/>, 2009.
- [3] R. Hinden and B. Haberman. Unique local IPv6 unicast addresses. RFC 4193, October 2005.
- [4] Dave Kosiur. *Virtual Private Networks*. John Wiley & Sons, Inc., 1998.
- [5] J. Oikarinen and D. Reed. RFC 1459: Internet Relay Chat Protocol, May 1993. Status: EXPERIMENTAL.
- [6] The Tor Project. Tor Rendezvous Specification. <http://www.torproject.org/svn/-trunk/doc/spec/rend-spec.txt>, 2008.
- [7] Andrew S. Tanenbaum. *Computer Networks, Fourth Edition*. Prentice Hall PTR, August 2002.
- [8] The Tor Project. <http://www.torproject.org/>.
- [9] Jun ya Kato. Cygwin IPv6 Extension Patch. <http://win6.jp/Cygwin/>, 2008.