2007 Circumvention Landscape Report: Methods, Uses, and Tools

2007 Circumvention Landscape Report: Methods, Uses, and Tools

Hal Roberts, Ethan Zuckerman, and John Palfrey[†]

The Berkman Center for Internet & Society at Harvard University

March 2009

† Each co-author is affiliated with the Berkman Center for Internet & Society at Harvard University. Hal Roberts is a researcher; Ethan Zuckerman is a fellow; and John Palfrey is a faculty co-director of the Berkman Center. Hivos and the Oak Foundation provided funding for this study, for which the authors are grateful.

Disclosure: The Berkman Center for Internet & Society at Harvard University has a long-standing academic partnership with the Citizen Lab of the Munk Centre for International Studies at the University of Toronto. Together, the Berkman Center and the Citizen Lab, along with other partners, run the OpenNet Initiative (the "ONI", online at http://opennet.net/), a not-for-profit academic consortium that studies and documents Internet censorship around the world. John Palfrey, one of the co-authors of this paper, is one of four principal investigators of the OpenNet Initiative. Separate from its work on the ONI, the Citizen Lab is also the developer of one of the tools discussed in this study, called Psiphon. The Berkman Center shares funders in several respects with many of the makers of circumvention tools. For instance, Internews Network (USA) supports circumvention research at the Berkman Center as well as work on Tor and Psiphon, and the U.S. State Department has funded DynaWeb, UltraReach, and Anonymizer as well as work at the Berkman Center. Neither the Berkman Center nor the authors of this paper have any involvement in any of the technologies or entities described in this report, including but not limited to Psiphon, Tor, DynaWeb, UltraReach, and Anonymizer. Neither the Berkman Center nor the authors stand to profit in any way financially from the success of any general technology or specific tool described in this report.

2007 Circumvention Landscape Report: Methods, Uses, and Tools

1. Preface

Over the past fifteen years, we have collectively transformed the Internet from an academic curiosity to an essential piece of the global communications infrastructure, an engine for commerce and a major platform for free expression. As states have connected their citizenry to the Internet, leaders of some of those states have discovered speech they find threatening, unsettling or counter to national values and norms. In turn, some states have taken measures to limit citizen access to the Internet and prohibited the publication of various types of information online.

In 2002, independent research teams at the universities of Cambridge, Harvard, and Toronto began to document the extent to which the Internet is not exactly a safe haven for activists -- and neither as free nor as open -- as optimists had claimed. More or less concurrently, these research teams began to probe the ways in which certain governments were restricting access to information online that they deemed to be subversive. An early report by Jonathan Zittrain and Ben Edelman at Harvard Law School, for instance, documented efforts by the Saudi Arabian government's Internet Services Unit to prevent its citizens from accessing thousands of web pages (http://cyber.law.harvard.edu/filtering/saudiarabia/). In the past five years, government filtering of the Internet has become endemic. These independent researchers banded together to form The OpenNet Initiative, a research project which joins Zittrain as well as Ron Deibert (Toronto), John Palfrey (HLS), and Rafal Rohozinski (Cambridge). The OpenNet Initiative presently tracks filtering in more than forty states (http://opennet.net/research).

Some states filter just a few websites, such as Singapore; others, such as Iran, filter tens of thousands. China has implemented a vast and complex system which involves filtering international websites as well as pervasive censorship by Chinese Internet companies of content published within the nation (http://www.rsf.org/article.php3?id_article=23924). Other nations have chosen simpler approaches. When the government of Myanmar discovered that citizens were using the Internet to report on the Saffron Revolution, they shut off access to the Internet, a crude but extremely effective form of Internet filtering (http://cyber.law.harvard.edu/home/home?wid=10&func=viewSubmission&sid=2999).

Tools to circumvent Internet filtering predate government filtering of the Internet. Seventeen-year-old high school student Bennett Hazelton founded Peacefire.org in 1996 to protest the use of Internet filtering software by US schools and to help high school students find tools to circumvent these filters (http://www.aclu.org/privacy/speech/15538lgl19970319.html, http://peacefire.org). Even in those early days of the Internet's popularity, hundreds of simple websites – proxies – existed to help users evade Internet filters.

As filtering has become more pervasive, a number of efforts have emerged to produce circumvention tools and distribute them to potential users. These tools are likely used only by a few million people, mostly from states where the Internet is filtered by the government in concert with private parties, to get access to filtered content. Some efforts have become profitable businesses; others are labors of love financed by their creators. Several circumvention efforts have built, maintained and marketed their tools with support from government and foundation funders who focus on human rights and free expression.

It's easy to understand why governments and human rights funders would be interested in supporting censorship circumvention tools. As discourse shifts from traditional media to new participatory media, the ability to access and create online information becomes equivalent to the ability to read, listen and speak freely. There's a long history of the importance of free media in closed societies, including the role of Radio Free Europe and Radio Liberty in ending state communism in the Soviet Union.

Because Internet filtering and circumvention is a highly technical field, there is a concern that funding may be supporting tools that do not work as promised, or that duplicate the functionality of existing

tools. The Berkman Center was invited by Hivos and The Oak Foundation to conduct an evaluation of the major circumvention tools, with a focus on their utility, possible areas for improvement and long-term sustainability. This report is one of the end products of that research.

Under the leadership of Hal Roberts, a senior researcher at the Berkman Center, we have produced a report on the functionality of ten circumvention tools, including lab tests and field tests in South Korea, Vietnam and two cities in China. The report describes the existing landscape of circumvention methods, tools, and uses. The report also identifies how the tools interact with different circumvention uses. The report consists of theoretical and tool analysis sections. The theoretical section contains a narrative description of existing filtering and circumvention methods and some usage modes for the circumvention tools. The tool analysis section includes a separate report on each of the following tools: Anonymizer Anonymous Surfing, Anonymizer China, DynaWeb FreeGate, UltraReach, Circumventor/CGIProxy, Psiphon, JAP, Tor, Coral, Hamachi. Each of these tool reports begins with a summary analysis of the strengths and weaknesses of the tool and is followed by the results of the in lab and in country tests and a detailed analysis of the tool and project.

Work began on this report almost two years ago; today, at the date of publication in early 2009, some of the information presented here is out of date. This report is certainly no longer a comprehensive overview of the field – important new tools have been introduced that we have not yet evaluated. The Berkman Center requested permission from the study's sponsors to release this report to the general public because we believe the methodology pioneered here will be useful for any future projects evaluating the utility of these tools and we think the broad findings about the methods and approaches of filtering and circumvention hold. We encourage other researchers to borrow freely from our methodology (described in detail in Appendix II of this report) and welcome anyone who wants to replicate, expand and repeat this work to let us know if we can be of assistance, technically or otherwise.

Looking to the future, we now think it likely that simple web proxies represent at least as great if not greater proportion of circumvention tool usage as do the more sophisticated tools included in this report. An assumption of this report was that only users at the margins would rely on simple proxies because of the trouble of constantly finding new proxies as old ones were blocked by countries. We now have some evidence that that assumption is false (both that users are not using the simple proxies and that filtering countries are blocking simple proxies quickly). We think that any followup studies of the technology of circumvention tools, like that in the following study, should rely on a better understanding of which tools are actually used, how they are used, and why people use them.

A. Summary of Findings

We were reassured to discover that most tools function as intended. They allow users to circumvent Internet censorship, even in countries like China and Vietnam, which use sophisticated technology to filter. However, we discovered that all tools slow down access to the Internet, that most tools featured serious security holes, and that some tools were extremely difficult for a novice Internet user to use.

Because the study was conducted in cooperation with tool authors, they have been alerted by our team to security holes, and several developers have already patched holes identified in the study. The Berkman Center also hosted a meeting in Oxford, UK that allowed the developers of all these tools to meet and share ideas about the evolution of their systems. This dialog between toolmakers may prove to be one of the most useful aspects of the study.

The Berkman report does not offer a summary of recommendations of which tools best serve users in circumventing Internet filtering. Different users will find that different tools meet their particular needs, depending on the priority and definition of the criteria below. However, at the end of each tool report,

we evaluate the tool based on six criteria:

- Utility (Does it retrieve filtered web pages? How fast is it compared to unmediated browsing? How well does/will performance scale?)
- Usability (How easy it for novice users to learn? How easy is it to use?)
- Security (How well does the tool protect the user from surveillance at various points in the network?)
- Promotion and marketing of the tool (How well is knowledge of the tool pushed out to users in filtered countries?)
- Fiscal sustainability of the project (To what degree is the project on a path toward fiscal self sustainability?)
- Openness (Is the code open source? Are the developers approachable and forthright about their techniques and future plans?)

Both the definition of and the weights attached to each criteria vary widely depending on subjective priorities of the evaluation (is openness a strength or a liability? is funding through foundation and government grants sustainable? is security more important than usability?). Furthermore, these evaluations are based on work that is now two years old and so does not necessarily reflect the current state of these tools. Most of the tool developers have submitted responses that address both the full tool reports and these evaluations and discuss recent developments with the tools. Some of those responses express strong disagreement with these evaluations. Many of the problems identified in the evaluations and in the detailed project reports have been fixed, and undoubtedly new problems have arisen for many of the tools. Because the testing is now out of date, we do not include numerical rankings or scores based on the evaluations.

Among other disagreements about the criteria for evaluation, some of the tool developers consider secrecy about their implementation details to be a strength rather than a liability. They argue that secrecy allows them to protect their tools (and users) from filtering and snooping. From a technical point of view, there is a well established, contentious debate among software developers about whether secrecy about implementation details is a robust strategy for security. We hold with those that believe that such an approach amounts to security through obscurity and generally leads to less secure software over the long term. A determined opponent can usually reverse engineer the working of any public system. If the system relies on secrecy for security, that opponent is more likely to be able to find a vulnerability in the system once she has reverse engineered it. Our preferred approach is to design a system that is secure even if the attacker knows how it works. An open approach allows the community to audit the system rather than leaving the auditing to the attackers who are willing to reverse engineer it, forcing developers to fix vulnerabilities rather than hide them. This mode of developing in the open is used for most major security standards, such as the https and ssl standards that provide widespread encryption for web browsing.

From a social point of view, we think openness is also important to help users (and researchers and funders) understand what kinds of trust they are investing in a given tool. Circumvention tools function as virtual ISPs, rerouting the traffic of their users around blocking filters and in the process gaining the same level of access to that traffic as the users' local ISPs. The access to that data represents a large investment of trust in the tools: trust that the tools will faithfully transmit the data and will not share, lose, or sell data about their users to undesired third parties. Openness on the part of the circumvention tools about how their tools work and what they are doing with the sensitive user data they control is

critical to helping users make decisions about which tools to use and how to use them.

The tools differ in their approaches to other technical problems:

- *Centralized vs. Peer-to-Peer Performance*. Some tools (Anonymizer, UltraReach, DynaWeb) use a set of centralized servers as proxies. Other tools (Tor, JAP, Circumventor, Psiphon) rely on peer-to-peer technology to allow volunteers to use their own computers as proxies. Centralized proxy systems incur linear increases in operating cost as the number of users grow but currently perform significantly better than peer-to-peer systems and can also improve performance straigthforwardly with increased funding per user. Peer-to-peer systems incur very little incremental operating cost as the number of users grow, but their poor performance is generally difficult to improve even with increased funding.
- *Trust*. In most proxy systems, the person or organization managing the proxy has the ability to monitor what content a user is looking at, and can filter what the user accesses. Centralized systems require the user to trust the operator of the centralized servers. Tor and JAP attempt to address this problem through sophisticated routing techniques, allowing the user to trust the architecture of multiple proxies rather than any one proxy operator (though JAP's default implementation is badly broken and Tor's leaves careless users open to attack). Psiphon and Circumventor require the user to trust the operator of the particular node accessed by the user. No one of these models is the most secure; each is best suited for a different set of use cases.
- *Discovery and Blocking Resistance*. Countries that filter the net tend to block access to Internet circumvention tools as well. Anonymizer, for instance, is blocked in most filtered countries. To solve this problem, tools use sophisticated methods to distribute the locations of new proxies, including mailing lists, chat systems and bulletin boards. Peer-to-peer tools like Psiphon rely on their peer to peer structure to limit blocking – because very few people use a given Psiphon node, it is less likely to be blocked. Centralized tools like UltraReach rely on sophisticated anti-blocking technologies to outwit filterers.

It's worth nothing that none of the developers we spoke to, individually and at our convening, foresaw a "silver bullet" that would "solve" the problem of filtering circumvention. All the tools rely, to a certain degree, on providing more proxies than the authorities can block and continuing to create new proxies as old ones are blocked. The preferred technical term for this strategy is "Whack a Mole," a reference to an American fairground game, and while none of the developers are thrilled about an ongoing arms race with censors, some are taking complex steps to ensure they'll have many more proxies than the government can shut down. We are confident that the tool developers will for the most part keep ahead of the governments' blocking efforts.

B. Implications of the Study

Most of the tools reviewed in the study require additional development (and concomitant funding) either to improve the design of the tools, or to help the tools scale to accommodate more users. Improving UltraReach, for instance, would involve supporting the tool's founder in purchasing more servers and bandwidth, or purchasing the founder's code and operating it on dedicated server farms, or both. Scaling Psiphon, on the other hand, might involve improving the existing code base to support a caching architecture.

We believe future development of circumvention tools should not focus on a single path or strategy. Filtering governments would find it easier to combat that one tool, even if well funded, than to combat a diverse set of tools. Moreover, no one tool is the best for all use cases, so the user base benefits from a

variety of tools that meet a variety of needs.

Extrapolating from the usage figures privately reported to us by tool developers (numbers that are basically unverifiable), we guess that the number of people using circumvention tools is around two to five million users worldwide. This number is quite high in absolute terms but quite low relative to the total number of filtered Internet users (China alone has over two hundred million Internet users). Even accepting likely high end estimates of the project developers, we believe that less than two percent of all filtered Internet users use circumvention tools.

This study points out a number of likely reasons for the low uptake of the circumvention tools:

- The performance of the tools is quite poor, ranging from marginally usable for the best tools to painfully bad for most of the rest (many tools required longer than the maximum allotted four minutes to load many sites).
- Though the tools have mostly remained functional despite blocking efforts by filtering countries, the countries have had very good success blocking the circumvention tools from appearing on search sites like Google. So circumvention tool projects have to rely on alternative, non-web methods of distribution. Most of these methods use technologies like instant messaging or email but are ultimately a technically-enhanced form of inefficient word of mouth.
- All circumvention tools require either that the user install additional software or that the user know someone in an unfiltered country who will install additional software. Even though most of the tools work hard to make installation of this software as easy as possible, any installation is a significant barrier of entry for many users, and users may be deterred by the unprofessional appearance of many tools.

We urge everyone involved with Internet filtering circumvention to think realistically about the appropriate goal for the number of users who will use circumvention tools. Even though additional work and investment will improve these tools, the problems the tools face become much harder to battle as the tools improve. Performance is a particularly difficult problem to solve entirely. Proxying a connection requires double the bandwidth of the original, direct connection (the bandwidth to get from A to B has to be reproduced first from A to C and then from C to B). So providing every filtered Internet user a proxied connection would require double the bandwidth used by every one of those filtered users. Even though there is room for much improvement in the distribution and installation of these tools, there is a fundamental ceiling in both of these areas. Even if circumvention tools were perfectly usable, we think that a majority of people in many filtering countries would choose not to use any of them due to social and political forces that discourage dissent and political barriers (including the risk of imprisonment).

Taking all of these limitations into account, we believe that universal usage of circumvention technology is an unrealistic goal, but usage by a critical 5-10% of influential users (or, say, those most at risk of persecution) in target countries is reachable with sufficient support. We hope that these early adopters will serve a key role in their communities, both encouraging wider use of the tools and pushing the information gained through their unfiltered connection to their wider communities.

C. Future Research

We believe the methods we advance in this paper to study the effectiveness of circumvention tools should be applied to a broader set of tools and repeated on an annual basis. We strongly advocate that any work in this field include field testing as well as lab testing, as our statistics showed that performance in censored countries was difficult to reproduce in lab testing. We believe a regular process

to test tools in censored nations would provide a useful data set for tool developers, governments, companies and foundations that wish to support free speech, and end-users.

We also see potential for future research to address the following questions:

- Which circumvention tools do people user? How do users discover the existence of circumvention tools? What factors lead a user to choose one circumvention tool over another?
- What is the relationship between circumvention and surveillance? What sorts of risks are various types of users in various countries taking when using tools to circumvent Internet restrictions? How do users view those risks?
- Will the spread of VPN client capabilities into modern operating systems increase the utility of VPN-based strategies for circumvention?
- Will publisher-side strategies (content mirroring, etc.) become more important than circumvention strategies in the future?
- Are there functional circumvention strategies that avoid the bandwidth constraints of the proxy strategies studied here?

2. Theoretical Landscape

The landscapes of filtering and circumvention methods are both relatively simple, though the implementation details of both filters and circumvention tools have dramatic effects on their effectiveness. The nature of the underlying Internet Protocol leaves room for only a handful of fundamental ways to filter content and to avoid that filtering.

A basic understanding of the fundamental workings of the Internet, including most importantly the Internet protocol (IP), is helpful when reading this report. A brief but good demonstration of how data is transferred via IP is here:

http://www.pbs.org/opb/nerds2.0.1/geek_glossary/packet_switching_flash.html

A more detailed description of Internet architecture is here:

http://cyber.law.harvard.edu/digitaldemocracy/internetarchitecture.html

There are no absolutely effective filtering or circumvention methods. The battle between those who would filter and those who would circumvent that filtering turns on the amount of resources each side expends and the effectiveness of those resources. So there are always ways to filter a piece of information if the filterer is willing expend enough resources, and there are always ways to circumvent a filter if the circumventor is willing to expend enough resources. The goal is to find tools that meet users' needs cheaply, and require disproportionate resources to block.

Circumvention is a large topic that reaches deeply into a number of other large topics, including filtering, privacy, surveillance, and content neutrality. This report focuses on the effectiveness of a representative range of circumvention tools against the currently dominant router based filtering methods. The report addresses filtering only insofar as is helpful to understand the effectiveness of the circumvention tools. The report addresses surveillance only insofar as it is relevant to the privacy features of the various circumvention tools, as used for circumvention.

For this report, we consider content to be value neutral. The fact that circumvention users can access not only information widely considered to be good, such as political dissent, but also information widely considered to be bad, such as pornography, is not considered except to mention which tools try to make these judgments themselves through their own internal filters.

A. Social, Political, and Technical Filtering

Filtering consists of social, political, and technical methods, all of which interact in ways that can strongly reinforce one another. In most filtered countries, the social pressure not to visit forbidden information is the most effective form of filtering (and applies equally well to on- and off-line information). Unfiltered Internet access increases the diversity of information and opinions users see. This increased diversity not only provides users with the new information and viewpoints but also, in itself, exerts a counteracting social pressure in favor of free information. Unfortunately, all of the tools included in this report require explicit, aware actions, discouraging users from ever entering the positive feedback loop of free information. The combination of the social pressure to filter information and the initial hurdle of using circumvention technology reinforce one another.

When social pressure fails, political pressures apply. In practice, the arrest of users for accessing blocked information is rare, but laws regulate at a higher level as well. For example, China has implemented regulations on Internet cafes that place Internet cafe users under increasingly strong surveillance, including requiring credentialing for access to the Internet. These sorts of measures both

make clear the threat of the application of political force and reinforce the social pressures to filter. Also, laws are most likely to be enforced against known political dissidents, who are some of the key early users of circumvention technology.

Technical filters are both the first and last line of defense against the dissemination of unwanted information. In the first place, technology allows the filtering country easily to designate which pieces of information it wants its citizens to avoid; this mere designation enables the social pressures to function. Technical filters only need to apply force (by actually blocking rather than merely designating offensive material) when social and political forces fail.

As this report describes, circumvention technology is currently effective enough that avoiding technical filters is relatively easy. It is the combination of social, political, and technical filtering methods that proves very difficult to break through and explains why circumvention tool use remains very small in proportion to the total number of Internet users in filtered countries (see usage metrics of individual tools for specific numbers). This report does not directly address the social and political issues of filtering but does address how these issues interact with technical filtering and circumvention tools. For instance, some tools emphasize privacy over usability. This emphasis encourages users who are fearful of the political repercussions of using circumvention but discourages more casual users who want a seamless browsing experience.

For a much fuller understanding of filtering technology both as theory and as implemented by countries in practice, consult the work of the Open Net Initiative project (http://www.opennet.net). The Open Net Initiative project compiles detailed, ongoing surveys of what and how dozens of countries filter the Internet.

B. Technical Filtering

Almost all data on the Internet is exchanged in a client server model, meaning simply that a client requests some data over the network that a server then returns over the same network. A filter can operate in three places in such a system – on the client requesting the data, on the server returning the data, or on the network between, during either the request or the response. Brief descriptions of server and client side filtering follow, but this report focuses for reasons described below on filtering on the network between the client and server.

Perhaps the most straightforward way to filter the Internet is to install software on a client machine that prevents the client from requesting offensive information in the first place. Many Internet accessible computer labs, including those at schools and libraries, use this sort of filtering because they have easy control over a small number of computers to be filtered.

There are two reasons why client filtering is very difficult on a national level. First, there are many more clients than servers or routers on the Internet. To implement comprehensive client filtering over a large network requires the installation of filtering software on all of the client computers on that network. In China, large scale client side filtering would require installing and maintaining software on around 150 million individual Internet accessible computers, an extremely difficult and expensive task even if all filtered users were supportive of the effort.

The second and more important problem with client side filtering is that personal computers are built as general purpose computing machines, so it is very difficult to impose strong limits on a single functionality. For an easy example, filtering software installed inside a browser is easily circumvented by installing an alternative browser. To prevent a user from installing an alternative browser requires monitoring installation of all software on the computer. This sort of monitoring on a national level

would require prohibitive administration costs and would drastically limit the functionality of all of the monitored computers.

As with client side filtering, the technology of server side filtering is simple in practice – you need either remove the offending content from the server or prevent the server from sending a response with the offending content. In practice, most Internet servers are located in non-filtering countries, so filtering countries need the participation of either the hosting content owner or the hosting country. Requests to content owners to block material are sometimes successful, as in the well published blocking of pro-Nazi material by Yahoo in France and the self-censoring of search results by Google, Yahoo, and others in China. In the face of such server blocking, though, the blocked material inevitably moves downstream to smaller entities, often entities that are explicitly hostile to the filtering country and will not participate with the filtering country. Human Rights Watch will never filter its own reports on China.

Another possible mode of server side filtering is to engage in hostile attacks on offending servers, for example via machine intrusions or denial of service attacks. Such attacks are expensive, however, and only temporarily effective and so have not been used widely to date.

The ideal Internet is a chaotically connected web of equal, interconnected local networks with no one network link more important than another. In practice, Internet traffic aggregates onto a small number of huge backbone data pipes and then disaggregates back out to the individual local networks. Filtering the vast majority of traffic on a large network requires only filtering the traffic on this small number of pipes (only regional traffic that never hits a backbone avoids such filtering). Specialized computers called routers control all Internet traffic, including that on the backbones. A router has to inspect at least the destination of every bit of data that passes through the network to perform its necessary work, making it highly efficient at at least one of the most important tasks required for a filtering machine.

Backbone routers handle a tremendous amount of data, making expensive any significant computations on all of the data. However, computing power has outpaced network performance sufficiently that filtering routers are able to perform increasingly sophisticated computations on all data on country level backbones, including searching for a large list of keywords.

All filtering countries use a combination of social, political, and technical filtering methods, and most filtering countries use a combination of technical filtering methods including client, server, and router filtering. Many countries use only a single router filtering method, such as IP blocking or DNS blocking. China has the most sophisticated technical filtering infrastructure, consisting of overlapping implementations of IP blocking, DNS blocking, and Keyword blocking. For the best available information on the filtering infrastructures of specific countries, see the country reports and bulletins published by the Open Net Initiative.

I. IP Block Filters

The most straightforward way to block traffic on a router is to block the IP addresses of servers hosting offensive material. The IP address is the number used to uniquely identify every computer (server or client) on the Internet, so blocking the IP address of a given machine makes it inaccessible. A router already needs to examine the destination IP of every bit of data it handles, so the only extra cost of filtering based on the destination (or origin) IP is the cost to lookup the destination IP address in a table of blocked IP addresses. Such lookups require relatively little computing power even at the scale required for a country level backbone router, making IP blocking a cheap method of filtering traffic on a large scale.

There is no way to access a blocked IP address directly through a filtering router, and the number of backbones (and therefore controlling routers) leading out of any filtering country is small enough that it is easy to add IP block filtering to every backbone leading out of a filtering country.

IP blocking is very blunt, since it can block only whole addresses (usually meaning whole servers) rather than particular bits of content. For example, if a particular news site publishes an offending story, the IP block filter must either block the entire news site or allow access to the entire news site. Even worse, many web hosting companies combine hundreds of web sites onto a single IP address. So blocking a single offending web site may block not only a single entire site but hundreds of other, not substantively related sites as well. Such over blocking has been well documented by the Open Net Initiative and others in many reports.

II. DNS Block Filters

Another way to filter traffic on the network is to block the lookup of DNS names of offensive sites. The DNS name is the plain text name generally used by end users to lookup a site (e.g., google.com). These DNS names are translated into IP addresses by DNS servers. To block specific DNS name lookups requires only removing those DNS zones from all of a country's DNS servers. There are more DNS servers than backbone routers, but the number is still relatively small, and the work of maintaining DNS block lists can be outsourced to individual ISPs simply by distributing the list to the participating ISPs.

Like IP blocking, DNS blocking suffers from over blocking because it can only block entire sites. The name "hrw.org" must be blocked entire, rather than individual pages accessed at that name. However, DNS blocking does not suffer from over blocking of sites co-hosted on the same IP address – if two sites aaa.com and bbb.com are hosted at the same IP address, DNS blocking can block aaa.com but not bbb.com, whereas IP blocking must block both aaa.com and bbb.com to block either.

DNS blocking is very simple to circumvent by simply configuring the client computer to use a DNS server in a non-filtered country (or even a non-filtered DNS server inside the filtering country). There are many public, easily accessible, non-filtered DNS servers that can be used for this purpose.

III. Keyword Block Filters

Keyword block filtering examines the content of the traffic, rather than merely the destination and can examine content in any part of a request, including a specific url in a web request or an offending keyword in an email. As such, keyword blocks can be much more precise than IP or DNS blocks, blocking only specific pages with a complete url (http://hrw.org/china but not http://hrw.org/usa) or only specific pages with an offensive keyword ("taiwan independence"). Keyword based blocking can be used for any Internet application that uses text, including web pages, email, and most instant messaging.

Compared to IP blocking and DNS blocking, keyword blocking is very expensive. A keyword filter must examine the entire stream of content, rather than just the very short destination data, and must perform text searching on that much greater amount of data. That text searching operation is much more expensive than the simple table lookup operation required by IP filtering. Nonetheless, several countries, including China, have been shown to use keyword filtering on a large scale.

Keyword blocking can be circumvented by encrypting all data sent over a connection, since encryption renders content into a form the filter cannot read. So a request that is blocked by a keyword filter as an HTTP request will not be blocked as an HTTPS request. Likewise, non-text data is generally impossible to filter by simple keyword filtering, since most such data must be interpreted as entire files, rather than as individual packets (packets are the discrete blocks of content that streams of Internet data on the Internet are broken into). Examining a stream of separate packets together requires stateful traffic

analysis, which is prohibitively expensive for reasons described below.

IV. Stateful Traffic Analysis

Keyword based filtering, though expensive, only requires the router to examine each packet as it passes through and to make a decision based on the single packet whether to allow the connection to continue. The router can make much more sophisticated decisions by looking at a stream of packets in aggregate, but such filtering requires that the router store information about each stream over a period of time (the stream's state). For example, a stateful filter might watch for packets containing parts of image files and correlate those packets into complete images. The filter could examine the whole image for certain characteristics, perhaps large areas of flesh tones indicating nude skin. Or a stateful filter might keep track of which website addresses a given client requests over time in order to find clients who make all their connections to the same websites and are therefore likely using a proxy. Or a stateful filter might correlate a number of related traffic signatures (connects constantly to a single address, is encrypted, and follows the request of a blocked IP address) to identify proxy users precisely and in real time.

Storing state about data streams is much more expensive than the comparatively simple operations required by keyword filtering since storing state involves performing extra write operations as well as performing much more complicated tests on the comparatively large amount of store information. No filtering countries are known to have deployed stateful traffic analysis on a large scale yet, though they might in the future as computing power becomes cheaper and circumvention becomes more common.

C. Circumvention Methods

All circumvention methods consist of hiding some combination of the destination and the content of each packet. Proxying conceals the destination by sending the data through a destination in an non-filtered country that send the request onto the ultimate destination and then sends the response back to the client. Encryption conceals the content. Many circumvention tools attempt to hide their traffic signatures as well, taking pains to avoid simple patterns like using a single, non-standard port or including easily identifiable keywords in non-encrypted traffic. The major differences between the methods used by the various circumvention tools lie in the methods used to proxy the data, in the methods for hosting the proxying servers, and in the techniques used for blocking resistance.

I. Proxying Methods

All of the below proxying methods will effectively circumvent IP and DNS blocking and will also circumvent keyword filtering when combined with encryption. Nonetheless, there are important functional differences between the various proxying methods.

a. HTTP Proxy

An HTTP proxy tool's security can be trusted as far as the operator of the proxy server can be trusted. HTTP proxy tools require either manual configuration of the browser or client side software that can configure the browser for the user. Once configured, an HTTP proxy tool allows the user transparently to use his normal browser interface. HTTP proxy systems benefit from the wide, non-circumvention related use of HTTP proxying technology.

HTTP proxying sends HTTP requests through an intermediate proxying server. A client connecting through an HTTP proxy sends exactly the same HTTP request to the proxy as it would send to the destination server unproxied. The HTTP proxy parses the HTTP request; sends its own HTTP request to the ultimate destination server; and then returns the response back to the proxy client. HTTP proxying is used widely for non-circumvention purposes to accelerate and even to filter web browsing on an

organizational and ISP level. HTTP proxying is well supported both by the HTTP standard and by a wide variety of highly optimized proxy servers, such as the freely available squid program.

There are a huge number of public HTTP proxies running on the Internet freely accessible to anyone for circumvention or other uses. Users can find these public servers via many different aggregation lists, such as http://www.samair.ru/proxy/. However, filtering countries read the same public proxy lists and block the IP addresses of most of these listed proxies. So users in filtered countries have to search manually and repeatedly for new proxies as the filtering countries constantly blocks existing proxies. More importantly, nothing is known about the hosts of the proxies found on these lists, so the user cannot trust the privacy or integrity of her requests (indeed filtering countries that want to snoop on circumvention users could and might do so by hosting public proxy servers). Also, public proxies generally do not provide encryption. Using a proxy without encryption is not only ineffective at circumventing keyword filters but also decreases privacy by transmitting HTTPS requests in clear text between the proxy and the user.

Circumvention tools that use HTTP proxying are mostly designed to avoid these problems. First, HTTP proxying circumvention tools automate the process of finding an unblocked HTTP proxy and configuring the browser to use it. In fact, the most sophisticated HTTP proxy circumvention tools fail over in real time when a given proxy server is blocked, so a user will never know when the tool has to switch to a new, unblocked proxy. Second, most circumvention tools that implement HTTP proxying host their own proxy servers, allowing the user to trust the proxy servers as much as she trusts the circumvention tool developers. Lastly, most HTTP proxy circumvention tools connect only to encrypted proxy servers, protecting connections against both keyword filtering and surveillance.

HTTP proxying is completely transparent to the end user once it has been configured (either manually or by a tool); this transparency allows the system to present the same user interface for direct and for proxied browsing sessions. However, HTTP proxying requires that user either manually configure his web browser or install software that will configure his web browser for him, both of which actions increase the cost of switching to a proxied browsing session and also leave detectable changes on the client computer.

b. CGI Proxy

A CGI proxy tool's security can be trusted as far as the operator of the proxy server can be trusted. CGI proxy tools require no manual configuration of the browser or client software installation, but they do require that the user use an alternative, potentially confusing browser interface within the existing browser. CGI proxy tools are not widely used for purposes other than circumvention and so do not benefit from a wide base of developers and users.

CGI proxying uses a script running on a web server to perform the proxying function. A CGI proxy client sends the requested url embedded within the data portion of an HTTP request to the CGI proxy server. The CGI proxy server pulls the ultimate destination information from the data embedded in the HTTP request, sends out its own HTTP request to the ultimate destination, and then returns the result to the proxy client.

As with HTTP proxy servers, there are a large number of public CGI proxy servers available on the Internet and aggregated into public proxy lists such as http://www.cgiproxylist.com/. And as with HTTP proxy servers, the use of these public CGI proxies suffers from continuous blocking, lack of encryption, and host trust problems. Unlike HTTP proxying, the only common uses for CGI proxying are anonymity and circumvention, so there is a much smaller user and development community for CGI proxy servers. As a result, CGI proxy servers are in some ways less well developed than HTTP proxy

servers; for example, many HTTP proxy servers include sophisticated caching support, which can hugely improve the performance and decrease the bandwidth use of a proxying server.

A CGI proxy server must be explicitly used through a web application. That web application must present an alternative user interface to the user, usually in the form of a second url address box, located under the browser address box but within the browser's html display window. The alternative user interface has great potential for confusing a novice user and will generally lead to user errors for even the most advanced users, since all users are heavily conditioned to typing urls into the main browser address box. Mistakenly typing an address into the main browser address box rather than the CGI proxy address box or mistakenly selecting a bookmark from the browser bookmark list will result in the leak of private data, if the requested url is itself private data (for example, a blocked url).

A CGI proxy client does not need to install any software on the web browser to use the CGI proxy. This lack of client installation decreases the time required to switch from non-proxied to proxied browsing and also leaves no installed software on the client computer. Also, the lack of client installation allows the use of CGI proxy servers on a locked down client computer on which the user cannot install software, such as a computer in an Internet cafe, a library, or a school.

c. IP Tunneling

An IP tunneling tool's security can be trusted as far as the operator of the tunneling host server can be trusted. IP tunneling tools require the installation of client side software that usually runs at a low level in the operating system. Once configured, an IP tunneling tool allows the user transparently to use his normal browser interface. IP tunneling benefits from the wide, non-circumvention related use of IP tunneling technology.

There are a variety of different technologies that allow tunneling of data at the IP level. Tunneling the IP streams themselves rather than HTTP requests and responses allows a tool to support circumvention for more than just web traffic. Some of the most commonly used technologies for IP tunneling include virtual private networks (VPNs), HTTP tunnels, and ssh tunnels. VPNs connect two or more remote networks into a single, virtual local network via an encrypted link over the Internet. VPNs are most commonly used to give remote users access to private intranets, rather than for circumvention proper. But the effect of a VPN is to give the user client a connection that originates from the VPN host rather than from the location of the client. Thus a client connecting to a VPN in a non-filtered country from a filtered country has access as if he is located in the non-filtered country. VPNs support not just web browsing but every other IP application, including email, instant messaging, and file sharing. Even though most VPN traffic is relatively easy to identify and therefore block, VPNs are overwhelmingly used for non-circumvention purposes. So blocking a given VPN implementation will have a high cost in terms of blocking mostly users who use the VPN for important, often commercial, non-circumvention uses.

An HTTP tunnel is essentially a VPN that works via standard HTTP (or HTTPS) requests and responses. The advantage of HTTP tunneling over standard VPN tunneling is that HTTP tunneling looks like normal HTTP traffic and is therefore more difficult to identify and block. HTTP tunneling even works over an HTTP proxy, which feature can be helpful when a user has access to the Internet only through a HTTP proxy (which is the case in many schools, libraries, companies, and other organizations). The cost of HTTP tunneling is that the performance is much worse than standard VPNs.

SSH (secure shell) is a program primarily used for secure remote access to the text console of a remote server. In addition to the secure text console support, ssh allows the creation of encrypted tunnels for single applications, allowing for instance a user to connect through a single ssh tunnel from a given

client computer to a remote host through an ssh host as if the traffic were coming from the ssh host. A separate ssh tunnel must be created for every remote computer to which the client needs to connect, so ssh tunnels are not very useful for general web browsing. They are instead generally used for applications that maintain a connection between a single client and a single remote computer over time, such as file sharing or remote desktop control. The biggest advantage of ssh tunnels over other sorts of IP tunnels is that they are very easy for an expert to setup and use versus the heavier VPN, HTTP, and other sorts of IP tunnels, which all require more configuration and setup.

d. Re-Routing

A re-routing tool transfers some of the trust in the system from the individual re-routing servers to the design of the network itself. Re-routing tools require the installation of client side software. Once configured, a re-routing tool allows the user transparently to use his normal browser interface. Re-routing tools are generally only useful for circumvention and anonymity and so do not benefit from widespread use and development.

All of the above proxying methods require that the client trust the proxying host, whether that host is an HTTP proxy, a CGI proxy, or a VPN server. Some users are not comfortable trusting even widely known and trusted circumvention projects, possibly because of doubts about the circumvention projects themselves but possibly because of doubts about what a host might be legally forced to do with private data. Re-routing systems decrease the amount of trust required of the proxy host by routing the data through a series of proxying servers, encrypting the data again at each proxy, so that a given proxy knows at most either where the traffic came from or where it is going to, but not both. As long as the user can trust that the owners of the individual proxy servers will not communicate with one another, she need not trust the individual proxy hosts. As with other proxy systems, the end effect is that a given request is routed through a non-filtered location (or in this case many non-filtered locations).

Re-routing systems are susceptible to attacks that correlate the timing of requests (whose origin is known but destination is unknown) and responses (whose destination is known but origin is unknown) to associate the requests and responses and therefore learn both the destination and origin of a given stream of data. For a filtering country to perform such an attack requires that both the requesting client and the responding server be located within the filtering country, so a user is only under risk of this attack when browsing sites that are within the filtering country. Also, the attack is, at least now, prohibitively expensive to perform on any sort of scale, so the attack becomes infeasible as the number of users of a given re-routing tools increases.

Because no single proxy server within a re-routing network has to be trusted, the system can use untrusted volunteers to host proxy servers, and the user can trust the network itself regardless of a lack of trust in any given host. Single proxy systems cannot use volunteers for hosting without requiring their users to trust individual proxies (though some projects such as Psiphon and Circumventor rely on social networks to establish trust for volunteer servers).

The performance of re-routing systems suffers from the need to re-route and re-encrypt traffic through multiple proxies. The increased cost of hosting a network of re-routing servers compounds this problem, since the number of servers and amount of bandwidth required for a given connection is directly related to the number of routing proxies. Each proxy in a series of re-routing proxies costs as much to run, in equipment and bandwidth costs, as each single proxy in a single proxy system such as a VPN. So hosting a network of servers for a re-routing system that routes every connection through a series of three servers costs three times as much to setup and run as a non-re-routing proxying system. Since the resources of proxy projects are generally limited to a set ceiling (either by the funding of the project or by the willingness of volunteers to host servers themselves), the performance of a re-routing system is

roughly divided by the number of servers through which it re-routes each connection (so a system that re-routes a connection through two servers will have roughly half the performance of a single proxy systems with the same resources).

e. Distributed Hosting

A distributed hosting system's security can be trusted as far as the operators of the various servers. Distributed hosting tools require either manual configuration of the browser or client side software that can configure the browser for the user. Once configured, a distributed hosting tool allows the user transparently to use his normal browser interface. Distributed hosting systems might benefit from the wide, non-circumvention related use of distributed hosting technology, though such technology is not widely available to date.

A distributed hosting system mirrors content across a range of participating servers that serve the content out to clients upon request. A distributed hosting system is essentially a caching HTTP proxy that is optimized to store large amounts of data across a number of different hosts. The primary advantage of a distributed hosting system is that it provides access to the requested data even when the original server cannot, for instance if the original server has been overwhelmed by traffic or even taken down by a denial of service attack.

To date, no distributed hosting systems have been deployed solely or even largely for the purpose of circumvention or for direct use by end users. For example, the google cache functions as a distributed hosting system, allowing access to cached versions of most websites from the highly distributed google server farm. However, google provides no direct user interface for accessing its cache. A user must either search for the desired page and click on the 'cached' link underneath the site within the search listings or construct a url in the form 'http://google.com/search?q=<url>' to view a cached page, and the google page is not current enough to use as a proxy for many pages.

II. Proxy Hosting Models

Circumvention tools rely on one of two methods for providing proxy servers: either centralized hosting by the project itself or peer hosting by volunteers.

Centralized hosting systems centralize the trust of the system and provide a simple, mostly linear relationship between cost, performance, and number of users. Because all of the servers are hosted by the project itself, users must trust the project to keep their data private. This configuration can be good or bad depending on the circumstance of the user and the user's opinion of the given project. Performance of a centrally hosted proxy system scales with the performance of the project servers, including both hardware and bandwidth. Increasing performance of the system is a simple matter of increasing the power of the servers and the bandwidth allocated to them. However, the project must also bear the cost for increasing the performance of the servers to support increasing numbers of users. So a centrally hosted proxy system will generally require increased investment in its server infrastructure as its number of users increases. Also, centrally hosted proxy systems are generally housed in server hosting environments with access to high bandwidth connections.

Peer hosting systems displace trust out to volunteer servers and allow for virtually unlimited scaling so long as the number of hosting volunteers scales with the number of clients. Projects that rely on peer hosting rely either on technical techniques such as re-routing to reduce the amount of trust placed on volunteer serving or on social networks to establish trust relationships between server and client users. In order to scale with growth of users, peer hosting projects must also address the freeloader problem of how to encourage users to volunteer as hosts rather than merely using the system as a client. Also, most

volunteer servers offer only low, consumer level bandwidth, which can sharply limit the performance of the proxy systems, even when only lightly loaded.

III. Circumvention Blocking

Many filtering countries use the same filtering methods to block access to and usage of widely used circumvention tools. Most prevalently, filtering countries block both web sites that provide access to (or even information about) circumvention tools as well as the IP addresses of all known proxy servers themselves. The blocking of circumvention project websites is particularly difficult for circumvention projects to combat, since the only way to grow the usage of circumvention tools is to make the locations of the tools widely known.

Countries also try to block proxy addresses in real time. It is very difficult to tell the difference between a legitimate user needing a list of valid proxy addresses and a user grabbing a list of proxy addresses to add them to a filter block list. So circumvention projects must assume that proxy addresses can get blocked very soon after they are published to users and must be able to react to those blocks, both during the initial startup of the circumvention tool (when the tool must discover the location of the proxies) and during the continued usage of the tool (when a current proxy may be blocked at any time).

To remain functional in the face of such blocking, a tool can either change the IP addresses of its proxies more frequently than they can be blocked or can provide each client with only a small subset of the available proxy addresses so that any given client only has enough information to block a small portion of the entire group of proxies.

Filtering countries are also known to have blocked circumvention tools by recognizing certain signatures of the tools (for example, an unusual port used by a tool or a unique keyword within an encryption key). The solution to signature based blocking is to minimize all such signatures from the traffic of a circumvention tool.

IV. The Cost of Anonymity

The simple act of proxying requests, which all of the above circumvention methods perform in one fashion or another, provides an increased level of anonymity by hiding the ultimate destination of the traffic. Encryption of data stream further increases anonymity by hiding destination information that might be present in the content of the packets and by hiding other potentially personal information in the content. Re-routing requests over multiple servers further increases the anonymity of a circumvention service by hiding even from the individual proxy hosts the combination of ultimate origin and ultimate destination of the data.

Each of the above anonymity features, however, incurs additional performance costs and so decreases the usability of the tool in relation to the resources invested in the tool. A highly anonymous tool like a re-routing system can theoretically provide the same performance as a single, unencrypted proxy, but it will cost much more to provide that same performance (or conversely be able to provide that same level of performance to many fewer users).

In addition, to hide the true IP address of the client from the destination web server, an anonymity tool must filter out javascript, java, activex, and other active content, since a malicious server can use them to determine the true IP of the connecting client. Even though such knowledge will not generally allow a filtering country to determine which users are connecting to a given filtered website, the knowledge can potentially reveal the identity of a user known to be using a circumvention tool. The use of cookies exposes users to a similar risk by potentially allowing cooperating web sites to identify a user visiting two different web sites as the same user. A filtering country could use this knowledge to, for example,

collect personal information from the user on one seemingly innocuous web site and then correlate that personal information with a visit to a fake offensive website (one claiming to support political dissidence, for example).

Filtering out javascript, active content, and cookies is relatively cheap in performance costs but hugely expensive in terms of lost usability, since many web sites rely heavily on these technologies. Each tool developer must decide between 1) filtering out these sorts of content to increase user anonymity at the cost of much of the web's functionality and usability and 2) providing access to full functionality of the web at the cost of the potential loss of anonymity.

D. Use Models

The following use models are mostly theoretical, based on interviews with experts on filtering rather than on direct, large scale research on circumvention usage. The intent of the use models is to provide a framework through which to understand how different uses of circumvention technology determine the utility of the various tools and to highlight some contrasting user requirements for the tools.

Establishing real world usage of circumvention tools through direct, large scale surveys of circumvention tool users would greatly inform (though not entirely determine) judgments about which of these uses is most common and therefore worthy of investment. Unfortunately, due to the inherit privacy issues in circumvention tool usage, such research is a difficult project in its own regard. Attempts as part of this report to contact even small numbers of users through circumvention project leaders were mostly unsuccessful for this reason.

I. Visibility

The technical visibility of an Internet user strongly affects the requirements for a circumvention tool. A blogger (or mailing list host, content creator, etc) who is one of only a few bloggers in a given country is much more visible and therefore at risk than a blogger who is own of a large crowd of bloggers in the country. A highly visible lone blogger must remain anonymous not just in the face of large scale, country backbone level filtering, but must remain anonymous in the face of surveillance targeted specifically at him. Filtering and surveillance methods that are prohibitively expensive on a large scale might easily be deployed against such a lone user. It is therefore important that a visible, lone user use circumvention tools that emphasize anonymity over usability. For a blogger who is one among very many in a given country, a lower level of anonymity is likely acceptable, depending on the particular blogger.

A user should also consider her political visibility when considering the use of circumvention technology. Actions that would almost certainly go unnoticed by the large, country level surveillance on a country's backbone routers are likely to be noticed when performed by a user who has already attracted the attention of government. Politically visible users, like technically visible users, will likely benefit from privileging anonymity over usability.

II. Consumption Model

Circumvention tools need not provide unfiltered access for all or even most users. If the goal is to disseminate filtered information widely, the best approach might be to get the information to a small, highly connected and skilled group of community leaders who can pass on the information to the larger, less connected community. In most filtered countries the number of Internet users is still small enough that such brokered modes of communication are necessary.

Brokered communication by a smaller number of users can rely to some degree on the higher level of

technical skill of the information brokers (though certainly not all such users will be expert users) and to a larger degree on the willingness of users to put up with a steep learning curve and usability sacrifices for the circumvention tools. Information broker users are more likely to be both technically and politically visible than other users, so anonymity is likely to be more important than for other users.

In contrast, a Radio Free Europe mode of information consumption requires that the highest possible number of users have access to and, most importantly, choose to use circumvention technology. In this case, ease of use and quality of user experience are the most important factors in order to encourage the widest possible use of the tools. As the number of users increases in this model, the visibility of each individual user decreases and therefore the need for anonymity is lessened somewhat.

III. Access Model

Users gain access to the Internet through a variety of methods that differ by the quality and performance of the network connection and the control over the client computer. Users on poor bandwidth connections will want to use tools that work better in poor bandwidth; our tests show that some tools work consistently better than others in low bandwidth and that other tools are effectively unusable without a high quality connection. Users of public computers in Internet cafes, libraries, and other public computer labs may not have the ability to install software on local computers or may simply not want to expend the effort to install software on a machine that will only be used for a short period of time; software that does not require installation will work better for these circumstances.

3. Aggregate Tool Analysis

The following tools were included in the study:

Anonymizer Anonymous Surfing – an HTTP proxy tool

Anonymizer China – an HTTP proxy tool

DynaWeb FreeGate – an HTTP proxy tool

UltraReach – an HTTP proxy tool

Circumventor / CGIProxy – a CGI proxy tool

Psiphon – a CGI proxy tool

Tor – a randomized re-routing tool

JAP – a fixed re-routing tool

Coral – a distributed hosting tool

Hamachi – a IP tunneling tool

These tools were chosen to represent most of the most popular tools and to represent a range of different technical and organizational models.

There are many tools with the same or similar functionality as the tools included in the study, including Gpass, Guardian, FirePhoenix, Invisible Browsing, Metaproxy, PHPProxy, a plethora of VPN and HTTP tunneling tools, and many others. An obvious extension of this study would involve applying the testing model documented in this study to those tools to gain a more comprehensive view of existing tools. We have also chosen not to include directly any of the lower level tools such as HTTP or CGI proxies or simple lists of proxy servers that do not attempt to automatically connect the user with non-blocked proxies, since these individual servers and lists of servers all suffer from the issues of trust, insecurity, and unreliability described in the relevant proxy method sections of this report.

Peer-to-peer file sharing tools like freenet, mute, kazaa, i2p, bittorent, and many others provide access to a set of data separate from the larger web. These tools offer various interesting features allowing distributed hosting of content, but none of them offers access to existing content on the web. These tools can be used as an alternative publishing medium for content likely to be filtered, but they are all limited by the requirement that someone manually post any such material to the network of each of the closed tools.

A. Tool Comparison Summary

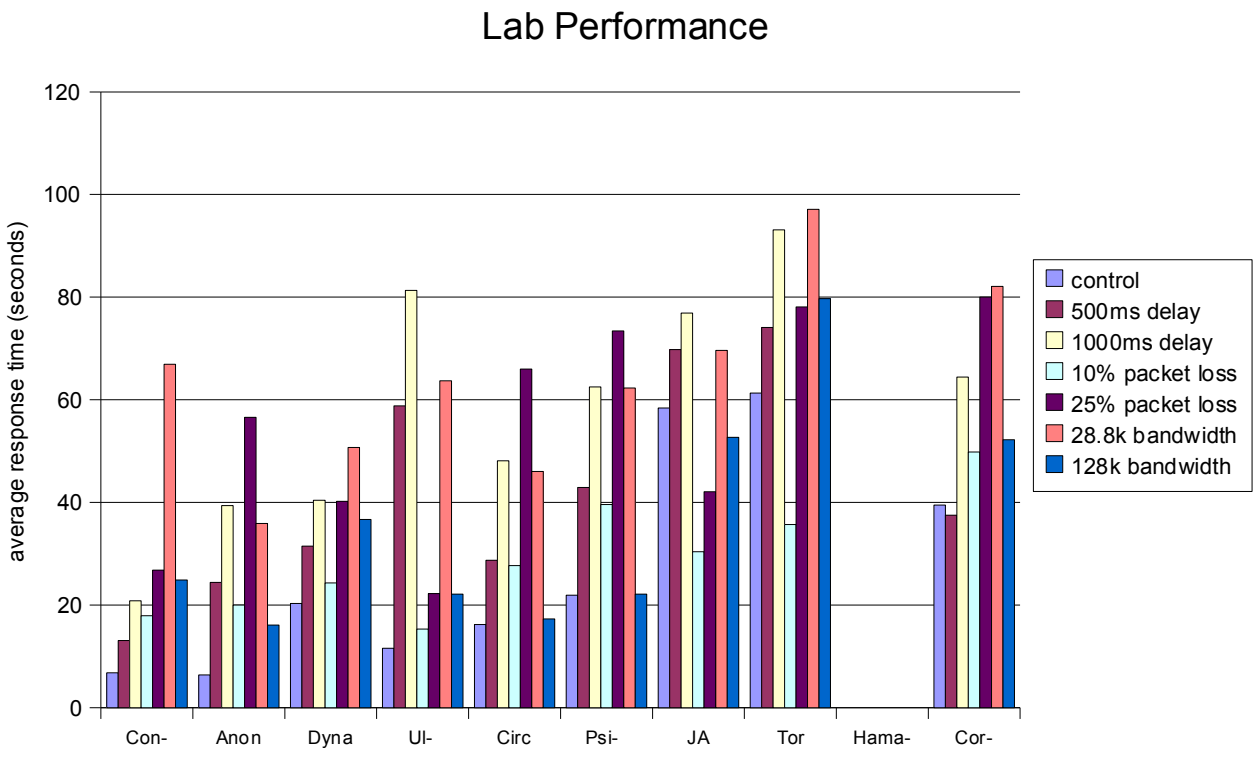
The following table presents a summary of some of the key characteristics of each tool. More details about all of the characteristics listed in the table are in the individual tool reports.

	Anonymizer	DynaWeb	Ultrareach	Circumventor	Psiphon	JAP	Tor	Hamachi	Coral
method	HTTP proxy	HTTP Proxy	HTTP Proxy	CGI Proxy	CGI Proxy	Re-routing	Re-routing	IP tunnel	Distributed Hosting
cost	$30/year	free	free	free	free	free	free	free basic	free
installation	yes	yes	yes	no	no	yes	yes	yes	no
platforms	windows	windows	windows	any	any	all	all	windows	any
server	central	central	central	peer	peer	central	peer	peer	central
antiblocking	yes	yes	yes	yes	yes	yes	no	no	no
languages	3	2	2	1	3	5	15	1	na
encryption	yes, optional	no	yes	yes	yes	yes	yes	yes	no
filters javascript	no	no	no	yes, optional	yes	no	no	no	no
filters active content	no	no	no	yes, optional	yes	no	no	no	no
filters cookies	no	no	no	no, optional	no, optional	no	no, optional	no	no
usage	no data	24 mil hits / day	70 mil hits / day	30 installs / day	8,000 servers	6,000 clients	1 Gbps	no data	1.5 mil clients / day
active dev	yes	yes	yes	yes, minor	yes	yes	yes	no data	yes, minor
free code	no	no	no	yes	yes	yes	yes	no	yes
published design	no	no	no	no	yes	yes	yes	yes	yes
testing process	yes	yes	yes	no	yes	no data	yes	no data	no
internal filtering	yes, optional	yes	yes	no	no	no	no	no	no

B. Aggregate Lab Test Results

For the lab performance test, each of the tools was used to fetch each of Alexa top ten global web sites under a variety of simulated network conditions. Each request was allowed up to two minutes to return a response; any request that took longer than two minutes was given a value of two minutes flat.

The following chart shows the average response time for all of those requests for each tool:



As expected, performance of all of the tools, as well as the control session, degrades significantly under the various simulated network conditions. The HTTP proxy tools (Anonymizer Anonymous Surfing, DynaWeb FreeGate, and UltraReach) perform the best, followed by the CGI proxy tools (Circumventor/

CGIProxy and Psiphon), followed by the re-routing and other tools. Somewhat surprisingly in light of the below in country tests, the HTTP proxy tools perform almost as well as the control session. Part of this result is due to a maximum allowed response time of two minutes for any given response, but certainly not all since the better performing tools had no requests over two minutes for most of the conditions.

Hamachi lacks values for these tests because the Hamachi server was unable to stay up for the duration of the tests.

For the lab filtering tests, the tools were setup on machines behind a test router running filtering software. The filters blocked the same Alexa top ten sites used in the lab performance test. Each tool was used to try to fetch each url through an IP block filter, through a DNS blocking filter, and then through a keyword blocking filter.

The results of these lab filtering tests were mostly as expected. The control session (which used a direct browser request with no circumvention tool) was unable to fetch any of the pages through any of the filters. All of the circumvention tools were able to successfully fetch, in some form or another, all of the blocked pages through the IP and DNS blocking filters. DynaWeb FreeGate and Coral, which do not use full encryption, were unable to fetch pages through the keyword filter, since the keyword filter was able to find some identifying text in the clear text of the unencrypted DynaWeb FreeGate and Coral sessions. All of the other tools, all of which encrypt all of their traffic, were able to fetch all of the pages through the keyword filter.

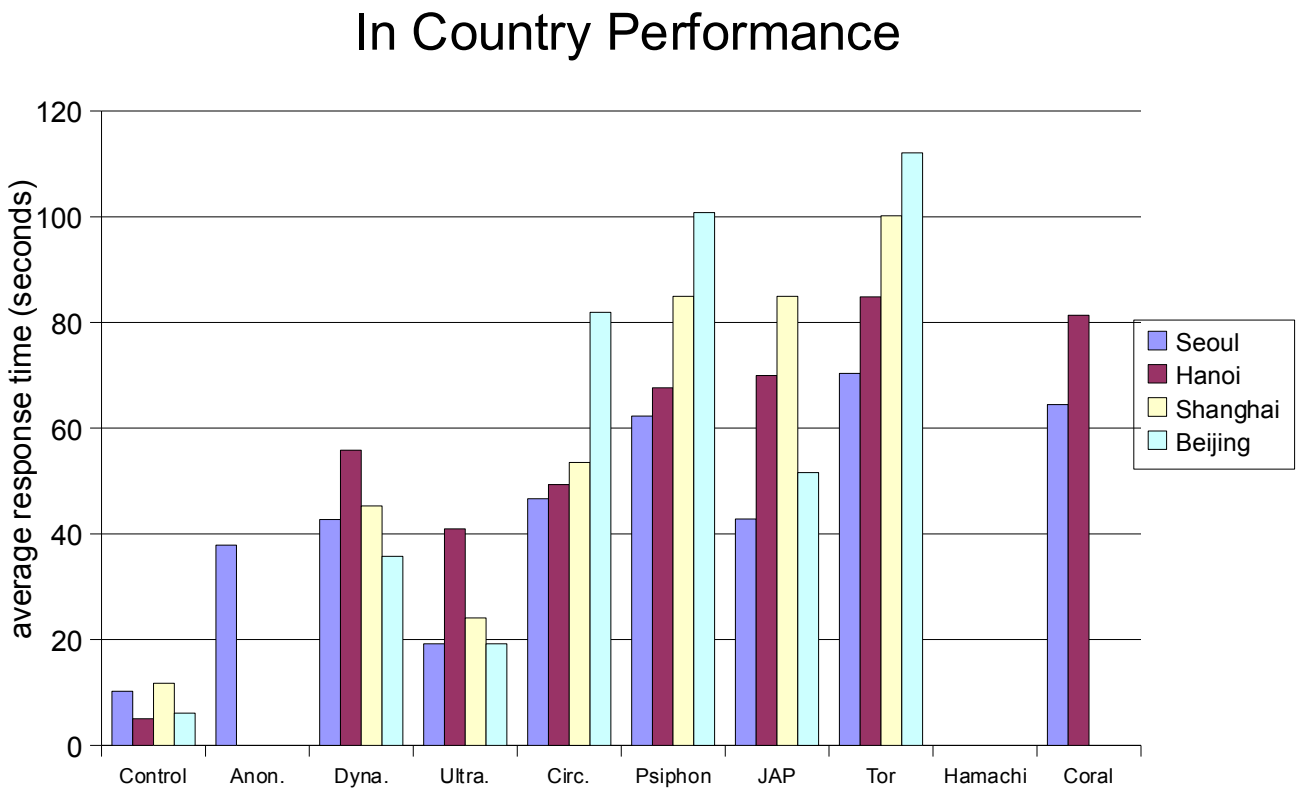
For more details about how the lab tests were setup and run, see the methods appendix.

C. Aggregate In Country Test Results

For the in country tests, a researcher traveled to Seoul, Hanoi, Shanghai, and Beijing to test each of the tools on sites known to be filtered by each country. Each tool was used to fetch about thirty urls. Ten of those urls were the Alexa top ten (unfiltered) sites for the given country on that date. The remaining twenty urls were sites filtered by the country, as determined by the Open Net Initiative. For the control in these tests, the average time only takes into account the first ten sites, since the rest of the sites were necessarily inaccessible through a direct request. The top ten sites for all countries were far slower to load than the blocked sites (the unfiltered sites consisted mainly of news and portal sites with dozens of images, whereas the filtered sites were generally simpler and more text oriented), so the control value is artificially inflated in comparison with the tool values.

Each request was allowed only four minutes to return a value; any request that took longer than four minutes to return was given a value of four minutes flat.

The following chart shows the average response time for all of those requests for each tool:



The performance of the tools in the in country results is strikingly worse than the performance of the direct connection, even with the inflated times of the control session. Browsing through all of the tools other than perhaps UltraReach was painfully slow for sites with many images, and most sites took more than a minute to load for almost all of the tools. As with the lab tests, the HTTP proxying tools performed the best, but the CGI proxying tools in the in country tests performed almost as poorly as the re-routing and other tools.

Blank values for Anonymizer Anonymous Surfing, Hamachi, and Coral indicate that the tools were unable to fetch any sites during the tests. Anonymizer Anonymous Surfing seemed to be blocked in the three of the locations (Hanoi, Shanghai, and Beijing). Hamachi was blocking itself in Vietnam and crashed before completing the tests in the other locations. The Coral network was down to all requests while the tests were performed in Shanghai and Beijing. In addition, both DynaWeb FreeGate and UltraReach consistently failed to return a small number of web pages, most likely because the tools themselves were blocking access to some sites deemed offensive or overly expensive. More information about these failures is included in the relevant tool reports.

For more details about how the in country tests were conducted, see the methods appendix.

D. Notes on Testing Methods

Two points should be noted about the performance testing methods, specifically about the use of the Alexa top ten sites for each of the respective tests. First, the Alexa statistics were chosen for their easy availability rather than their accuracy. The Alexa rankings are determined by web browsing habits of users of the Alexa browser toolbar. Users of the Alexa browser toolbar represent a relatively tiny and

self chosen portion of the web browsing community. It is probable that the sorts of users who choose to use the Alexa browser toolbar will choose to visit different sites than the web browsing public at large. Nonetheless, the Alexa rankings are widely used because they are the best publicly available data of their sort.

Second, to the degree that the Alexa rankings are accurate, using the top ten websites biases the tests toward tools that provide caching on their servers (since the most popular web sites will generally already be stored in the servers' caches). For the in country tests the simpler, more text oriented filtered web sites were returned much faster than the more complicated, image oriented, top ten web sites. However, the relative performance of the various tools (for example, how much faster UltraReach was than JAP) was roughly the same across the filtered and the top ten sites.

4. Anonymizer Anonymous Surfing Report

A. Summary

Anonymizer Anonymous Surfing is an HTTP proxy tool designed primarily to provide anonymity in general and protection from malicious sites. Anonymizer Anonymous Surfing hosts its own pool of proxy servers and uses proprietary techniques to keep them unblocked.

When not blocked, Anonymizer Anonymous Surfing successfully provides access to filtered sites. The tool provides an easy to use interface, though one that is confusing in some potentially dangerous ways. The tool's performance during in country tests was poor for image heavy sites and okay for simple, text oriented sites. Anonymizer Anonymous Surfing includes an optional "malicious site" filter that is turned on by default and uses unclear criteria for site malice. The tool developers claim to use a variety of anti-blocking techniques, but the tool was blocked in three of the four international testing locations.

Of particular concern is the particularly dangerous combination of successful blocking by filtering countries and confusing behavior by the tool in the case of blockage that may lead the user to disclose private information. Until the tool fixes its behavior in the case of blockage, Anonymizer Anonymous Surfing can be recommended for widespread use only for users who are not concerned about losing their anonymity.

This section applies to the retail version of Anonymizer available to the public for a subscription cost at the Anonymizer website. Note that the section following this one addresses the version of Anonymizer developed specifically for circumvention use in China. Anonymizer has also developed a CGI proxy tool for circumvention use in Iran that is not addressed in this report.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	Anonymizer
control	6.8	6.4
500ms delay	13.1	24.4
1000ms delay	20.8	39.4
10% packet loss	17.9	20
25% packet loss	26.8	56.6
28.8k bandwidth	66.9	35.9
128k bandwidth	24.9	16.1

In aggregate, Anonymizer Anonymous Surfing performed 12% slower than the control connection. Anonymizer Anonymous Surfing was the fastest of all the tools for the in lab tests.

Anonymizer Anonymous Surfing successfully returned results for all of the test sites for all of the IP block, DNS block, and keyword block filters. For the following two sites, Anonymizer Anonymous Surfing returned its malicious site warning page:

http://baidu.com

http://www.qq.com

The first is the Chinese search site that cooperates with the Chinese government to filter content. The second is a Chinese language portal site. It is unclear why these sites were deemed malicious, since neither is obviously so (unless censoring counts as malice). See the below section on Internal Filtering

for more information on Anonymizer Anonymous Surfing's malicious site filter. When returning these malicious site warning pages, Anonymizer Anonymous Surfing leaked the name of the blocked sites. These leaks could harm users who request pages that are blocked by both the resident country and Anonymizer Anonymous Surfing, since the leaks would allow a filter at the router to determine that the user was trying to visit a blocked site.

C. In Country Testing Results

Following are average response times for requests to about thirty urls in each of four different locations:

	Control	Anonymizer
Seoul	10.2	37.88
Hanoi	5	no results
Shanghai	11.7	no results
Beijing	6.1	no results

As with all of the circumvention tools, the tool was much slower in country than in the lab. In aggregate, Anonymizer Anonymous Surfing requests were 359% slower than direct requests, making it the second fastest of the tools tested in country.

However, those figures represent only the test run in Seoul, since Anonymizer Anonymous Surfing was blocked in Hanoi, Shanghai, and Beijing. In all of these countries, Anonymizer Anonymous Surfing reported that it was unable to connect to its proxy servers.

Anonymizer Anonymous Surfing displayed its malicious site warning page for the following site:

> http://naver.com

Naver.com is a popular Korean language search and portal site. It is not clear why Anonymizer Anonymous Surfing deemed that site malicious.

D. Availability

Anonymizer Anonymous Surfing is available as boxed software or download and is sold on a subscription basis. A one-year subscription costs thirty dollars.

E. Install

Anonymizer Anonymous Surfing requires the installation of a client side program. The installation is a thirty-three megabyte executable file that requires minimal input from the user and works as expected with the default settings. The program installation includes the standard windows program components, including a program files directory, registry files changes, and .dll libraries installed into the operating system. There is an uninstallation option for the program, but after running the uninstallation, many files and registry entries remain on the computer. Installing the Anonymizer Anonymous Surfing product requires that the user also install unlicensed (and disabled) versions of Anonymizer's other consumer tools (Anti-Spyware, Nyms, and Digital Shredder).

Anonymizer Anonymous Surfing runs on Windows, Linux, and OS X, but the Linux and OS X versions are only available with a command line interface.

F. User Interface

Anonymizer Anonymous Surfing configures itself by default to run automatically on startup. During startup, the tool displays a splash screen for several seconds. Once Anonymizer Anonymous Surfing has initialized itself, it immediately minimizes itself into a task bar icon.

To configure Anonymizer Anonymous Surfing, the user can either click on the task bar icon or double click on the desktop icon for the program. Either method brings up the configuration screen, shown below:

Most users will never need to access the configuration screen but instead will be able merely to use IE to browse as they normally would. All functions of IE work exactly as normal, including bookmarks and other such functions, when running on top of Anonymizer Anonymous Surfing.

For most users, the interface of Anonymizer Anonymous Surfing essentially is IE, since the only interaction with Anonymizer Anonymous Surfing is running IE on top of it. Those users who need to use the user interface for any reason need navigate only the home screen (shown above) and the configuration screen for the Anonymous Surfing product, the link to which is on the left in the above screenshot. The Anonymous Surfing configuration screen allows the user to determine the status of the program (most importantly whether the Anonymous Surfing program is active) and the real and proxied IP addresses of the browsing session as well as to turn Anonymous Surfing on or off. If Anonymous Surfing is turned on, the user can further choose whether to encrypt the connection to the proxy and whether to block malicious web sites.

Among the tested HTTP proxy tools, only Anonymizer Anonymous Surfing offers a quick activation switch that allows users quickly to switch between proxied and direct browsing. Due to the limited in country performance of all of the circumvention tools, such a feature can provide the user with a useful mechanism for quickly switching to lower performance proxied browsing just while browsing blocked or sensitive sites and then switching back to higher performance direct browsing for unblocked, unsensitive sites.

G. Server Component

Anonymizer Anonymous Surfing hosts all of its own proxy servers and does not require the installation

of any server component by the user or his peers.

H. Blocking Resistance

The Anonymizer Anonymous Surfing developers claim to use a number of proprietary technical methods to avoid blocking by filtered countries, but they will not share the techniques in written form or for (even limited) publication.

I. Internationalization

Anonymizer Anonymous Surfing is available in Chinese, English, and Farsi translations.

J. Internal Filtering

Filtering of "malicious web sites" is turned on by default in Anonymizer Anonymous Surfing, but the user can turn off this feature via the configuration screen. When turned on, the filter redirects requests for filtered sites to a block page. The block page that explains that the page has been flagged by the malicious web site filter and that the filter blocks "many sites used for spyware, viruses, spam robots, phishing scams, and other privacy violations," but no specific reason for the blocking of a specific page is given. The user is given the option to "Go ahead to the web site, but filter bad content", "DANGEROUS: Go ahead to the web site without filtering bad content", or "Request that this web site be removed from the blocklist." So the user can continue to view a flagged page against the strong warning of the tool.

Some of the sites flagged as malicious by the Anonymizer Anonymous Surfing filter are puzzling, especially http://baidu.com, which is certainly an objectionable site in the political sense that it censors search results but which provides a simple interface that has no reports of badware features. The problems with widespread over blocking found by the Open Net Initiative in an earlier, CGI proxy version of Anonymizer (see http://opennetinitiative.net/advisories/001/) do not seem to exist in the tested version of Anonymizer Anonymous Surfing, at least with the same websites.

K. Security

There two potentially dangerous problems with the Anonymizer Anonymous Surfing user interface. First, Anonymizer Anonymous Surfing fails to warn its user when it fails to connect to its servers but instead falls back to a direct (unprotected) Internet connection. Second, Anonymizer Anonymous Surfing offers the user the option to turn off encryption without warning that doing so will decrease the security in comparison to not running Anonymizer Anonymous Surfing at all.

If Anonymizer Anonymous Surfing cannot contact its proxy servers, it does not fail, actively warn the user, or prevent IE from fetching pages. It merely fails over to configuring IE use a direct connection to the Internet. This design choice hides the problem from the user and thereby exposes the user to the risk of performing sensitive work (for instance, accessing a banned website) under the mistaken understanding that Anonymizer Anonymous Surfing is keeping that work private. To check whether Anonymizer Anonymous Surfing is protecting the connection, the user has to check the Anonymous Surfing screen in the Anonymizer Anonymous Surfing user interface. Just checking the Anonymizer Anonymous Surfing user interface home page could actually further deceive the user, since this page seems to indicate that the Anonymous Surfing product is working by way of a green icon and an 'Activated' label next to the product name (see above screen). The researcher performing in country tests experienced this behavior first hand and only suspected that Anonymizer Anonymous Surfing was not working because its connection speed was too fast.

Given that Anonymizer Anonymous Surfing failed to find its proxy servers during tests in Hanoi,

Shanghai, and Beijing, this problem is a serious one that could put users in filtered countries at significant risk of performing risky behavior under the illusion of protection from Anonymizer Anonymous Surfing.

Second, Anonymizer Anonymous Surfing offers the option to enable or disable "Surfing Security SSL Encryption" without properly explaining the consequence of this action. Among the "Protection Indicators" below the option, only the "Wi-Fi Security" indicator changes from "Protected" to "Not Protected" when the encryption option is disabled. This display implies that the encryption option effects only wireless security. In fact, encryption is important to protect all traffic, wireless or wired, from the snooping of entities that control the network, including filtering countries. More importantly, unencrypted proxies disable much of the protection of HTTPS connections by decrypting the traffic between the proxy and the browser. So the traffic between the browser and the HTTPS web server, which would normally be encrypted for the whole length of the network, will instead flow unencrypted over the network between the browser and the Anonymizer Anonymous Surfing proxy server.

As mentioned in the lab tests results, Anonymizer Anonymous Surfing leaks the name of the blocked site when blocking a malicious site. If the malicious site is itself private data (for instance of the site is also blocked by the country), this leak could result in the exposure of private data.

Anonymizer Anonymous Surfing does not filter out javascript or active content for most sites, but it attempts to filter out malicious content on sites deemed by its filter to be malicious.

As the number of people who connect to each proxy increases, the security of an individual Anonymizer Anonymous Surfing user will increase, since the origination of traffic from a given proxy server will be more difficult to associate with any one user.

M. Support

Anonymizer Anonymous Surfing provides a user guide and a set of frequently asked questions on the support section of its website. Interactive help is available via phone or email directly from the company. There is no user support forum for users to interact with each other.

N. Development Activity

The first version of Anonymizer Anonymous Surfing was released in 2002. It is currently under active development and will continue to be developed for at least the next year.

O. Community

Anonymizer Anonymous Surfing does not maintain an active user community. The project maintains a mailing list for communicating out to its users, but it has specifically avoided creating any active interaction with its user community.

P. Code Availability

The Anonymizer Anonymous Surfing code is proprietary and not available to the public. The Anonymizer developers did make code for a CGI proxying tool available to the US International Broadcasting Bureau (part of the Voice of America) to develop an Iran specific circumvention tool.

Q. Development Process

Anonymizer Anonymous Surfing's design documentation is proprietary and not available in any form that may be published.

R. Evaluation

Utility – Anonymizer Anonymous Surfing was blocked (and therefore unusable) in three of the four

tested locations.

Usability – In the one site where Anonymizer Anonymous Surfing was able to be tested, it was relatively easy to use and loaded pages more quickly than most comparable tools.

Security – Major security holes exist in the product. Notably, if the product fails to connect to a secure server, it connects to an insecure one without alerting the user.

Promotion – The product is advertised widely in the US, but less so in countries where it would be widely usable.

Sustainability – Anonymizer Anonymous Surfing is a commercial, subscription-based product that produces a profit for its developer.

Openness – While the code is not available for the product, the developers were open and communicative with researchers about how the tool worked.

S. Response

The developers of Anonymizer declined to submit a response.

5. Anonymizer China Report

A. Summary

Anonymizer China is a derivative of the retail Anonymizer Anonymous Surfing product customized specifically for circumvention use in China. Anonymizer China hosts its own pool of proxy servers and uses proprietary techniques to keep them unblocked.

Anonymizer China successfully provides access to filtered sites with an easy to use interface. The tool's performance was adequate in lab tests, but in country performance was not determined since no in country tests were performed for the tool. Anonymizer China closes all of the major security problems found in the retail Anonymizer Anonymous Surfing tool and so provides a relatively secure option for Chinese users. However, the developers restrict use of the Anonymizer China tool only to users located in China, so it is only an option for Chinese users unless and until the developers open up access to more users.

This section applies to the version of Anonymizer specifically customized for circumvention use in China. The section preceding this one addresses the retail version of Anonymizer. The Anonymizer China tool was tested subsequently to all other tools due to a misunderstanding with the Anonymizer developers about which tool was most appropriate for circumvention evaluation. Anonymizer has also developed a CGI proxy tool for circumvention use in Iran that is not addressed in this report.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	Anon. China
control	6.8	16.8
500ms delay	13.1	33.7
1000ms delay	20.8	52.8
10% packet loss	17.9	39.2
25% packet loss	26.8	86.3
28.8k bandwidth	66.9	76.2
128k bandwidth	24.9	25.5

In aggregate, Anonymizer China performed 87% slower than the control connection. Anonymizer China was not included in the aggregate statistics because the tests were performed several weeks later than the other lab tests, but if it had been, it would have been the sixth fastest of ten tools tested.

Anonymizer China successfully returned results for all of the test sites for all of the IP block, DNS block, and keyword block filters. Unlike the Anonymizer Anonymous Surfing tool, the Anonymizer China tool did not return a malicious site warning page for any tested sites.

C. In Country Testing Results

Because the Anonymizer China tests were performed after the researcher travel to Asia, no in country tests were performed.

D. Availability

Anonymizer China is available as a free download at a url that changes roughly weekly to avoid blocking by China. The url to download the tool is published via a mailing list whenever it is changed. The Anonymizer China project relies on existing social networks to distribute these urls to new users.

The page that hosts the Anonymizer China download also hosts a signup page for the mailing list that distributes the new urls.

Even though anyone who knows the current url can download the Anonymizer China client, the tool is restricted to usage from Chinese IP addresses, so only users located in China can use it. Users outside of China (and Iran, for which Anonymizer hosts another tool) are expected to pay for and use the retail Anonymizer Anonymous Surfing tool.

E. Install

Anonymizer Anonymous Surfing requires the installation of a client side program. The installation is a twenty-eight megabyte executable file that requires minimal input from the user and works as expected with the default settings. The program installation includes the standard windows program components, including a program files directory, registry files changes, and .dll libraries installed into the operating system. There is an uninstallation option for the program, but after running the uninstallation, many files and registry entries remain on the computer. Unlike Anonymizer Anonymous Surfing, the Anonymizer China client does not require installation of unrelated software with the circumvention client.

Anonymizer China runs on the Windows platform.

F. User Interface

Anonymizer China configures itself by default to run automatically on startup. During startup, the tool displays a splash screen for several seconds. Once Anonymizer China has initialized itself, it immediately minimizes itself into a task bar icon.

To configure Anonymizer Anonymous Surfing, the user can either click on the task bar icon or double click on the desktop icon for the program.

Most users will only need to access the configuration screen to change the proxy server location. The vast majority of user time will be spent using IE to browse as normal. All functions of IE work exactly as normal, including bookmarks and other such functions, when running on top of Anonymizer China.

For most users, the interface of Anonymizer China essentially is IE, since the only interaction with Anonymizer China is running IE on top of it. Those users who need to use the user interface for any reason need navigate only the home screen and the configuration screen for the Anonymous Surfing product. Notably unlike the retail Anonymizer Anonymous Surfing product, the Anonymizer China product does not allow the user either to disable anonymous surfing or to disable encryption within an anonymous session. The removal of the option to disable these features decreases the chance that a user might accidentally open herself up to surveillance or filtering.

As discussed below on Blocking Resistance, Anonymizer China users will need to use the Proxy Preferences screen to change the proxy host used by the client roughly weekly to prevent the tool from getting blocked. New proxy settings are sent to users through a mailing list that users can sign up for on the same page where they download the Anonymizer China client.

G. Server Component

Anonymizer China hosts all of its own proxy servers and does not require the installation of any server component by the user or his peers.

H. Blocking Resistance

The primary strategy for blocking resistance is to setup a new proxy server roughly weekly and send the location of the new proxy server to users through a mailing list. Users can sign up for the mailing list on the same page where they download the client. As China becomes more efficient at blocking these proxy sites, Anonymizer China will setup and publish new proxy sites more frequently, though this strategy will break down if China is able to block the proxy sites within a day or so. Also, this strategy requires significantly more user interaction (keeping up with a mailing list, and regularly reconfiguring the client) than the automated resistance strategies offered by several other tools.

I. Internationalization

Anonymizer Anonymous Surfing is available in Chinese, English, and Farsi translations.

J. Internal Filtering

We were unable to find any internal filtering performed by the Anonymizer China client. This lack of internal filtering is a notable difference between the Anonymizer China client and the retail Anonymizer Anonymous Surfing product, which includes a malicious site warning system.

K. Security

None of the three significant security issues that exist within the retail Anonymizer Anonymous Surfing product exist within the Anonymizer China product. Most importantly, when the Anonymizer China product fails to make a connection to its proxy server, IE fails to load any web pages. Even though a warning to the user of why the pages are not being loaded would be preferable, not loading the pages is the much better alternative to loading them directly without the help of a proxy. Likewise, the Anonymizer China client disables the option to turn off encryption, closing that potential security hole, and does not support malicious site filtering, closing the resulting data leak shown by the retail product.

As the number of people who connect to each proxy increases, the security of an individual Anonymizer China user will increase, since the origination of traffic from a given proxy server will be more difficult to associate with any one user.

M. Support

Anonymizer China provides a user guide and a set of frequently asked questions on the support section of its website. Interactive help is available via phone or email directly from the company. There is no user support forum for users to interact with each other.

N. Development Activity

The first version of Anonymizer China was released in 2002. It is currently under active development and will continue to be developed for at least the next year.

O. Community

Anonymizer China does not maintain an active user community. The project maintains a mailing list for communicating out to its users, but it does not have a forum for active interaction with its user community.

P. Code Availability

The Anonymizer China code is proprietary and not available to the public. The Anonymizer developers did make code for a CGI proxying tool available to the US International Broadcasting Bureau (part of the Voice of America) to develop an Iran specific circumvention tool.

Q. Development Process

Anonymizer China's design documentation is proprietary and not available in any form that may be published.

R. Evaluation

Utility – Anonymizer China was not tested in country, but performed relatively well in lab tests.

Usability – Anonymizer China is as usable as the Anonymizer Anonymous Surfing product.

Security – The major security holes of Anonymizer Anonymous Surfing are closed in Anonymizer China. The system relies on Anonymizer to protect access to its servers and not to disclose user behavior data to a third party.

Promotion – Anonymizer China is not promoted as widely as DynaWeb, UltraReach, or Psiphon in China, its only market.

Sustainability – Anonymizer China is a subscription product, subsidized by profits from the sales of the retail Anonymizer Anonymous Surfing product.

Openness – While the code is not available for the product, the developers were open and communicative with researchers about how the tool worked.

S. Response

The developers of Anonymizer declined to submit a response.

6. DynaWeb FreeGate Report

A. Summary

DynaWeb FreeGate is an HTTP proxying tool designed specifically for circumvention use in China. The DynaWeb FreeGate project hosts its own collection of proxy servers and uses a variety of proprietary methods to keep its proxy servers unblocked in the face of aggressive, ongoing attempts to block them on the part of the Chinese government.

DynaWeb FreeGate provides an easy to use interface that successfully bypasses most types of filtering in the face of active blocking attempts. DynaWeb FreeGate's performance is poor for image heavy sites but okay for simpler text oriented sites. The DynaWeb FreeGate network implements its own set of filters that block a wide range of sites that are offensive to the DynaWeb FreeGate developers. Most importantly, the DynaWeb FreeGate tool uses an error prone encryption mechanism that fails to encrypt some of its traffic, allowing snooping by the network and making the tool vulnerable to keyword block filters.

Because it uses an error prone encryption mechanism that fails to encrypt all of its traffic, DynaWeb FreeGate can be recommended for widespread user only by users who are not concerned about anonymity.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	Dynaweb
control	6.8	20.3
500ms delay	13.1	31.5
1000ms delay	20.8	40.4
10% packet loss	17.9	24.3
25% packet loss	26.8	40.2
28.8k bandwidth	66.9	50.7
128k bandwidth	24.9	36.7

In aggregate, DynaWeb FreeGate performed 37% slower than the control connection, making it the second fastest of the nine tools tested in the lab. The average for all tools was 85% slower than the control connection.

DynaWeb FreeGate successfully circumvented the lab IP and DNS block filters but was blocked by the keyword block filter because the tool failed to encrypt the host name of the requested web server. This lack of encryption also opens up users of DynaWeb FreeGate not only to keyword blocking but also to surveillance of their web browsing activities.

C. In Country Testing Results

Following are average response times for requests to about thirty urls in each of four different locations:

	Control	Dynaweb
Seoul	10.2	42.72
Hanoi	5	55.83
Shanghai	11.7	45.3
Beijing	6.1	35.77

As with all of the circumvention tools, the tool was much slower in country than in the lab. In

aggregate, DynaWeb FreeGate requests were 444% slower than direct requests, making it the third fastest of the nine tools tested in country. The average response time for all tools during the in country tests was 616% slower than the control session.

In addition, DynaWeb FreeGate consistently returned a 502 error when attempting to load the following sites in country:

http://www.dprk-stamp.com

http://www.dprk-book.com

http://sina.com.cn

http://taobao.com

http://msn.com

http://myspace.com

When requests for the above pages were repeated from the U.S., requests for the first four sites continued to return a 502 error, but requests for msn.com and myspace.com were returned successfully. These 502 errors are most likely the result of deliberate IP block filtering by DynaWeb FreeGate itself. DynaWeb FreeGate admits to filtering out requests for what it considers to be bad content and considers such filtering to be a feature of the tool.

D. Availability

The DynaWeb FreeGate application is available as a free download at the Chinese language DynaWeb FreeGate site at https://us.dongtaiwang.com/. Because the site is only in Chinese, it is very difficult to find via an English language search for 'DynaWeb' or 'FreeGate'. The site that is returned first in google searches for 'DynaWeb' and 'FreeGate' returns a link to the company that makes DynaWeb FreeGate, but that site includes no link to download the tool itself. However, searching for dongtaiwang, which is the Chinese name of the tool, returns the Chinese language dontaiwang.com site where the tool can be downloaded with some careful examination of link locations. There is no charge for downloading or using DynaWeb FreeGate.

E. Install

The DynaWeb FreeGate.exe application must be installed and run on the client computer to use the DynaWeb FreeGate system. This .exe file, however, requires no installation other than downloading the single .exe file to any location and running it. The tool was tested for this report by simply downloading the provided .exe file to the desktop and repeatedly running the file from its desktop location.

Installing DynaWeb FreeGate leaves only the .exe file itself and a few registry entries that store DynaWeb FreeGate configuration information. Even though unintsalling the DynaWeb FreeGate application requires only deleting the FreeGate.exe file, the tool provides no uninstall option to remove its registry entries, making it difficult to erase evidence of having run DynaWeb FreeGate from a computer.

DynaWeb FreeGate runs only on the windows platform.

F. User Interface

A user can access the DynaWeb FreeGate proxies by starting the FreeGate user interface, which locates a non-blocked DynaWeb FreeGate proxy, configures IE to use that proxy, and then starts IE.

Once IE starts up, it is operating in proxy mode through the DynaWeb FreeGate proxies, so the user browses as she normally would. Using DynaWeb FreeGate in this default mode is straightforward since it does its work without user intervention and then provides the user with the browser interface to start browsing.

Non-default uses of DynaWeb FreeGate are not nearly as clear, either in the concise documentation or in the interface itself. For example, the FreeGate client allows the user to browse a cached version of some news sites without a connection to the Internet. But the great majority of users will find no need to use anything other than the default behavior of the application.

In total, DynaWeb FreeGate generally takes less than ten seconds to deliver the user to the proxied IE interface, including a second or so to start the FreeGate application, a few seconds to find an available proxy server, and a second or so to start IE.

G. Server Component

DynaWeb FreeGate hosts all of its servers. There is no need for any user to install any server software.

H. Blocking Resistance

The DynaWeb project leaders claim that the project uses a number of proprietary techniques to avoid aggressive, ongoing blocking efforts by the Chinese government, but they consider the techniques confidential and will not share any information about them. They claim to be constantly planning and implementing changes to improve the tools' blocking resistance.

In addition to the proprietary blocking resistance built into the tool, DynaWeb distributes lists of its proxy servers via a number of out of band channels, including, email, alternate web pages, and even an instant messaging bot.

I. Internationalization

DynaWeb FreeGate is available in Chinese and English translations. The application requires no input from the user, so many users should be able to run the application even without a proper translation.

K. Security

The DynaWeb project is concerned primarily in providing the highest quality user experience at the cost of anonymity concerns. Most importantly, DynaWeb FreeGate uses an error prone encryption method that does not encrypt all of its request data, exposing that data to snooping by intermediary routers, including country level backbone routers. All of the other tools discussed in this report other than Coral (which uses no encryption at all) encrypt the entire stream of HTTP data, either through the HTTPS protocol or through an underlying encrypted IP tunnel. DynaWeb FreeGate instead encrypts or otherwise hides the contents of each individual HTTP header as well as of the content. When all headers are encrypted, this approach works well, but in practice it is difficult to capture the cases of all HTTP headers. The data leak found in the lab testing for this report was due to the failure to encrypt a P3P header and a custom yahoo header. The danger of this approach is that data leaks will continue to occur as new standards for headers are developed and as sites implement their own, custom uses for HTTP headers.

DynaWeb FreeGate also does not filter any javascript, active content, or cookies from requests or responses, providing users with access to such content at the cost of exposing them to identification by servers.

The security of DynaWeb FreeGate users, though limited by the encryption method, will increase as the number of users increases, since more users for the tool will lead to more users on a given proxy. The more users on a given proxy, the harder it is to track the traffic from that proxy to a specific user.

If the DynaWeb FreeGate network fails for any reason, the browser either will not start (if the proxies are not found initially) or start returning errors (if the tool loses its connection to the proxies while already in operation).

M. Support

The user guide for the FreeGate application, available via the help button within the application, is only a couple of paragraphs long and provides only a general overview of the tool along with some specific usage tips. There is a very active Chinese language forum for community support of DynaWeb FreeGate at http://qxbbs.org.

N. Development Activity

The DynaWeb FreeGate project was first released in March 2002 and has been actively developed since.

O. Community

The DynaWeb project has a very active user community that gathers at http://qxbss.org to support and otherwise interact with each other and with the DynaWeb developers. The DynaWeb project leaders actively interact with the user community.

P. Code Availability

The DynaWeb FreeGate code is no freely available.

Q. Development Process

The design of the DynaWeb FreeGate system is not documented outside of the project.

R. Evaluation

Utility – DynaWeb FreeGate enables users to evade IP and DNS filtering, but does not encrypt all hostnames, which leaves it vulnerable to keyword blocking, reducing its utility in its target nation, China.

Usability – The tool is easy to use, well documented and well supported in Chinese.

Security – DynaWeb FreeGate used an overly complicated encryption method that failed to encrypt some data.

Promotion – The product is well known and widely used.

Sustainability – DynaWeb FreeGate relies on a large set of servers to provide access to the Internet. These servers are expensive to operate, and it's not clear if DynaWeb can support the expansion of this server farm.

Openness – The code for DynaWeb FreeGate is unavailable and the developers were unwilling to share technical information on how the system operates.

S. Response

The developers of DynaWeb FreeGate submitted the following response to the report:

> Thanks for the pioneering work in "2007 Circumvention Landscape Report" ("the Report").
>
> DynaWeb is operated by Dynamic Internet Technology Inc. (DIT). DynaWeb provides proxy url links that can be easily redistributed. DynaWeb also has the FreeGate software that connects users to the DynaWeb network.
>
> Along with Ultrasurf, DynaWeb has been among the most popular anti-censorship tools in China since 2002. By the end of December 2008, DynaWeb was serving more than 1 million users everyday. The peak traffic was 1300Mbps. About 25 percent of the hits were from China, and another 25 percent of hits were from Iran. When IP blocking is more active, DynaWeb utilized more than 15,000 IPs in a single day.
>
> During the last seven year, the DynaWeb team has collected a large amount of new information, and has faced many issues and decisions not fully discussed in the computer security and privacy community. Within the length limit, we would like to share some of them that may suggest further improvement of such report. When we replied to the email survey from the author two years ago, we did not recognize that the information may be helpful for the authors. The numbers used below are current numbers.
>
> DynaWeb's objective is to serve non-pornographic traffic to countries that has Internet filtering from totalitarian governments. With limited manpower and funds, our highest priorities are user security, expanding the user base (not necessarily through tools we own or operate) while under attacks, and the ability to scale up quickly when more funds are available. We view this work as urgent and lifesaving, as demonstrated by the traffic rise during the SARS epidemic and the Melamine Milk Crisis.
>
> It is not clear for us what kind of objectives a tool would have when marked as best by the criteria and methodology set by the Report.
>
> 1. Security: DynaWeb has never received any report about security breach.
>
> By the end of 2008, DynaWeb secured more than 300 million user sessions. The DynaWeb team works closely with the Falun Gong community and Chinese dissidents community. The Falun Dafa Information Center (www.faluninfo.net) documented more than three thousand cases of death due to persecution, and many more cases of arrests and harassment. None of

these cases were caused by security issues related to DynaWeb, although DynaWeb is widely used by Falun Gong practitioners and dissidents. Many dissidents openly publish articles online, mentioning that they are using DynaWeb.

2. Resilience to attacks: China only blocks popular tools like FreeGate, UltraSurf, Gpass, etc. Methods used to block DynaWeb have been upgraded many times. For example, Bill Xia presented DynaWeb's 2003 finding about TCP Reset at the 2004 HOPE conference (http://dit-inc.us/report/hope2004/main.html). In last two years alone, there were five upgrades.

This part has cost the team the most efforts and is the most challenging part for any tool to be popular in China. However, these efforts and achievement are hard to recognize without first hand experience with those blocking methods.

3. Scalability: A proxy network starts with the establishment of infrastructure. DynaWeb's current operational scale, in terms of bandwidth and IP, is at least ten times more than other volunteer-based proxy networks.

Engineers from the "Global Information Freedom Consortium" evaluated the scalability of Freenet in 2001. We are unable to identify a way to attract far more volunteers to contribute more bandwidth and IPs beyond the 1000 Freenet nodes we estimated at that time. So, we dropped our effort to attract more Freenet node volunteers and to customize Freenet for Chinese users. The Freenet-China engineers later released various tools like Garden Network, DynaWeb (with FreeGate as one of the software to access it), UltraSurf, GPass, FirePhoenix, etc. Those tools have been multiplying total users base for anti-censorship every few months.

On the other hand, according to publicly available information we can find, Tor's number of nodes have stayed between 1000 to 2000 over the last two years.

4. Promotion: Dynamic Internet Technology has made many promotional efforts. For example, it has been providing mass mailing service to Voice of America and Radio Free Asia. It sends out millions of emails to China everyday for that contract. The email carries DynaWeb proxy links where people can download FreeGate software. It is an important strategy for DynaWeb to provide proxy links so that it can be easily redistributed.

We are unaware of any other tools which have such scale of promotion capability to China.

DynaWeb further links to other tools we found to have the potential to be popular. We encourage users to try and keep multiple tools. This practice may lower DynaWeb's popularity, but it complies to our objectives of enlarging the total anti-censorship user base.

The Report seems to link Google search results to promotion and "online visibility." In China, searching "FreeGate" on Google will trigger TCP reset and disconnect the user from Google. "FreeGate" is on many censorship blacklists in China due to its popularity. Regular large scale web distribution for any popular tools won't reach users in China. There are many details on how users got to know DynaWeb. Much of the promotion is "facilitated" by DynaWeb, instead of "pushed" as mentioned in the Report. Any "push" itself will have filtering problems.

5. Funding: Our bottleneck is funding for operations and development to constantly stay ahead of censors. To preserve funds for the upcoming growth of users from China, DynaWeb has limited users from outside of China starting January 1, 2009. After limiting non-China

traffic, we started to receive more interests to cover cost for certain countries.

In the last seven years, DIT has been contracting with Voice of America's Chinese branch and Radio Free Asia for DynaWeb service for their contents and customized version for them. Voice of America's Persian branch and Radio Farda recently transferred their anti censorship contracts from another provider to DIT.

Most content providers who are blocked in China found most of the Chinese visitors visit them through DynaWeb or Ultrasurf. Most of these contents are served directly from DynaWeb's cache. Now, we are able to lower the cost to be comparable to content network provider like Akaima. Besides seeking funds from private foundation and government, DynaWeb may charge content providers to serve their contents to more users.

As to the report itself, our overloaded engineer is only able to read a few pages around the comparison table, Psion and FreeGate sections. We have so many questions about it. With limited time, English language and communication skill, we will post those questions on http://www.dit-inc.us/report/DynaWeb-response.html.

7. UltraReach Report

A. Summary

UltraReach is a circumvention tool that uses HTTP proxying. UltraReach is designed specifically for circumvention use in China.

UltraReach provides an easy to use interface that is able to access all content in the face of active blocking attempts. UltraReach's performance is the best of any tool tested in filtering countries, the only tool to display okay speed for both image heavy and simple, text oriented sites. UltraReach implements its own filters that block offensive material according to its own criteria.

UltraReach can be recommended for widespread use as the best performing of all the tested tools, though users concerned about anonymity should be warned to disable browser support for active content.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	Ultrareach
control	6.8	11.6
500ms delay	13.1	58.8
1000ms delay	20.8	81.3
10% packet loss	17.9	15.3
25% packet loss	26.8	22.2
28.8k bandwidth	66.9	63.7
128k bandwidth	24.9	22.1

In aggregate, UltraReach performed 55% slower than the control connection, making the fourth fastest of the nine tools tested in the lab. The average of all tools was 85% slower than the control connection.

UltraReach successfully bypassed all three of the IP block, DNS block, and keyword block filters for all ten of the tested sites. The request for one site failed with a bad gateway (502) error during the IP block filter but returned a successful response immediately thereafter.

C. In Country Testing Results

Following are average response times for requests to about thirty urls in each of four different locations:

	Control	Ultrareach
Seoul	10.2	19.19
Hanoi	5	40.93
Shanghai	11.7	24.1
Beijing	6.1	19.17

As with all of the circumvention tools, the tool was much slower in country than in the lab. In aggregate, UltraReach responses were 213% slower than direct requests. The average aggregate response time for all tools was 616% slower than the direct requests. UltraReach was the fastest of all the tools during the in country testing.

UltraReach was not blocked in any countries. UltraReach returned an UltraReach search page instead of the requested site for the following sites:

http://www.big.or.jp

http://daum.net

http://nate.com

http://coithienthai.com

As discussed in the Internal Filtering section below, UltraReach uses an internal filter on its proxy servers that filters out some sites. The above sites were blocked by that filter.

In addition, UltraReach consistently returned a connection refused error for the following site:

http://thongluan.org

UltraReach returned a successful response for the above site from the U.S. a couple of weeks after the in country test in Vietnam where the error occurred. The cause of this error is unclear.

D. Availability

UltraReach is available for free as a download from the UltraReach web site at http://UltraReach.com. There is no charge for installing or using UltraReach. The web site is easily located via a google search or guessing the domain name.

E. Install

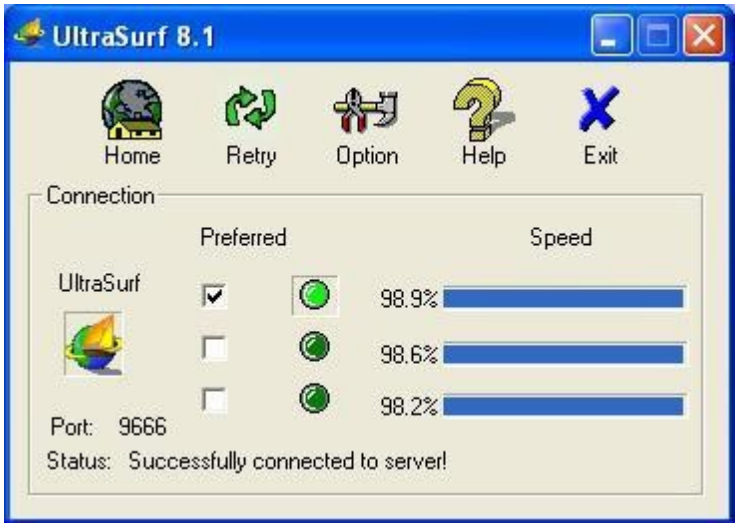
The UltraReach tool requires the installation of a local u.exe file. Installation consists of downloading a file called "u.zip" and unzipping that file into any directory. The resulting u.exe file can be run from anywhere. The tests for this report were run with u.exe on the desktop of the testing computer. The only possible confusion with the simple installation process is that some novice users do not understand what to do with a .zip file and none of the instructions mention what to do with the u.zip file. The large majority of users will know to unzip that file, though, and run the u.exe they find within.

UltraReach takes care to install no files other than the u.exe file on the computer during either installation or ongoing usage and leaves no registry changes after it exits, so uninstallation requires only deleting the u.exe file.

UltraReach runs only on the Windows platform.

F. User Interface

On startup, the u.exe program displays the following small screen showing the status of three UltraReach proxies. Once the program has connected to its proxies, it automatically configures IE to use those proxies and then starts IE. Once started, IE behaves exactly as it normally would, other than for the fact that it is now connecting to web sites through a remote IE proxy.



The interface provides buttons that allow the user to visit the UltraReach home page in IE, to retry connecting to the proxies, to set some options, to read the UltraReach help document, or to exit UltraReach. The Home and Retry buttons are somewhat unintuitively labeled, and the help button leads to the Chinese language help file in the English version. The vast majority of users, though, will have no need to do anything with the user interface other than start it, browse in IE, and then close it when done.

The application takes less than a second to start and anywhere from one to several seconds to connect to the proxy servers. Total startup time is less than ten seconds.

G. Server Component

UltraReach hosts all of its own servers. There is no server component for a peer to install.

H. Blocking Resistance

UltraReach uses a set of proprietary methods for maintaining blocking resistance in the face of blocking attempts by China and other countries.

I. Internationalization

UltraReach is available in Chinese and English translations. There is documentation in Chinese and English, though the help file available from the user interface is only in Chinese; English language users must download the English language user guide from the web site.

K. Security

UltraReach is designed primarily as a circumvention tool rather than an anonymity tool. It trades off strong anonymity for the user by allowing javascript, active content, and cookies all to pass through unfiltered.

If the UltraReach network fails for any reason, the browser either will not start (if the proxies are not found initially) or start returning errors (if the tool loses its connection to the proxies while already in operation).

As the number of people who connect to each proxy increase, the security of an individual UltraReach user will increase, since the origination of traffic from a given proxy server will be more difficult to associate with any one user.

J. Internal Filtering

UltraReach includes an internal filter that blocks its own list of sites.

M. Support

UltraReach makes a Chinese language user guide available via its web site and via a help button on the user interface, and an English language user guide available via its web site. The user guide is reasonably clearly written and includes an overview of the tool, a description of all of its user interface options, a set of frequently asked questions, and a description of some of the technical features of the underlying architecture.

A large, active Chinese language user community provides support for the tool.

N. Development Activity

The UltraReach project release the first version of UltraReach in 2001. The tool has been under active development since that date.

O. Community

The project has an active Chinese language user community at http://qxbss.org, where the project leaders interact with the users.

P. Code Availability

Code for the UltraReach tool is not freely available but may be made available for audit by individuals at the discretion of the project leaders.

Q. Development Process

There is no publicly available description of the design of UltraReach about the architecture of the tool was gleaned through private interviews with the project leaders.

R. Evaluation

Utility – UltraReach successfully circumvented IP, DNS and keyword blocks. It was the fastest of all tested tools in in-country tests, but was still significantly slower than accessing the web directly.

Usability – UltraReach is easy to use, well documented and well supported in Chinese.

Security – UltraReach is not designed as an anonymity tool and permits javascript and other features that may leak user information. Like the Anonymizer and DynaWeb products, it requires the user trust a central server not to reveal user data.

Promotion – The product is well known and widely used in China.

Sustainability – Like DynaWeb, the product requires a server farm, which requires annual bandwidth and hardware investment. It is unclear whether UltraReach can sustain this farm in the long term.

Openness – While the code is not available for the product, the developers were open and communicative with researchers about how the tool worked.

S. Response

The developers of UltraReach submitted the following response:

> Congratulations on your pioneering work in surveying the circumvention landscape. I am impressed by the comprehensive coverage of this report. While I appreciate the value of the

work you've brought in this work, I'd like to take this opportunity to provide some feedbacks regarding the interpretation of the test results.

1) I am pleased to learn that in the In County Test Results UltraReach is ranked the fastest among all tools under survey. At Ultra Reach, we have primarily focused on high performance with necessary privacy and security features in our research and development. We believe that performance is about user experience, it is also reflected in the capability of handling significantly increased data traffic with affordable resource. Scalability is emphasized from design to implementation, as well as in operations.

2) A couple of entries in the In Lab Testing Results have weighed on UltraReach's overall performance, placing it in the fourth place among 10 tools surveyed. This appearance of inconsistency with relevant results in the In Country Test Results could be caused by the coincidental migration of Ultra Reach from version 8.1 to version 8.2 when the In Lab Testing was conducted. During this period of time, hardware resources had been shifted to the in-production operation of version 8.2, while version 8.1 resources were being gradually phased out.

3) When UltraReach tool is started, the IE features that possibly compromise user security are turned off by modifying user's Windows Registry. The Windows Registry is restored on the exit of the tool. More specifically,

(1) "Java VM" feature is disabled.

(2) Active scripting setting is not modified because it doesn't have security concern when UltraReach proxy is used. Active scripts such as javascript from a web server that attempts to reveal a user IP address will get the UltraReach proxy IP address instead of the user IP address.

I appreciate your efforts in producing first-hand, authoritative coverage of competing tools. If there is any further information that I am able to provide, please don't hesitate to contact me.

8. Circumventor/CGIProxy Report

A. Summary

CGIProxy is a cgi script that uses cgi proxying to support circumvention. Circumventor is a set of installation scripts that bundle together CGIProxy with a windows version of the Apache web server to provide a secure, easy to install version of CGIProxy. Circumventor was developed primarily for use by students to avoid school filtering in the U.S. but has expanded to focus on international filtering as well. Circumventor uses a peer-to-peer hosting model that relies on volunteers to host servers for their peers and also makes those peer hosted servers available to the larger community.

Circumventor provides an easy to use client interface that requires no client software installation and successfully provides access to content through all kinds of filters. The performance of Circumventor is poor for both image heavy and simple, text oriented sites. Circumventor filters out active content, but it fails to encrypt all HTTP requests, leading to unintended data leaks. Circumventor relies on volunteers to host servers for their peers, but the server installation is difficult, has undocumented effects on the host computer, and provides no clear uninstallation path.

Circumventor can be recommend for widespread client use but only for server use by expert users who are able to navigate the difficult install process and understand the effects of the install on the host computer. Circumventor users should also be aware that some data leaks are possible.

Circumventor and CGIProxy are separate projects but have been bundled together for the sake of this report because only bundled do they offer the same range of features (including widely accessible user installation and proxy list management) that other tested tools offer. Since Circumventor bundles the CGIProxy tool within itself, we use the term "Circumventor" to refer to the Circumventor/CGIProxy bundle for the purpose of this report.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	Circumventor
control	6.8	16.22
500ms delay	13.1	28.7
1000ms delay	20.8	48.1
10% packet loss	17.9	27.7
25% packet loss	26.8	66
28.8k bandwidth	66.9	46
128k bandwidth	24.9	17.3

In aggregate, Circumventor performed 41% slower than the control connection. Out of all tools tested in the lab, Circumventor was the third fastest. The average across all tools during the lab tests was 85% slower than the control.

Circumventor was able to retrieve all ten sites through the IP block, DNS block, and keyword block filters. However, the keyword filter did detect one data leak during the response for yahoo.com. During that request, the browser submitted the following clear text request:

```
GET /us.yimg.com/i/ww/them/1/search_1.1.png HTTP/1.1
Host: us.il.img.com
```

This request represents a data leak because it 1) was sent directly to the web server in question instead of

to the Circumventor server and 2) was sent in clear text as an HTTP request rather than encrypted as an HTTPS request. Sending the request directly to the server allows a router on the network to discover the identity of the web server the user is browsing. Sending the request in clear text allows a router on the network to determine both the identity of the web server and the particular file being requested. Either of these bits of data could allow a snooping router to find out that a particular user was requesting a sensitive site even though the user believed herself protected.

The cause of this leak was the failure of CGIProxy to transform an image embedded in a css stylesheet. To ensure that all traffic or a given page goes through the Circumventor cgi proxy, the script must make sure not only the html for a given page but also all of the supporting css, image, and other supporting files load from the Circumventor cgi proxy server rather than directly from the designated servers. One of the weaknesses of cgi proxy systems is that all such embedded content is not obvious (as in this case with the guilty css image) and that the tool must transform every single link to avoid data leaks.

Circumventor also skewed the display of many of the requested pages, causing the whole page to shift as much as half a screen to the right. This display problem is most likely the result of a bad interaction between the html for the requested page and the html for the address bar that tool inserts into the requested page.

C. In Country Testing Results

Following are average response times for requests to about thirty urls in each of four different locations:

	Control	Circumventor
Seoul	10.2	46.63
Hanoi	5	49.34
Shanghai	11.7	53.53
Beijing	6.1	81.93

As with all of the circumvention tools, the tool was much slower in country than in the lab. In aggregate, Circumventor requests were 601% slower than direct requests, making it the fourth fastest of nine tools during the in country tests. The average across all tools during the in country tests was 616% slower than the control.

Circumventor was not blocked in any of the locations tested and was able to load all requested pages except for two. The tool consistently loaded random garbage characters instead of the content for http://google.com.vn, and it consistently loaded a garbage title and a blank page for http://freechina.net. The cause of these problems is not clear but is not related to the country filters, since the errors reported themselves when the pages were tested from an unfiltered connection in the U.S.

As with the in country tests, the display of many pages was skewed to the right.

For both the in country and lab tests, the Circumventor server ran without problems for weeks without need for any active administration.

D. Availability

Circumventor/CGIProxy is freely available for non-commercial uses via download from the Circumventor web site. The project web site is easily findable via google and other search engines.

E. Install

Circumventor requires no client installation. A client need only point her browser to an existing Circumventor/CGIProxy server to start browsing via the proxy. As a browser based tool, Circumventor

leaves no side effects other than browser history files and is supported by all platforms.

F. User Interface

The Circumventor client user interface consists of an address bar into which to enter a url and button to fetch that url. The address bar is included by default at the top of each page returned, so the user can continue browsing through Circumventor simply by typing a new url into the Circumventor address bar. The following shows a screen of a user browsing google.com via Circumventor:

In addition to the address bar, the interface allows the user to set a number of options for each request. These options determine whether the tool will: filter cookies, filter javascript, filter ads, filter the referrer, and show the address bar on the requested page. There is also a link to a cookie management page that allows the user to view and delete cookies and HTTP credentials. The tool by default turns on the options to filter javascript, filter the referrer, and show the address bar at the top of each page.

The address bar, various options, and Circumventor message all take up about 200 pixels of space on most sites, which is a significant amount of the screen real estate (notice in the above screen shot that the address bar has caused the google.com home page to scroll down past the first page).

Circumventor suffers from the same user interface problem as do all cgi proxy systems: the required alternative interface (the Circumventor address bar) raises the strong likelihood of confusion with the browser address bar, which confusion can cause data leakage.

Circumventor uses a self-signed ssl certificate. The self-signed certificate causes IE to present a warning screen to the user the first time a user connects to the site each browsing session. This warning screen includes a strong warning from IE that the destination site is insecure; as such, it is likely that the screen will discourage some users from using the tool.

The startup time for Circumventor is merely the time required to load the tool's home page, from less than a second to several seconds depending on the speed of the server.

G. Server Component

Circumventor relies on volunteers to install and run servers. The project encourages users to install servers to make them available to friends (or to themselves in a different location, for instance a student might install a server at home to user while at school).

The Circumventor installation process is a cumbersome one that will likely discourage most users and have major side effects on the hosting machine. Before installing the Circumventor tool, the user must install two major pieces of software – the perl scripting language and a variant of the apache web server called OpenSA (which seems now to be a defunct project). Installation of both of these packages requires downloading and running their own .exe installation programs, each of which requires the user to navigate a half dozen dialog boxes. Each of the perl and apache programs are large, complex pieces of software that can expose the computer to significant vulnerabilities, especially when the computer user does not understand them.

After installation of the prerequisite programs, the user must run the Circumventor .exe install file. The Circumventor installation process presents the user with a series of unzip screens and black command line boxes with sometimes cryptic text. Ideally, the whole process will just work. If the installation does not work, the only help is a suggestion to send a log file to the project email address.

Once successful, the installation scripts sets up the server to start automatically every time the computer starts, even though it never asks the user for permission to do so, and there is no indication on the desktop that the server is running nor any obvious way to stop the server. It is certain that some users will continue running Circumventor unaware for months and years.

After installation, the server requires little or no administration to maintain, and on most servers bandwidth will cap the usage of the server before it puts significant load on the other resources of the host computer.

By default, the location of each new Circumventor installation is published to a mailing list of Circumventor users, so each Circumventor installation may be used not only by the installer's peers but also by subscribers to the Circumventor mailing list. A host may opt out of publicizing her server address to this wider list. The tool does not explain to the host that a possible consequence of including herself on this list will be to have her IP address blocked in China and other filtering countries. Such blocking might block any other web sites the user is hosting at the same IP address and also might block the Circumventor proxy for the host's intended peer users.

H. Blocking Resistance

Circumventor mainly relies on social networks to avoid blocking. Instead of relying on public proxy lists that a blocking country can snoop, the tool relies on each individual peer in a non-filtered country to setup a Circumventor server for one or more known peers in filtered countries, sending the address of the Circumventor server only to her colleagues and therefore making it very difficult to snoop. Circumventor is also careful to avoid any traffic signatures that might allow a filter to pick out its signature from other HTTPS traffic, including using randomized text in the metadata of its encryption keys.

It may be possible to block Circumventor in the future by recognizing its traffic signature at the router. A filter might for instance recognize that the signing data on its ssl certificates consists of randomly

generated strings or that a single user is sending solely HTTPS requests to the same server over a long period of time with a timing that matches proxy use. Circumventor will have to adjust to these new filters by more carefully obscuring its traffic patterns.

I. Internationalization

Circumventor is available on in English, though the client user interface is likely simple enough to be used by someone who does not read English.

J. Internal Filtering

Circumventor does not perform any content filtering of its own.

K. Security

As mentioned above, there is an inherit security flaw in the necessary alternative browsing interface of all cgi proxy tools. Almost every user will at some time confuse the browser address bar and the cgi proxy tool address bar and potentially leak data by doing so.

Circumventor filters out javascript by default, but this is an option that can be turned off. The latest version of CGIProxy attempts to transform links that are created via javascript when it is not filtered.

The number of users does not strongly affect the security of CGIProxy, since the main mode of usage is for a few peers to use each Circumventor server.

Circumventor runs over HTTPS, which means that all traffic is encrypted via HTTPS. However, Circumventor uses a self-signed ssl certificate for its HTTPS server, which use opens the user to the possibility of unknowingly connecting to a counterfeit version of the server. Also as mentioned in the test results above, Circumventor fails to encrypt some of its requests due to the failure to transform all urls into proxied equivalents.

The Circumventor server keeps a log of all requests, which might be made accessible against the wishes of the server host, for example if the user is required by law to turn them over.

Lastly, the security of any given Circumventor session relies on the trustworthiness of the hosting peer. Ideally, the hosting peer will be well known (and trusted) by the end user. In practice, many social networks are very loose, for instance formed out of acquaintances in online communities. These loose social networks are particularly prone to abuse, including hosting of rogue nodes by filtering countries. The public lists of Circumventor servers distributed by the project through its mailing list are particularly prone to such abuse.

M. Support

The Circumventor project only provides very bare documentation, including a few sentences of installation instructions and a handful of frequently asked questions. Interactive support is available only by emailing the leaders of the project.

N. Development Activity

Circumventor was released in 2003. The stand alone CGI Proxy script was first released in 1996.

O. Community

There is a somewhat active Circumventor community that subscribes to a project mailing list.

P. Code Availability

The code for Circumventor and CGIProxy, in the form of perl scripts, is readable and editable by the

user. However, neither project allows free redistribution of the code.

Q. Development Process

No design documents are available for either Circumventor or CGIProxy other than the code itself.

R. Evaluation

Utility – Circumventor is able to evade the main forms of Internet filtering, but sometimes breaks the layout of complex pages and doesn't handle non-English content well.

Usability – The software can be very difficult to install and lacks documentation. The project's model of relying on direct user support for a complicated installation process without documentation is unlikely to scale.

Security – A serious security hole exists around images embedded within cascading style sheets. The system uses a self-signed HTTPS certificate, and has all the anonymity problems of a peer-to-peer system. Installation of the server requires installation of large, complex prerequisite packages that are likely to intriduce network vunerabilities to the hosting system.

Promotion – The Circumventor product is visible online, but not well promoted.

Sustainability – The project survives on donations, but costs are fairly low as usage is low and there is little additional development on the tool.

Openness – The code for the tool is available with the tool, but not under a free software license that allows redistribution of changes. The developers are open to input and suggestions.

S. Response

The developer of Circumventor submitted the following response:

> The report (originally) stated that "The Circumventor installation process is a cumbersome one that will likely discourage all but expert users", however we have had tens of thousands of users install the Circumventor successfully, the vast majority of whom were self-described computer novices. The install is probably not as difficult for most people as described in the report -- for example, the report says the install "requires the user to navigate a half dozen dialog boxes", but the only thing the user has to do is click "Next" and accept all the defaults.
>
> We ourselves have already published articles admitting that Psiphon would probably make Circumventor obsolete. The longer install process for Circumventor is enough of a reason to recommend Psiphon over it.
>
> However, the longer install process for Circumventor is probably not a sufficient reason for why Berkman Center and similar groups did not promote the Circumventor tool to foreign human rights groups from 2003 to 2007, before Psiphon existed, when Circumventor was the only available option. We did not have the funding or the connections to promote it ourselves, and we reached out to many groups such as the Berkman Center during that period to ask for their help in promoting the tool to overseas users, but all of them refused.
>
> While none of these groups ever provided any official reason for this, the most likely explanation is that the Circumventor was considered "politically incorrect" because Peacefire.org promoted it as a way for users under 18 in the United States to circumvent Internet censorship as well, and "mainstream" anti-censorship groups did not want to be

associated with that. (As noted above, many novice computers used it easily to circumvent Internet blocking at work or at school, and it would have worked just as easily to circumvent international censorship as well.)

The report says that Psiphon is widely used among overseas groups. Virtually all of these groups could have benefitted from knowing that the Circumventor existed as an alternative four years before Psiphon was released.

This is relevant to the present-day situation because the Peacefire site currently lists other methods for evading censorship that are easier to use than either Circumventor or Psiphon (for example, a mailing list for people to sign up to receive new proxy sites by e-mail, where the latest proxy sites are almost never blocked). If mainstream anti-censorship groups withhold information about these tools from foreign groups for reasons of "political correctness" (that is, because these services are also used by Americans under 18 to circumvent Internet blocking), they will be doing a disservice to users in censored countries who could benefit from these services.

The developer of CGIProxy submitted the following response:

This study has some good information, but I'm disappointed in how it evaluates the Circumventor installer package only, and not CGIProxy itself. Most of the criticisms of CGIProxy in the study stem from one of:

a) factually wrong information;

b) considering Circumventor without considering CGIProxy;

c) using a very old version of CGIProxy (such as one included with Circumventor);

d) not understanding how to configure CGIProxy correctly.

There are some significant comments and corrections I'd like to add, without which some parts of the study would be false or misleading -- notable topics include speed, security, ease of installation, page accuracy, and code availability.

I'm trying to keep this concise; details of anything below are available upon request. Feel free to address questions to me at james@jmarshall.com.

1) Regarding the specific “Circumventor/CGIProxy Report” section of the report, the responses below are labeled from A to R, corresponding to the subsections of that section to which they respond. (Not every subsection has a response.)

A. Summary

Note that CGIProxy was initially created to address international filtering in early 1996 (when China and Singapore first announced their intentions to censor the Web), and that has been its primary focus ever since. Installation of CGIProxy is very easy, even automatic, on a Unix/Linux server; Circumventor is only needed for Windows machines.

Some of the confusion in this whole section is related to the confusion between CGIProxy and Circumventor, which are two related but independent projects.

From some testing details, it appears that the latest release of CGIProxy wasn't used, but instead a version dated June 2006 or before. I don't know which versions of CGIProxy are in which versions of Circumventor, so installing from Circumventor could have resulted in an older CGIProxy being tested. As of January 2009, the latest release of CGIProxy is 2.1beta19, available from its web site.

B. Lab Testing Results

When comparing CGIProxy's performance to that of other tools, it's important to compare apples to apples. Most of CGIProxy's CPU usage comes from its support of complex Javascript, unique among CGI-based proxies, which enables arguably the most popular-to-censorees sites on the Web such as Web-based email. However, such support takes a LOT of CPU time on both the server and the browser. When comparing CGIProxy's performance to similar tools without JavaScript support, a fair comparison would be to set $PROXIFY_SCRIPTS=0 in the script, and also to turn off the "remove scripts" checkbox. CGIProxy has gone through much performance tuning in its 13 years, and I think it would compare favorably.

Also, CGIProxy is made to be run in a mod_perl environment for dramatic speed improvements. mod_perl is a common free Apache add-on that, among other things, speeds up Perl CGI scripts. In particular, when comparing Perl CGI scripts to PHP scripts, mod_perl should be used so that the scripts are using the same basic server mechanism. (This is poorly documented on the CGIProxy site, I know.)

Regarding the data leak: I'd like to know the details, so I can fix it -- which page, etc.? CGIProxy supports images in CSS, so if this is happening this is a bug. I could not reproduce this problem.

That embedded content is not always obvious is indeed a weakness in all link-transforming systems. The biggest problem in my experience is non-standard HTML/CSS/Javascript/etc. "features" in browsers, when used by page authors. In the future, I'd like to establish liaisons with browser developers so that CGIProxy can stay on top of these non-standard details.

The shifting of pages half a screen to the right is actually a bug in IE, where in certain cases it _left-justifies_ "centered" content at the center rather than actually centering it. This has been worked around in CGIProxy since version 2.1beta13, released in September 2006. This is not a problem when using Firefox.

In future studies, I hope Firefox is tested along with IE. I can say that the Javascript support in CGIProxy has fewer bugs when using Firefox than IE. This is not from any political decision on my part; it is simply because there have been mature Javascript debugging tools for Firefox for much longer than there have been for IE, so bugs with Firefox have been fixed faster than bugs with IE.

C. In Country Testing Results

See comments for subsection B regarding CGIProxy performance; they apply here too.

Regarding the two URLs that are described as unsupported by CGIProxy-- these pages did not used to work at all, even without any proxy software, but now that they do, they work fine through CGIProxy.

If CGIProxy ever has trouble with Chinese or other characters, please let me know-- UTF-8

content (i.e. international characters) has been supported since late 2006; UTF-16 has been supported since late 2005.

D. Availability

CGIProxy is freely downloadable from the CGIProxy web site, and is also informally but widely distributed via mirrors, email, and other means (since the web site is blocked in countries where the software is most needed).

F. User Interface

CGIProxy has no "banner ad" in its top form, which actually takes up about 80 or 50 vertical pixels depending on configuration. In fact, I did not know that this banner ad was being inserted into Circumventor until I read this report. I agree that 200 pixels is too much, and the banner ad violates my basic style principles anyway.

G. Server Component

The fact that CGIProxy is not Circumventor is especially true here. Circumventor is an installer for Windows machines only. For CGIProxy, there is an online tool (developed elsewhere) that automatically installs it on any (accessible and authorized) server on the Internet, requiring only simple form input from the user. Even without the installer, manually installing CGIProxy on a Unix server requires copying one file into one directory (folder)-- that's it! On existing Windows servers, the same is often true.

K. Security

Circumventor might use a self-signed SSL certificate, but CGIProxy makes no such restriction-- the server owner can use any valid certificate.

CGIProxy does not keep any log of all requests, other than what the Web server may keep. Most servers (including Apache) can be easily configured to turn off such logging.

M. Support

Agreed, external documentation of CGIProxy is not what it should be. When I get the time to work on it, one priority is creating docs specifically tailored to circumvention, in several languages. In the meantime, I do answer almost all of the CGIProxy-related email that comes my way, much of which is from CGIProxy administrators around the world in touch with their own user communities.

Internally, the CGIProxy code is _heavily_ documented, in the hopes of helping users customize the code as needed, as well as learn the technology involved. Over 60% of the Perl program code is comments. From feedback I get, both goals are satisfied.

O. Community

See the comments for subsection M regarding email. Additionally, there are many existing forums for CGIProxy users in various languages on the Web.

P. Code Availability

This is not correct. I heartily encourage people to distribute the code however they can to help the cause of bypassing censorship. In fact, the whole premise of CGIProxy requires that! I do say on my site that CGIProxy is "free for non-commercial use", in the hopes of (non-censorship-related) commercial licenses one day helping CGIProxy pay for its own development. (There is a little of that so far, but not much.)

Q. Development Process

This is technically true, but note that “the code itself” contains extensive design docs within its comments. See my remarks for subsection M regarding code comments.

R. Evaluation

To summarize, responding to each point in turn:

Utility-- This is a result of using old versions of CGIProxy. Newer versions have solved these problems.

Usability-- CGIProxy has an automatic online installer, manual installation on existing servers is also trivial, and there is extensive documentation in the source code.

Security-- I could never reproduce the security hole, and assume it has been fixed. The choice of SSL (HTTPS) certificate is up to the server owner and has little to do with CGIProxy.

Promotion-- Indeed, there is no marketing budget. However, CGIProxy has been downloaded over a million times from the main site alone, through word of mouth alone.

Sustainability-- This is not true. Ongoing development of CGIProxy continues, funded solely by its author, not donations. Costs are high.

Openness-- The code is widely available and well-commented. I don't understand what the objection is here.

I can't help but notice how Circumventor/CGIProxy is evaluated compared to the other package that most closely resembles CGIProxy in type, the top-scoring ONI-sponsored Psiphon. Not to pick on Psiphon (the more anti-censorship tools, the better), but the ratings given don't make sense. For example:

Utility-- Psiphon can't handle JavaScript or Flash, and so not nearly the breadth of pages that CGIProxy can, yet they are ranked equally.

Security-- Psiphon and CGIProxy have the same (alleged) security problems here, yet they are ranked differently.

Promotion-- A Google search on “CGIProxy” yields about 5 times as many results as a search on “Psiphon”, and yet Psiphon is rated much higher because of its “high visibility online”.

Sustainability-- Work on CGIProxy has continued almost uninterrupted since 1996. Work on Psiphon happened for three months, then the project was abandoned two years ago. Yet, Psiphon outscores CGIProxy here too.

Openness-- CGIProxy's source code has always been “freely available, well-documented, and readable”, and the author has always been very “open to input and suggestions”. Source code for Psiphon is NOT available-- check the website. Yet Psiphon is rated much higher than CGIProxy here too.

What gives?

2) Section 2.C.I.b (Theoretical Landscape / Circumvention Methods / Proxying Methods / CGI Proxy, starting on page 14) discusses the caching benefits of HTTP proxies. Note that a CGI-based proxy can work in conjunction with a standard HTTP proxy on the same machine to gain these same performance and cost benefits. CGIProxy has been designed to do this since its early days, and most large installations of CGIProxy do exactly this.

3) Section 2.C.IV (Theoretical Landscape / Circumvention Methods / The Cost of Anonymity, starting on page 18) says tool developers must choose between filtering out cookies and JavaScript for anonymity, at the cost of the Web's functionality, and allowing full access to the Web at the cost of anonymity. However, CGIProxy provides full Web functionality AND anonymity, by altering all cookies and JavaScript network calls to be re-routed back through the proxy.

9. Psiphon Report

A. Summary

Psiphon is a circumvention tool that uses cgi proxying. Psiphon is intended to support circumvention via social networks. The goal of the project is to allow a volunteer in a non-filtered country to host a Psiphon server for a small number of his peers in filtered countries.

Psiphon provides an easy to use client interface that requires no client software installation and an easy to use and install server component. Psiphon provides access to content through all kinds of filters, but the performance of Psiphon is poor for both image heavy and simple, text oriented sites. Psiphon filters out active content, but it fails to encrypt all HTTP requests, leading to unintended data leaks.

Psiphon can be recommended for widespread client and server use, but users should also be aware that some data leaks are possible.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	Psiphon
control	6.8	21.9
500ms delay	13.1	42.9
1000ms delay	20.8	62.5
10% packet loss	17.9	39.6
25% packet loss	26.8	73.4
28.8k bandwidth	66.9	62.3
128k bandwidth	24.9	22.1

In aggregate, Psiphon performed 83% slower than the control connection. Among the nine tested tools, Psiphon was the fifth fastest among the nine tools in the lab testing. The average across all tools during lab testing was 85% slower than the control.

Psiphon was able to retrieve all ten sites through the IP block, DNS block, and keyword block filters. However, the keyword filter did detect one data leak during the response for yahoo.com. During that request, the browser submitted the following clear text request:

```
GET /us.yimg.com/i/ww/them/1/search_1.1.png HTTP/1.1
Host: us.il.img.com
```

This request represents a data leak because it 1) was sent directly to the web server in question instead of to the Psiphon server and 2) was sent in clear text as an HTTP request rather than encrypted as an HTTPS request. Sending the request directly to the server allows a router on the network to discover the identity of the web server the user is browsing. Sending the request in clear text allows a router on the network to determine both the identity of the web server and the particular file being requested. Either of these bits of data could allow a snooping router to find out that a particular user was requesting a sensitive site even though the user believed herself protected.

The cause of this leak was the failure of Psiphon to transform an image embedded in a css stylesheet. To ensure that all traffic or a given page goes through the Psiphon cgi proxy, the script must make sure not only the html for a given page but also all of the supporting css, image, and other supporting files load from the Psiphon cgi proxy server rather than directly from the designated servers. One weakness of cgi proxy systems is that all such embedded content is not obvious (as in this case with the guilty css

image) and that the tool must transform every single link to avoid data leaks.

The Psiphon address bar either mangled itself or disappeared entirely after loading some websites.

C. In Country Testing Results

Following are average response times for requests to about thirty urls in each of four different locations:

	Control	Psiphon
Seoul	10.2	62.28
Hanoi	5	67.62
Shanghai	11.7	84.97
Beijing	6.1	100.77

As with all of the circumvention tools, the tool was much slower in country than in the lab. In aggregate, Psiphon requests were 856% slower than direct requests, making it the seventh fastest of the nine tools tested. The average across all tools during in country testing was 616% slower than the control.

Psiphon successfully bypassed the filters in all tested countries and was not blocked in any country. The site freechina.net failed to load, likely because the site required javascript to function. A few other sites intermittently timed out.

For both the in country and lab tests, the Psiphon server ran without problems for weeks without need for any active administration.

D. Availability

The Psiphon server is available as a free download from the Psiphon website. The Psiphon website is easily findable via a google search.

E. Install

A client Psiphon user need not install any software – he merely needs to browse to the Psiphon url given to him by a Psiphon server host. As a browser-based tool, Psiphon supports all platforms.

F. User Interface

The Psiphon client user interface consists of an address bar into which to enter a url and button to fetch that url. The address bar is included by default at the top of each page returned, so the user can continue browsing through Psiphon simply by typing a new url into the Psiphon address bar. The following shows a user browsing google.com via Psiphon:

In addition to the url box and go button, the address bar includes buttons to visit and manage bookmarks, to load or not load images, and to move or close the address bar.

Psiphon suffers from the same user interface problem as do all cgi proxy systems: the required alternative interface raises the strong likelihood data leakage and general confusion. As mentioned in the testing results, the Psiphon address bar either does not display itself correctly or interferes with the site content on some sites, though its small size makes any such interference minor.

Psiphon uses a self-signed ssl certificate, which causes IE to present a warning screen to the user the first time she connects to the site each browsing session. This user screen includes a strong warning from IE that the destination site is insecure; it is likely that the warning will discourage some users from using the tool.

The startup time for Psiphon is merely the time required to load the tool's home page, from less than a second to several seconds depending on the speed of the server.

G. Server Component

Psiphon relies on a social network of peers in non-filtered countries to host Psiphon servers for their peers in filtered countries. Individual Psiphon nodes require authentication and are not generally available for public use, so using Psiphon usually requires knowing someone in an non-filtered location who will install a server for the particular user.

Installation of the Psiphon server is quick and easy. It requires only running a .msi file and then running the psiphon.exe program that is installed on the desktop. The only input required by the user is a url path for the server.

Once started, the server user interface allows the administrator to start and stop the Psiphon server, to add and remove users, to view the request log, and to set some configuration parameters, including whether to start the Psiphon server on startup and whether to allow clients to view images:

The server requires little or no administration to maintain, and on most servers bandwidth will limit the usage of the server before it puts much load on the other resources of the host computer.

H. Blocking Resistance

Psiphon relies mostly on the opacity of its social network to maintain blocking resistance. In comparison to projects with large, centrally hosted servers, there are so many more Psiphon servers that it is difficult to find and block them all. More importantly, the locations of the servers are not published through any single mechanism but are instead communicated through a variety of technical and non-technical means, including mail, im, irc, and private web pages.

It may be possible to block Psiphon in the future by recognizing its traffic signature at the router. A filter might for instance recognize that the signing data on its ssl certificates is the same for most Psiphon servers; that a single user is sending solely HTTPS requests to the same server over a long period of time with a timing that matches proxy use; or some other such pattern. Psiphon will have to adjust to these new filters as they are implemented by more carefully obscuring its traffic patterns.

I. Internationalization

The Psiphon server is available in English, French, and Russian and is being translated into Russian, Farsi, Arabic, and Chinese. The Psiphon client interface uses only minimal text and so is usable in any language.

J. Internal Filtering

Psiphon has no internal filtering, other than allowing server nodes to filter out all images.

K. Security

As mentioned in the method section, there is an inherit security flaw in the necessary alternative browsing interface of all cgi proxy tools. Most users will at some time confuse the browser address bar and the cgi proxy tool address bar and potentially leak data by doing so.

Psiphon filters out all javascript and active content to protect the anonymity of the user. Many pages rely on the use of javascript for navigation and other core functionality. Psiphon will not work well with

those sites.

Psiphon runs over HTTPS, so all traffic is encrypted. However, Psiphon uses a self-signed ssl certificate for its HTTPS server, which use opens the user to the possibility of unknowingly connecting to a counterfeit version of the server. The Psiphon user guide includes directions for comparing certificate fingerprints to mitigate this risk, but it is unlikely that most users will read the user guide. Also, as mentioned in the test results, Psiphon fails to encrypt some of its traffic.

Each individual Psiphon server keeps a log of Psiphon requests. That log might be made accessible against the wishes of the server host, for example if the user is required by law to turn them over. Each Psiphon server also pings the central Psiphon host to find out its own IP address. It is difficult for many users to determine their own IP addresses, so this feature is an important usability feature. However, the communication with the central server opens up a risk that that data might be compromised in one way or another. Concerned Psiphon server administrators can turn off this ping feature to maintain better privacy.

Lastly, the security of any given Psiphon session relies on the trustworthiness of the hosting peer. Ideally, the hosting peer will be well known (and trusted) by the end user. In practice, many social networks are very loose, for instance formed out of acquaintances in online communities. These loose social networks are particularly prone to abuse, including hosting of rogue nodes by filtering countries.

M. Support

The Psiphon project provides a professional, attractive, and clearly written user guide that discusses not only the technical features of the tool but also the underlying concepts about how it works and what risks the user is taking in using the tool. The guide is available in English and Spanish and is being translated into other languages. The Psiphon project has also undertaken a successful campaign to publicize its tool, resulting in very fast growth in usage of the tool on its release. The Psiphon project is notable among all analyzed projects for the quality of its documentation and outreach work.

The Psiphon project also hosts a fairly active discussion and announcement forum for the tool, where it provides support for users and hosts community discussions.

N. Development Activity

The first version of Psiphon was released on December 1, 2006. Development on the project has been active since the release.

O. Community

The project has a fairly active user community at the Psiphon hosted forums, with which the Psiphon project interacts regularly.

P. Code Availability

The Psiphon code is freely available and redistributable under the GNU General Public License.

Q. Development Process

The Psiphon user guide and presentation available on the web site provide a good high level description of the design of Psiphon.

R. Evaluation

Utility – Psiphon was able to evade all forms of filtering tested, but was quite slow in field tests. Its architecture is such that this speed limitation is not easily addressed.

Usability – The tool is well designed, usable, and well documented. Support is available in several languages from a user community.

Security – Psiphon presents duplicate browser location bars to the user, making data leaks through user error inevitable during extended use. Psiphon leaks data about some requests to the network. Psiphon's peer to peer model relies on social networks of trust, but many psiphon nodes are hosted by unknown peers.

Promotion – The product has been widely promoted and celebrated and is highly visible online.

Sustainability – Unlike server-based solutions, the costs associated with Psiphon are confined to software development and support. Thus far, these costs are borne by funders interested in the social relevance of the tool, and development remains active.

Openness – Code is freely available, well documented, and readable, and the developers are open to input and suggestions.

S. Response

The developers of Psiphon submitted the following response:

> Thank you very much for sharing your assessment of psiphon, which we have found very useful and will take into consideration as we move forward in our planning, research and development.
>
> With respect to the specific issues raised in your assessment, most were known issues when we released version 1.0, but were not considered prejudicial to the release, and all were on the development list for version 2.0 of psiphon, which is presently in development and testing phase. Notably, version 2.0 resolves the javascript issue as well an end-to-end encryption allowing users to access to gmail and streaming media. Indeed, it is fair to say that there are enough substantial alterations to the entire system of psiphon 2.0 to warrant an entirely separate review. We encourage you to do so in the next phase of your research and we would be happy to comply with any requests for information.
>
> We have some minor issues with your assessment of psiphon 1.0 we would like to raise. First, the psiphon project quite conscientiously included a major media outreach and awareness raising strategy that is not mentioned in your assessment. While not a technical component per se, our successful media outreach and awareness raising strategy managed to get psiphon on most of the world's major media outlets, and in doing so helped shed light not only on Internet censorship practices worldwide, but on tools to help citizens get around them (including many of the other tools reviewed by your project). More importantly, it also ensured that many people would hear about psiphon, especially in non-censored locations, leading in turn to a very rapid download rate of the software – all of which is essential to the distributed model upon which it rests. We believe it would be good to assess propagation strategies employed by each of the tools in a comprehensive assessment.
>
> Second, we disagree with a remark you make in your assessment that most users will not likely read the user guide. We spent a considerable amount of time putting together a very attractive and easy-to-follow guide and are working to localize it into multiple languages. We are doing this precisely because we believe that a significant number of people will read the guide, especially if it is effectively distributed with the help of partners worldwide and encouraged by us and others. Our belief is that more users will read guides if they are designed well – something we are proud to have accomplished and which has been widely acknowledged by psiphon user feedback.

Overall, we applaud your effort, and others like it, and hope to see more careful assessments of this sort done in the future.

10. JAP Report

A. Summary

JAP is an anonymity tool that uses a pre-determined form of encrypted re-routing to hide the combination of the source and destination of each data packet from all parts of the network, including the individual proxies. In contrast with Tor, which randomly routes a given data stream among available routers, JAP provides a choice of several re-routing paths through a set series of routers, allowing the user to choose which route will provide the best mix of anonymity and performance and using that same set path for all of the user's connections.

Even though the primary purpose of the tool is to provide anonymity for the user, the tool also is an effective circumvention tool, since the hiding of source and destination combined with the encryption of the content effectively prevents filtering. The JAP re-routing servers are hosted by a small number of organizations certified by the JAP project.

JAP provides an easy to use interface that allows the user to bypass all kinds of filtering. Performance of the tool is poor for both image heavy and simple, text oriented sites. JAP offers additional anonymity by passing traffic through multiple routers but fails to deliver the promised anonymity by hosting the most popular routers in insecure configurations.

JAP can be recommended for use only by users who don't need anonymity or who understand the anonymity implications of JAP network composition and know to choose one of the non-default networks.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	JAP
control	6.8	58.4
500ms delay	13.1	69.8
1000ms delay	20.8	76.9
10% packet loss	17.9	30.4
25% packet loss	26.8	42.1
28.8k bandwidth	66.9	69.6
128k bandwidth	24.9	52.7

In aggregate, JAP performed 126% slower than the control connection, making it the sixth fastest of the nine tools tested. The average across all tools during lab testing was 85% slower than the control.

JAP successfully bypassed all of the IP block, DNS block, and keyword block filters for all of the ten test urls. No data was leaked in any of the requests, and all of the pages loaded successfully. In one instance, the JAP connection stop responding, but connecting to a different set of re-routers fixed the problem.

C. In Country Testing Results

Following are average response times for requests to about thirty urls in each of four different locations:

	Control	JAP
Seoul	10.2	42.78
Hanoi	5	69.97
Shanghai	11.7	84.97
Beijing	6.1	51.57

As with all of the circumvention tools, the tool was much slower in country than in the lab. In aggregate, JAP requests were 655% slower than direct requests. Of the nine tools tested, JAP was the fifth fastest tool during the in country tests. The average across all tools was 616% slower than the control.

JAP was not blocked in any of the tested locations, did not fail to return any pages, and did not cause any display problems with any of the tested pages.

D. Availability

JAP is available as a free download from the JAP website, which is easily findable via a google search. The project will be moving to a tiered hosting model at some point in the future. The tiered model will segregate paying users from non-paying users and provide the paying users with a higher level of network performance.

E. Install

Use of JAP requires the installation of the JAP tool on the client. The JAP tool is a local proxy that accepts connections from the web browser and forwards them on through the re-routing network.

The JAP client is a Java application, so it requires that Java be installed on the computer. The JAP installation program tries to determine whether Java is present and, if not, install it for the user. On the test system used for this report, the installation program failed to install Java and aborted with a cryptic error message. After installing Java manually, the JAP installation worked fine. This installation bug might be fatally confusing for many users. The project provides an alternative installation package that includes a bundled Java which would make this problem moot.

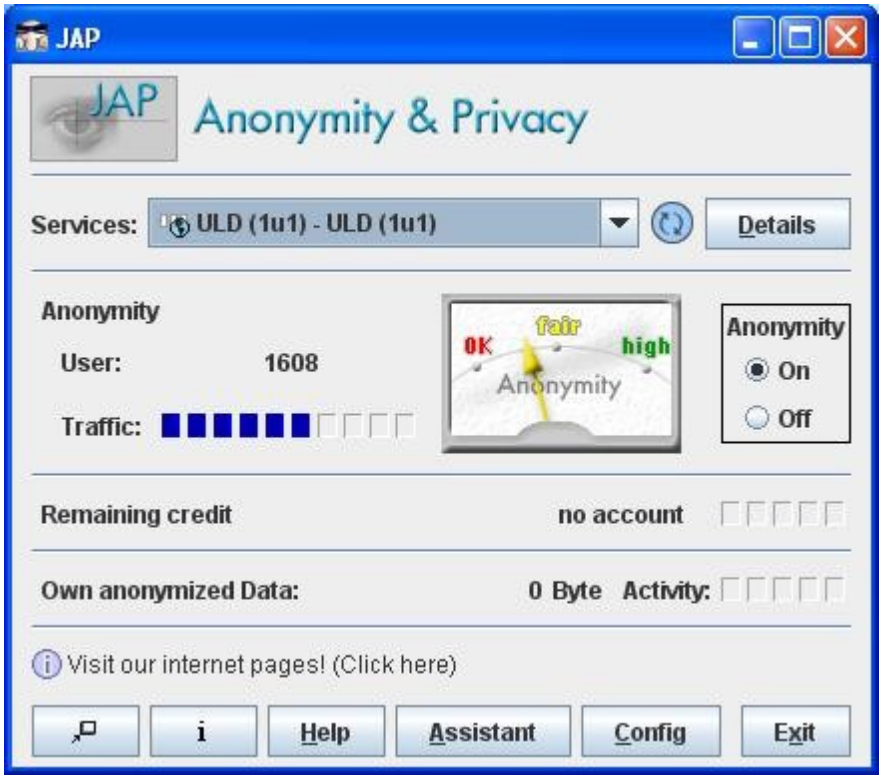
The first time JAP runs, the user is presented with a configuration assistant that guides her through choosing a port for the local JAP proxy, configuring her browser to use the proxy, configuring the non-anonymous mode for the proxy (whether the proxy should refuse to load any pages if it is not able to connect to the JAP network), and configuring her browser to disable javascript and active content. Each of the browser configurations require that the user manually configure the browser herself, though the assistant provides detailed instructions for doing so.

JAP runs on all platforms that support Java, including Windows, OS X, and Linux.

The JAP installation installs a variety of files on the local system, though it also includes an uninstall program to remove the program and its files when needed.

F. User Interface

To connect to the JAP network, the user starts the JAP client. The client presents the user with a small interface that displays the status of the connection to a re-routing network and the level of anonymity for the network once connected. The user interface allows the user to configure the client, including choosing a different re-routing network.



Basic operation of the interface is fairly simple, since the user need only know how to read the display to determine whether she has successfully connected to a re-routing network ("Service" in the above screen) and to change the network if the default network is hanging for whatever reason (which happened once during a few hours of testing the tool). The interface provides a large number of additional configuration options for more advanced users, including the ability to act as a forwarding server for other JAP users who are not able to access a JAP re-routing network directly.

The tool does a good job of prompting the user at various times to keep him informed of relevant security issues, including reminders to disable active content in the browser and the anonymity meter on the front page that shows the level of anonymity granted by the current re-routing network.

Startup time for the client involves both the application startup, which takes a few seconds, and the much longer re-routing network connection time, which can take a minute or longer. JAP does provide the option of turning anonymity off without shutting down the client or reconfiguring the browser, but turning anonymity back on requires the often long wait to reconnect to a re-routing network.

A JAP user will spend the great majority of his time on the browser itself. When using the browser to browse through JAP, the browser behaves completely normally, with the only change visible to the user the decreased performance and the ability to access normally blocked sites.

G. Server Component

JAP relies on a small set of institutional volunteers and service providers certified by the JAP project to host its servers. As such, the JAP server software is not designed or distributed for widespread use.

As noted above, the JAP client includes the option of acting as a forwarding server for other JAP clients that aren't able to access a JAP server directly, but this functionality is an integrated part of the client.

There are currently only a small number of hosting providers for JAP services. As described in the security section below, this scarcity is a problem because the anonymity provided by JAP depends on the

diversity of its service providers.

H. Blocking Resistance

JAP uses a forwarding service built into every client to resist blocking. Clients that cannot directly connect to one of the small number of re-routing networks can instead connect through a forwarding service running on one of a much larger number of clients. These forwarding client addresses are only distributed one a time a time and require solving a captcha (deciphering a string of letters or numbers in a pictures, which is much easier for a person than for a computer) for each client address. This captcha requirement makes it much harder for an automated client to download and block all of the forwarding clients. The blocked client will still need to have unblocked access to a directory server that can provide the client addresses, but these directory servers are much lighter and easier to setup than the core JAP routing servers and so can be proliferated and moved around much more easily.

The obvious cost of this approach is that it will further decrease the already poor performance of the JAP network by funneling the data through a low bandwidth consumer connection.

I. Internationalization

JAP is available in English, French, Czech, Dutch, and Portuguese.

J. Internal Filtering

JAP does not filter the data that flows through its network in any way.

K. Security

The increased security of a re-routing system over a simple single proxy system lies in the fact that no single entity knows both the destination and the origin of the connection. If the re-routing network consists of two routers, only the first router knows the origin of the connection and only the second server knows the destination of the connection. The user needs to trust no single router with the complete information about his connection.

This increased security is negated if the same entity operates all routers on the network or if the entities hosting the routers share information about the connections. The JAP project theoretically certifies the JAP re-routing networks based not only on the identity of the hosting entities but also on the promise of the entities not to cooperate in this fashion.

This increased anonymity increases the cost to provide the same level of performance to the same number of users as a single proxy system. It costs roughly twice as much in aggregate to host to servers that must download and then upload each HTTP request as it does to have a single proxy do the same work. This increased cost is passed on to the user (or the funder) in the form of increased monetary cost or decreased performance.

Unfortunately there is a serious hole in the current implementation of the JAP re-routing networks. The JAP project itself hosts all of the routers in most of the available re-routing networks. In fact, the overwhelmingly most popular JAP network is the “Dresden-Dresden” network, which the project describes in its documentation as a research network that they use specifically to correlate connections between the two participating routers. The particular bit of documentation that describes the “Dresden-Dresden” network warns users that it should only be used for research uses. However, there is no indication in the client interface that there is anything unsafe with this network, and this particular network on one particular day accounted for 3915 out of 6104 total active JAP users. The vast majority (1638) of the remaining users were connected to the “ULD-ULD” network, both of whose nodes are run

by the “Independent Centre for Privacy Protection”.

One possible explanation of the preponderance of these insecure re-routing networks is that all of the securely mixed networks were setup to require payment as part of the testing process for JAP's transition to a mixed tiered free / pay system. Even though the payment system is still under testing and so requires no actual payment yet, using one of the payment required servers still requires registration by the user, setting up a much higher barrier of entry for the user than using one of the free (though less anonymous) servers (this registration requirement can be especially burdensome among the privacy conscious users who are JAP's audience). But even if the overwhelming use of the non-anonymous re-routing networks is an artifact of the current, temporary transition to a tiered payment system, the tool should warn the user when she connects to an unsafe network.

JAP itself does not filter out javascript or active content, but it actively encourages the user to configure his browser to disable such content and provides instructions on how to do so.

As discussed in the above section on re-routing, JAP is theoretically vulnerable to various kinds of timing attacks that involve correlating the timing of traffic coming into an entry node and traffic coming out of an exit node.

M. Support

JAP has an excellent self-documenting configuration wizard that guides users through the initial configuration of the system. It also provides clear, comprehensive help files integrated into the client application and a user guide and frequently asked questions document on its website. The project also host an active user support forum on its website.

N. Development Activity

JAP was first released in 2001 and is under active development.

O. Community

The JAP project hosts an active user discussion forum on its website, in which the project leaders participate.

P. Code Availability

The code for JAP is freely available for use and redistribution under the BSD license.

Q. Development Process

JAP is largely a research project run by a number of cooperating faculty. As such, the project has published a number of detailed academic papers about various attributes of its design.

R. Evaluation

Utility – JAP was able to evade the major forms of content filtering but was very slow in field testing.

Usability – JAP provides a functional but complex user interface that may confuse some users. JAP includes a java installation, which did not install cleanly for the report testers.

Security – The default behavior of the product defeats its own security architecture and is vulnerable to timing attacks.

Promotion – The product is basically research software and will not be widely promoted until more progress is made on developing a network of servers and developing a financial infrastructure to support the product.

Sustainability – The developers are aware of the need to create a revenue stream for the service and are exploring premium services which could sustain the costs of maintaining routing servers.

Openness – Code is available for the product and developers were extremely cooperative about discussing operations of the system.

S. Response

The developers of JAP submitted the following response:

> In your study, you did not mention which cascades you did use for testing. Was it Dresden? Was it CCC? Or even payment Cascades? This does not get clear. The attributes of the system, e.g. performance, highly depend on the chosen cascade and on daytime. Users can choose the cascade they want, so choosing the slowest cascade for testing the performance may be no good idea. In our next product release which will come in the following two weeks, users will moreover get detailed performance information about each service.
>
> We have reports from several countries in the world (UAE, Iran,Tunisia,...) that most of the JAP/AN.ON services and the website are blocked. You did not mention that in your study.
>
> In order to close the security issues you mentioned, we have, in the scope of the commercialisation as "JonDonym-Service",
>
> - strongly penalised singe-hop-Mix-Cascades ("cascades" with only one Mix and / or one operator). The penalty results in showing to the user that they are not very secure, at least not so secure as on a distributed cascade. Moreover, users get a constant warning on single-hop-cascades.
>
> - introduced a lot of cascades with two or three operators, spread over different countries.
>
> As better cascades will have a strong advantage on getting users, we have "persuade" the free operators mentioned (Dresden, ULD) for form better-suited cascades (except for the Dresden main mix). Moreover, there is no more such a thing as a "Default-Cascade". They are all automatically loaded on JAP start and one of them is chosen at random.
>
> The mentioned vulnerability for timing attacks is a general attack on all existing anonymisation systems. It may be repelled by a lot of constant dummy traffic and other really sophisticated techniques, but research and development on this matter is still going on. We do not know of anyone that has (shown) a suitable real-time solution for this problem.

11. Tor Report

A. Summary

Tor is an anonymity tool that uses randomized, encrypted re-routing to hide the combination of the source and destination of each data packet from all parts of the network, including the individual proxies. In contrast with JAP, which routes a given data stream through a fixed, predetermined set of routers, Tor randomly assigns a set of routers to each connection. No single Tor proxy knows the origin, the destination, or even what other proxies are involved in the network other than the ones immediately connected to it.

Even though the primary purpose of the tool is to provide anonymity for the user, the tool also is an effective circumvention tool, since hiding of source and destination and encrypting the content effectively prevents filtering.

The Tor re-routing servers are hosted by volunteers. Anyone can host a Tor server – the anonymity of the connection relies not on the trustworthiness of the individual servers but rather on the network design, which prevents a given router from knowing both the origin and the destination or even which other routers it would need to cooperate with to get that information.

Tor provides an easy to use interface that allows the user to bypass all tested forms of filtering, though the performance for international connections is very poor for both image intensive and simpler text oriented sites. Tor provides strong anonymity only if the user is careful to submit data to HTTPS protected servers.

Tor can be recommended for widespread circumvention use, but only for expert users concerned about anonymity. Increased use of the tool might trigger blocking that Tor is not yet ready to defend itself against.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	Tor
control	6.8	61.3
500ms delay	13.1	74.1
1000ms delay	20.8	93.1
10% packet loss	17.9	35.7
25% packet loss	26.8	78.1
28.8k bandwidth	66.9	97.1
128k bandwidth	24.9	79.7

In aggregate, Tor performed 193% slower than the control connection, making it the eighth fastest tool of nine tools tested. The average across all tools during the lab tests was 85% slower than the control.

Tor successfully bypassed all of the IP block, DNS block, and keyword block filters for all of the ten tested sites, and no data leaks were found for any of the sites. One site (baidu.com) failed to load initially during the lab performance tests but loaded successfully in subsequent follow up tests.

C. In Country Testing Results

Following are average response times for requests to about thirty urls in each of four different locations:

	Control	Tor
Seoul	10.2	70.38
Hanoi	5	84.86
Shanghai	11.7	100.2
Beijing	6.1	112.1

As with all of the circumvention tools, the tool was much slower in country than in the lab. In aggregate, Tor requests were 1016% slower than direct requests. Tor was the eighth slowest of the nine tools tested in country. The average across all tools was 616% slower than the control.

Tor was not blocked in any of the tested locations, and it successfully loaded all of the tested urls through the various country filters.

D. Availability

Tor is available as a free download from its web site. The Tor web site is easily findable via a google search.

E. Install

To use Tor, the user must install the Tor client. The Tor client itself consists only of a command line socks (TCP/IP) proxy that routes generic TCP/IP traffic through the Tor network. Tor recommends that web browsing users install a bundle of tools: Tor, Vidalia, and Privoxy. Vidalia is a graphical interface for Tor that allows the user easily to start, stop, and monitor his Tor connection. Privoxy is a privacy proxy that filters out ads and other offensive content from HTTP responses and provides the browser with an HTTP proxy interface (in front of the generic TCP/IP socks interface presented by Tor).

The bundled installation package available from the Tor website installs all three of the above tools in a simple, minimal Windows install process. After installation, Tor, Vidalia, and Privoxy are all started, and Vidalia and Privoxy icons are installed into the task bar

The download page on the Tor website includes about 350 words of warnings and disclaimers about what kinds of anonymity it does and does not provide. The warnings are all valid and important for users to know, but the length of the text alone may be enough to dissuade some users from using Tor.

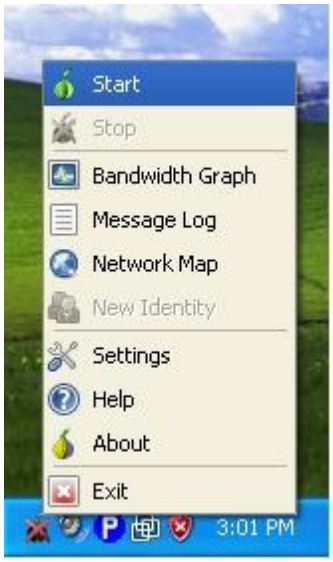
The project includes a page of installation instructions that walk the user through the installation process, but this page lacks instructions for configuring IE to connect through Tor, which configuration will be the most difficult part of the install for many users. Instead, Tor recommends the use of the Firefox browser, which has support for a graphical button that automatically connects it to Tor. Most users do not have Firefox installed and so will need either to install Firefox to use the recommended Torbutton or to manually configure IE.

Tor is available for the Windows, OS X, and Linux platforms.

F. User Interface

The primary user interface for Tor when used with IE is Vidalia. As part of the installation process, Vidalia installs itself into the Windows task bar, where it remains through reboots. Vidalia, Tor, and Privoxy all automatically restart themselves when the user logs in to the client computer.

Vidalia is the user's primary interface with the bundled application. Using the Vidalia task bar menu, the user can view various kinds of information about the connection to the Tor network, but these are mostly only of academic use. Most users will only ever need to use the task bar menu to start and stop the Vidalia tool, and most will not need do that.



Vidalia also provides a configuration screen with some basic configuration options, including the option to setup and configure the client as a server node within the Tor network.

For most users, using Tor after the initial configuration will consist merely of using the standard browser interface. The only difference the user is likely to notice using the browser connected to Tor is that performance will be slower and blocked sites will become accessible.

Tor takes from one to thirty seconds to connect to the Tor network once it starts up. For IE users, the only way to switch between direct browsing and browsing through Tor is manually to edit the IE HTTP proxy configuration. The Torbutton extension to Firefox offers the ability to toggle the Tor connection on and off through a toolbar button.

G. Server Component

The viability of the Tor project depends on sufficient volunteers agreeing to host Tor servers. Tor relies on the generosity of volunteers to host enough client servers to provide sufficient resources for the network. The relatively poor performance of Tor in comparison to other circumvention tools is partly a reflection of the difficulty of recruiting enough volunteers and partly a reflection of the comparatively low bandwidth of volunteered consumer connections.

To install a Tor server, a user needs to install the standard Tor bundle and use Vidalia to configure Tor as a server. The Vidalia interface allows the user to limit the bandwidth used by the server, to serve only as middle-man (see below), and to setup an IP block filter that restricts which sites clients can access through the server.

As with running any proxy server, anyone who runs a Tor server that connects out to a destination web site runs the risk of being associated with bad behavior. A Tor client might, for example, use the server to connect out to a server hosting training information for terrorists. In such a case, the server host could be identified as the one guilty of the bad behavior. Tor is the only proxy tool in this study that relies on volunteers hosting proxy servers for strangers (Psiphon and Circumventor use volunteer hosts, but they are geared toward hosting within social networks of peers). To allow users who want to avoid this risk to host servers, Tor allows the user to configure the server in middle-man mode, which makes the server that only route data between two other Tor servers, never out to the external network. Tor also allows the user to control which sites her server will connect to, as discussed in the internal filtering section below.

H. Blocking Resistance

Tor is currently defenseless against blocking. The IP addresses of all Tor servers are published freely, so anyone who wants to block Tor needs only to block those published IP addresses. To date, Tor has not garnered enough attention from blocking countries to get itself blocked, but it must consider what steps to take should it grow enough to warrant blocking.

The Tor project is actively working on making blocking Tor more difficult by allowing each client to volunteer as a low bandwidth bridge to the larger Tor network. The addresses of these low bandwidth bridges would not be published and so would not be blockable. The problem of blocking Tor would thus be one merely of distributing each of the large number of these individual bridge addresses. This approach is similar in concept to the UltraReach approach of hiding the IP addresses of its core proxy servers behind a much larger set of constantly moving forwarding servers.

The obvious and large drawback to this approach is that it will limit the performance of Tor even more. For many if not most users, Tor is already prohibitively slow. Adding an additional, low bandwidth choke point to the network will make the performance much worse.

I. Internationalization

Vidalia, the graphical front end to Tor, is available in 15 different languages, including English, Chinese, and Farsi.

J. Internal Filtering

The host of a Tor server can configure Tor to filter out any set of IP addresses and/or ports. The network attempts to route around that filtering, however, to find a server that is willing to fulfill the request. Theoretically, this setup allows individual servers to maintain control over their own outgoing connections while allowing users to obtain any information through the overall network. In practice, if many servers filter out the same sorts of traffic, the performance for that sort of traffic will degrade severely.

K. Security

Re-routing tools trade decreased performance for increased anonymity. The cost of having multiple servers route the same traffic (instead of just a single proxy) is that the number of servers required to handle the same amount of traffic is multiplied by the number of hops within the re-routing network. So a connection that goes over a three hop re-routing network will require three times the server resources (including total bandwidth) as a connection that goes over only a single proxy. The user gains the increased anonymity of the multiply routed connection in return for the decreased performance.

The Tor approach to security is to rely on the design of the network rather than on the trustworthiness of the individual servers for anonymity. The network is designed to be hosted by a large number of volunteers, any one of which knows only a few of the others. The only way to build a complete log of traffic over the network is for all of the individual servers in a given re-routing path to cooperate with one another. Since the volunteers hosting the servers in any given path are unlikely to know one another, the risk of such cooperation is low. This approach solves the problems JAP has with having to trust a relatively small number of server hosts not to cooperate with one another.

The first cost of this approach is in decreased performance, since there are more clients than server volunteers and since the server volunteers are mostly hosted on relatively low bandwidth consumer connections. The second cost is that malicious server hosts can disrupt users in ways not directly related to anonymity. For example a malicious exit server could insert its own html into a page.

The anonymity of a given user increases with the number of users on the Tor network, since increasing the number of users makes it more difficult to identify a single connection from the network with a single user.

Tor itself does not filter out javascript or active content, but it actively encourages the user to configure his browser to disable such content and provides instructions on how to do so.

As discussed in the above section on re-routing, Tor is theoretically vulnerable to various kinds of timing attacks that involve correlating the timing of traffic coming into an entry mode and traffic coming out of an exit node. A recent, widely publicized paper by a team at the University of Colorado falls into this class of timing attacks. While theoretically possible, these attacks have only been shown to be possible only on a small scale or at the cost of making themselves visible to the Tor project leaders.

Of more concern is the possibility of an exit node hijacking the content of a connection. With every connection a user makes, the user must trust the exit node for the given connection to return the page requested by the user. A rogue exit node could replace its own content within any HTTP (though not HTTPS) request. Such content replacement could be used, for example, to replace the destination location of a data form, so that the exit node could redirect submitted data to its own server before redirecting the request onto the original destination. Since anyone can host a Tor exit node, it would be very simple for an entity such as a filtering country to run a handful of its own Tor exit nodes and use them to capture the identities of many Tor users. A Tor user can protect himself from rogue exit nodes by submitting data only through HTTPS connections, but even expert users will have difficulty following such an HTTPS policy.

M. Support

The Tor project provides an extensive set of user documents and frequently asked questions, as well as a mailing list and irc channel where users can support each other.

N. Development Activity

Tor was first released in 2002 and has been under active development since.

O. Community

The Tor project has an active user community that gathers on project hosted mailing lists, on a well developed wiki page, and in various pieces of supporting software.

P. Code Availability

The code for Tor is freely available and redistributable under the BSD license.

Q. Development Process

The design for Tor is exhaustively detailed through a series of design papers available on the Tor web site.

R. Evaluation

Utility – Tor was able to evade the tested forms of censorship, but was the slowest of all tools tested.

Usability – The installation of Tor is quite complex, requiring multiple packages to be installed. But the process is well documented and supported by graphical tools, and tools are available in a wide range of languages.

Security – Tor is the most secure of the products we studied, but is still vulnerable if users fail to understand what the tool does and does not block and what behaviors they must avoid using the tool.

Promotion – The product is well promoted within the US and Europe and has a strong web presence.

Sustainability – The project relies on universities and other major bandwidth customers to share bandwidth and provide server space. Thus far, a large number of universities have been willing to participate. However, there is no sustainable model for core support of the tool.

Openness – Source code is freely available and widely studied in the security community. The developers are extremely open and willing to discuss operations of the system.

S. Response

The developers of Tor submitted the following response:

> Since the publication of the report, the Tor Project has responded to and improved a number of identified weaknesses. The main focus has been on usability of the software. Tor is easier to understand, configure, and install. We've worked on finding translators for the various parts of the suite of tools that comprise Tor. We developed and enhanced a Firefox plugin called Torbutton. The current Torbutton mitigates all known browser-based anonymity attacks. Torbutton is included in our bundles and is automatically installed into the user's Firefox browser configuration. Torbutton can be found at torproject/torbutton (https://www.torproject.org/torbutton/) or Mozilla/torbutton (https://addons.mozilla.org/en-US/firefox/addon/2275).
>
> In response to the demand for portability and ease of use, we created the Tor Browser Bundle. The Tor Browser Bundle lets you use Tor on Windows without needing to install any software. It can run off a USB flash drive, or any portable media, and comes with a pre-configured web browser and is self contained. The Tor IM Browser Bundle additionally allows instant messaging and chat. The bundle contains the Tor controller Vidalia, a portable version of Firefox, a caching http proxy called Polipo, the instant messaging client Pidgin with Off The Record client to client encryption for secure chats, and of course, the Tor proxy software. The Tor Browser Bundle can be found at torproject/torbrowser.
>
> There is an arms race between censors and circumvention tools. We've developed a number of features to prepare for the next few steps in this race. Some of these features involve the ability to use non-public relays and normalization of the Tor traffic to allow those in content denied environments the ability to continue to reach the information and communities they need. Much more research and experimentation needs to occur in the next few years to further enhance these features.
>
> As part of its dedication to transparency and openness, The Tor Project has published its three year development roadmap, focused on providing anti-censorship tools and services for the advancement of Internet freedom in closed societies. While Tor's original goal was to provide online anonymity, many people around the world use Tor to get around Internet censorship, as well. The roadmap is focused on providing anti-censorship tools and services for the advancement of Internet freedom in closed societies. This 3 year roadmap can be found at https://www.torproject.org/press/2008-12-19-roadmap-press-release.

12. Coral Report

A. Summary

The Coral Content Distribution Network is a distributed hosting system that uses a large set of caching proxies located around the world to redistribute web content. To access any web content via Coral, a user needs just to add ".nyud.net:8080" to the end of any url. The Coral name servers use a peer to peer DNS layer to direct each name request to location of a proxy with the requested content. Coral is not intended to be a circumvention tool specifically but is instead meant to provide general purpose distribution of content, for example allowing web site publishers to provide high bandwidth access to multimedia content without having to buy lots of bandwidth. The caching proxies that serve content for the Coral network are all hosted by the Coral project.

Coral provides an interesting model for how a distributed hosting tool might provide circumvention technology. But it is not ready for widespread circumvention use. There is no intuitive interface for the system, there is no encryption to protect the user's privacy, and the tool fails to load much of the filtered content.

B. Lab Testing Results

Following are average response times for requests to the same ten web sites in a number of different simulated network conditions:

	Control	Coral
control	6.8	39.5
500ms delay	13.1	37.5
1000ms delay	20.8	64.4
10% packet loss	17.9	49.8
25% packet loss	26.8	80
28.8k bandwidth	66.9	82.1
128k bandwidth	24.9	52.2

In aggregate, Coral performed 129% slower than the control connection, which is the seventh fastest of the nine tested tools. The average across all tools was 85% slower than the control.

Coral successfully bypassed the IP block and DNS block filters for all sites. However, because Coral requests are not encrypted, the keyword block filter blocked all Coral requests.

C. In Country Testing Results

Following are average response times for requests to about thirty urls in each of four different locations:

	Control	Coral
Seoul	10.2	64.47
Hanoi	5	81.35
Shanghai	11.7	(no value)
Beijing	6.1	(no value)

As with all of the circumvention tools, the tool was much slower in country than in the lab. In aggregate, Coral requests were 784% slower than direct requests, making it the sixth fastest of the nine tools tested in country. The average across all tools was 616% slower than the control.

Coral was not blocked in any of the locations; however, the Coral network did not respond during in country testing in Shanghai and Beijing, so no values are available for those locations. The Coral network was tested from multiple U.S. locations in conjunction with the Shanghai and Beijing tests, and

it failed to respond from the U.S. locations as well, so it is most likely that the problem was related to the Coral network as a whole rather than to specific blocking of the tool by China.

For all tests, Coral failed to load many images and css files for many of the sites. The cause of these load problems is that Coral does not transform any of the content it proxies, so while the main html of a page can load through filters, supporting images and other files are still requested directly and therefore vulnerable to blocking.

In response to a request for "http://hrw.org", coral returned the following page:

> Error when attempting to use the Coral Content Distribution Network (http://www.coralcdn.org/).
>
> The hostname specified in the Coralized URL has been blacklisted from the system.

D. Availability

Coral itself is free in the sense that it requires no software to run.

E. Install

Coral requires no software other than the browser.

F. User Interface

To use Coral, the user just appends ".nyud.net:8080" to the end of any url. For example, to retrieve google.com via Coral, just enter "google.com.nyud.net:8080" into the browser. Coral will return the address of a Coral node that contains the requested content (or that will fetch the content in real time).

Coral does not perform any transformations on the content returned by the remote server, so images, css files, and other supporting material will still be retrieved directly from the server (google.com in the above example).

There is a plugin for Firefox that transforms image locations for a given page into their Coral equivalents, and another plugin that transforms all of the links on a given page into Coral links. In casual testing, neither of these plugins proved sufficient nor reliable enough to make Coral generally useful for circumvention.

The dominant uses for Coral are to access cached versions of sites that are overloaded by too much traffic and to publish high bandwidth content. In the circumvention context, these modes can be helpful in allowing access during a filtering motivated denial of service attack and in allowing a publisher to provide an alternative location for content that might be filtered. These uses are more reasonable for Coral than the regular browsing mode of the other tools included in this study.

G. Server Component

The Coral project hosts all of its own servers on the Planet Lab network (http://www.planet-lab.org).

H. Blocking Resistance

Coral does not see itself as a circumvention tool and so includes no blocking resistance. Even though Coral was not blocked in any of the locations we tested, blocking it would require only blocking DNS lookups for the nyud.net domain or alternatively blocking the publicly available addresses of the Coral servers.

I. Internationalization

Coral itself has no user interface, so there is no need for internationalization.

J. Internal Filtering

During the in country tests, Coral was found to be internally filtering hrw.org, but hrw.org was found not to be blocked during subsequent lab tests.

K. Security

Coral should not be used by users who are concerned about privacy, since it uses no encryption. Requests to and responses from the Coral network are in clear text and are therefore easy to snoop anywhere on the network between Coral and the requesting client.

M. Support

The Coral project publishes some simple support documentation on its website, though usage is simple enough that support is minimal. The project also runs a user mailing list that gets some support requests.

N. Development Activity

The Coral project published its first release in March 2004 and has been under active development.

O. Community

The Coral network has a small user community that discusses Coral on a low volume coral-users mailing list hosted by the project.

P. Code Availability

The code for Coral is freely available and redistributable under the GPLv2 license.

Q. Development Process

Coral is an academic research project and so has resulted in a number of highly detailed papers on its design and performance. As well, the project's web site provides a great deal of detail about the design of the tool.

R. Evaluation

Utility – Coral is slow for all sites, is blocked in some countries, and is often down altogether. Coral lacks encryption, so it cannot circumvent keyword based filtering.

Usability – Coral's base use requires users to type a complex URL. It does not handle subsequent link requests well and sometimes fails to format pages correctly.

Security – Because Coral does not encrypt connections, it fails to provide any meaningful security to circumvention users.

Promotion – The system is not widely promoted for circumvention use.

Sustainability – Because the system relies on a network of mirror servers, it can be expensive to operate. The limited availability of the system suggests that volunteer resources may be scarce.

Openness – The Coral developer was mostly approachable but also mostly uninterested in use of the tool for circumvention. The code for the system is not published.

S. Response

The developer of Coral declined to submit a response.

13. Hamachi Report

A. Summary

Hamachi is a Virtual Private Networking tool designed for easy peer to peer use. As a VPN tool, Hamachi allows users to connect their computers together into virtual local networks using encrypted tunnels over the wider Internet. Hamachi is not designed primarily for circumvention use but can act as a circumvention tool by providing computers in remote countries access points in non-filtered countries. Hamachi is a peer-to-peer tool, hosting none of its own servers but allowing an individual in a non-filtered country to run a server for a peer in a filtered country.

Hamachi demonstrates the possibility of developing an easy to use interface for a fully functional VPN system for circumvention use. It also demonstrates the feasibility of providing easy, zero configuration access to servers behind consumer firewalls and routers. However, Hamachi is not ready for widespread circumvention use due to the instability of the tool and the lack of documentation and support for circumvention use.

B. Lab Testing Results

Hamachi was unable to complete the lab performance tests due to instability. The server consistently crashed or hung after only a few requests.

The only test the tool was able to complete was the lab filtering test. Hamachi successfully bypassed all of the IP block, DNS block, and keyword block filters to load all of the test web sites.

C. In Country Testing Results

Hamachi was unable to complete the in country tests due to the instability of the server.

The instability of the tool was caused at times by the entire server program crashing and sometimes by the HTTP proxy component of the tool hanging and requiring a reboot of the server program to respond. Hamachi declined to provide usage statistics for this report, but the tool is widely popular among gamers, who use it to create virtual local networks for multiplayer games. The popularity of the tool indicates that many users find it stable for every day use. Further testing could be done to determine the cause of the stability, but the lack of documentation or developer support for circumvention makes the tool a poor choice for circumvention use in any case, so any further work along the direction of a personal VPN would be better spent investigating improving another tool or developing a new one.

D. Availability

A basic version of Hamachi is available as a free download. A premium version is available for five dollars per month. The most important distinction between the two versions is that the free version has a 250 megabyte cap on HTTP proxy traffic.

E. Install

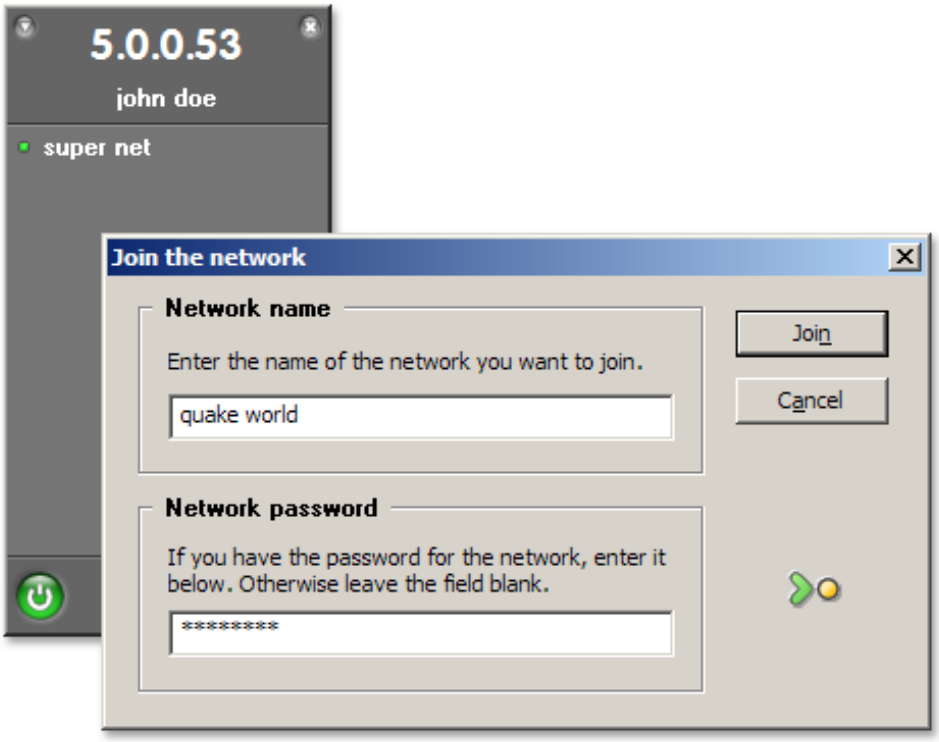
Hamachi requires the installation of client software. The client installation package is a standard Windows installer that requires minimal interaction by the user. Once the software has been installed, the user must join the Hamachi network of a peer to use the tool for circumvention (see below section on the user interface for more information on Hamachi networks). Finally, the user must configure her browser to use the address of the Hamachi peer as an HTTP proxy.

Hamachi runs only on the Windows platform and installs a range of files and registry entries on the client machine.

F. User Interface

The strength of the Hamachi tool is its simple user interface for setting up and joining virtual networks. There are a number of powerful VPN tools on the market, but they are all developed for use by institutions to setup virtual networks for individuals to join (to allow a corporation to give employees off campus access to the company intranet, for example). The result is that the tools are very powerful but very complicated to use, especially on the server side, and also mostly very expensive.

Hamachi trades sophistication for a simple user interface, allowing even novice users easily to create virtual networks. To create a virtual network, a user just chooses the create network option from the networks button on the main Hamachi interface and enters a globally unique name for the network. Hamachi hosts a central directory server that registers the chosen unique name with the requesting Hamachi client. To join that network, any other client needs know only its unique tag (and a password if the network requires authentication); to join a network by its name, the client connects to the central directory server and finds out the location of the client hosting the network. It's hard to overstate how much simpler this process is than the process for setting up and connecting to a virtual network in, for instance, the built in Windows VPN tool, Cisco's VPN tool, or the free Linux VPN StrongSWAN.



In addition to the user interface proper, Hamachi transparently handles the problem of connecting through firewalls and routers that use natural address translation. Running any server application (including Psiphon, Circumventor, Tor, and other circumvention tools that use peer-to-peer hosting) on a consumer Internet connection can be difficult because most consumer Internet connections these days include 1) restrictive firewalls that by default prevent most incoming traffic and 2) natural address translation, which allows many computers to connect to the Internet through a single IP address at the cost of requiring manual configuration for incoming connections to the computers. All of the peer hosting tools discussed in this report require manual configuration of these routers and/or firewalls for many home users to setup servers, which configuration is difficult to support because of the vast range of such products.

Hamachi solves this problem through the innovative use of an intermediary server. The two Hamachi clients connect to one another initially through the intermediary server but then continue the connection directly with one another, without routing traffic through the intermediary. As with the simple Hamachi user interface, it is difficult to overstate the usability value of this functionality over requiring users to configure their routers and firewalls themselves.

Connecting the user to a Hamachi virtual network, however, is not sufficient to provide circumvention functionality. The user must also configure her browser to use the peer computer on the network as a proxy in IE. Hamachi is mostly not intended by its creators for use as an HTTP proxy, so this step is not well documented or supported from within the interface.

On startup, Hamachi takes from one to thirty seconds to connect to the central directory server, find the location of any virtual network peers, and connect to those peers. Switching between anonymous and direct browsing requires manually adjusting the configuration of the browser.

G. Server Component

There is no separate Hamachi server application – the single Hamachi program serves as both a client and a server. As described above, using Hamachi as a virtual network server requires only that the user click on the option to create a network and then choose a unique name and a password for the network.

Giving a peer access to the Internet through the Hamachi virtual network requires an additional step of either setting up an HTTP Proxy to run on the virtual network or setting up the virtual network to have access out to the Internet

Hamachi includes a basic HTTP proxy server built into the standard installation. Unfortunately, configuration of Hamachi to run the HTTP proxy server requires editing an obscure configuration file to add few configuration directives that are documented only in the Hamachi discussion forums rather than in the core documentation. The ease of setting up the HTTP proxy server would be hugely increased by making it a simple toggle switch in the graphical interface.

It is possible as an alternative to the HTTP proxy to setup Windows to allow Internet connection sharing over the Hamachi virtual network. Clients connecting to the virtual network could then configure their computers to route all Internet traffic through the virtual network. This method provides tremendous flexibility for the client at the cost of additional liability for the server peer, since the client gains completely unfettered access to the Internet through the server (instead of merely HTTP access).

Another alternative for outgoing access is to setup the Hamachi server to work with a standalone HTTP, socks, or other proxy server.

H. Blocking Resistance

Hamachi's reliance on a central directory and mediation server makes it very vulnerable to blocking – blocking either of those servers is sufficient to block the tool entirely. The Hamachi developers are uninterested in supporting circumvention use and so have no plans for adding blocking resistance to the tool.

I. Internationalization

Hamachi is available in English only.

J. Internal Filtering

Hamachi no includes no mechanisms for internal filtering of content.

K. Security

Hamachi provides only the mechanism for the creating the virtual network over an encrypted tunnel. The user is still responsible for filtering out active content on any browsed web pages (or the host must setup a proxy to filter out that content).

As with all peer-to-peer hosting networks, the client must trust the peer hosted server not to snoop on connections, hijack content, or otherwise interfere with the connection.

M. Support

A Hamachi user guide is available on the Hamachi web site. The user guide provides clear instructions for setting up a virtual network with the tool and using the virtual network for file sharing and iTunes music sharing. There is little documentation of setting up a server for circumvention use – the user will have to search through the support forum for the relevant bits of instructions. Hamachi hosts an active user forum for support and general discussion of the tool.

N. Development Activity

The Hamachi tool was first released in 2004. It is under active development and will continue to be under active development for at least the next year.

O. Community

Hamachi hosts an active community forum on its web site.

P. Code Availability

The code for Hamachi is proprietary and not available to the public.

Q. Development Process

The Hamachi developers declined to share any information about their development methodology for this report.

R. Evaluation

Utility – The server component of Hamachi was unstable and was not able to perform in field tests.

Usability – The Hamachi tool provides a slick, very easy to learn interface that handles the complex task of setting up a peer to peer VPN. The tool works through the common home routers and firewalls that require manual configuration with all other peer to peer servers.

Security – Because Hamachi implements a secure VPN, the only major security vulnerabilities exist at the server side.

Promotion – Hamachi is not designed as a circumvention tool. It is widely available, however, and might be usable for circumvention if more stable.

Sustainability – Hamachi sells a premium version of the product which sustains software development.

Openness – Not only was code unavailable and developers uncooperative, they have explicitly discouraged the product's use for circumvention.

S. Response

The developers of Hamachi requested that we not communicate with them about this report.

14. Conclusions

The basic technology of circumvention is relatively easy and well known – it requires only that a tool encrypt the traffic and redirect it through an intermediate server in a non-filtered country. The difficulty of developing effective circumvention tools lies in making them usable, fast, secure, and blocking resistant. All of the circumvention and anonymity focused tools included in this study manage to circumvent both lab and in country filters and, notably, to provide an effective and reasonably easy to use interface. This point should be stressed – that all of the circumvention and anonymity tools included in this study succeeded at the basic task of retrieving blocked content in a filtering country. In that sense, the tools have individually and as a group been a success.

But all of the tools had some level of problems with performance and security. The performance issues go some way toward explaining the relatively small usage of the tools compared to the number of filtered Internet users. The security issues will be increasingly important as circumvention technology grows in usage, especially as more and more users rely on these tools for a level of anonymity that in most cases the tools are not actually providing. As noted in the preface, this report is now two years old. Some of the specific performance and security problems described belows have since been fixed. But we are confident that these broad conclusions with the performance and security problems of circumvention tools remain.

A. Performance

Of the nine tools studied, only UltraReach provided performance marginally acceptable for day to day browsing in filtered countries, most of them took more than a minute a return most of the sites, and even UltraReach performance was much worse than direct browsing performance. Performance is a vital part of the browsing experience and, as such, is a key component of usability. The performance of all of the tested tools must improve to garner usage by a much larger portion of the filtered Internet users.

Centrally hosted projects (Anonymizer Anonymous Surfing, Anonymizer China, DynaWeb FreeGate, UltraReach, JAP, Coral) can improve the performance of their tools by some combination of technology development and infrastructure improvement. Both development and infrastructure require additional investment, but infrastructure investment provides the most reliable, direct way to improve performance.

Peer hosted projects (Circumventor/CGIProxy, Psiphon, Tor, Hamachi) suffer from the need to rely mostly on consumer class Internet connections for performance and suffer from a deadly combination of the low bandwidth, high latency, and bidirectional capping of such connections. The only way to improve the performance of peer hosted projects is either to recruit more volunteers to host servers (in the case of Tor) or to improve the underlying technology to make better use of the available connections.

B. Security

Strong security through a network not designed with security in mind is hard. Every tool tested had serious security issues. All of the HTTP proxy tools (Anonymizer Anonymous Surfing, Anonymizer China, DynaWeb FreeGate, UltraReach) require that the user trust their data to the servers hosted by the project. Both of the CGI proxy tools (Circumventor/CGIProxy and Psiphon) suffer from holes that generate some direct, unencrypted traffic. DynaWeb FreeGate fails to encrypt much of its traffic. The CGI proxy tools and Anonymizer Anonymous Surfing have user interfaces that are likely to lead to data leakage with regular usage. Tor suffers from the possibility of rogue exit nodes hijacking data, and JAP suffers from a poor network infrastructure that negates the central premise of re-routing. Coral and Hamachi have significant security problems but are non-functional for circumvention in any case.

At the current time, UltraReach, Tor and Anonymizer China provide the best possibility for security. But UltraReach users must know to turn off active content and javascript within their browsers, which fact UltraReach does not disclose. And UltraReach admits to implementing its own filtering system and so will not be acceptable to many users. Anonymizer China is only available to Chinese users and so is not an option for most filtered countries.

Appendix II: Methods

Following is a description of the methods used for all of the tool tests along with the set of questions asked about each tool to produce the tool analysis. This sort of testing is necessarily highly technical and requires an engineer. In particular, the researcher must be able to tell the difference between poor setup or configuration and blocking. Likewise, the researcher must be able to diagnose not just that a particular test failed but why. We have made a strong effort to use straightforward, repeatable methods during the testing, but many of the methods are highly technical. So any engineer should be able to repeat the process with the same or new tools as long as the engineer is a skilled user of both Windows and Linux machines.

1. Lab Setup

A. Lab Setup Summary

Testing Computer:	Lenovo Thinkpad T60, 1.25G RAM
Virtualization Software:	Vmware Workstation
Host OS:	Windows XP Professional
Virtual Computer OS:	Windows XP Professional
Virtual Computer RAM:	512M
Testing Browser:	Internet Explorer 7.0
Virtual Router OS:	Ubuntu 6.10

B. Lab Setup Details

All of the tests were performed on a single mid-range laptop running Windows within multiple vmware images. The laptop was a Lenovo Thinkpad T60 with 1.25G of RAM. Vmware is a virtualization package, which allows a user to run many different virtual computers within a single real computer. Vmware allowed us to run each tool on its own separate, clean computer without having to setup a whole lab full of real machines or having continually to install and uninstall each tool on the same computer.

Vmware offers a number of different packages, including a free version. We used vmware workstation, which costs about $200. The vmware workstation package offers above the free version two features helpful for this testing: multiple snapshots for each virtual computer and virtual computer cloning. A trial version of vmware workstation is available that offers full functionality for thirty days.

A complication of using vmware images is that each image requires its own separately licensed version of Windows. If you install a standardly licensed version of Windows on a vmware image and clone that image, Windows will try and fail to reauthenticate itself on the first boot of the cloned image. The only easy way to circumvent this problem without buying a different license for each testing image is to install a site licensed version of Windows. Harvard University purchases a site licensed version of Windows, so we had easy access to such a version.

One virtual computer was setup with Windows for each tested tool. All tests for a given tool were run on the virtual computer assigned for that tool, even for tools with no client software. The tool virtual computers were setup on a virtual network with a virtual router computer running Ubuntu Linux. The virtual router was connected through Natural Address Translation (NAT) to the Internet. Each tool virtual computer connected through another layer of NAT running on the virtual router. Running all traffic through the virtual router allowed us to snoop on the network traffic as necessary and also to setup the filters and network degradations for the lab tests.

C. Setup Script

Steps to install vmware image lab:

1. Download and install the latest version of vmware workstation from http://www.vmware.com.

2. Create a new image for the virtual router with 512M RAM and 10G of on demand disk space.

3. Setup the image to have two network devices: one connected to outgoing NAT and the other connected to the vmnet2 virtual lan.

4. Install the latest version of Ubuntu Linux available from http://www.ubuntu.com.

5. Use the following script to configure NAT from the virtual lan to the outgoing NAT connection. Make this a script at /usr/local/sbin/start-masquerade.sh and add the script to /etc/rc.local to run automatically at startup.

```
iptables -P INPUT ACCEPT
iptables -F INPUT
iptables -P OUTPUT ACCEPT
iptables -F OUTPUT
iptables -P FORWARD DROP
iptables -F FORWARD
iptables -t nat -F
iptables -A FORWARD -i eth0 -o eth1 -m state –state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -L
```

7. Configure the network interface on the virtual lan to have a static address of 192.168.217.1.

8. Install the bind9 package (apt-get install bind9) and configure the server to run bind9 at startup. Edit /etc/resolv.conf to use localhost for dns and 'chattr -i /etc/resolv.conf' to prevent dhcp from overwriting this value (we found the vmware name server to be unreliable, and you'll need the name server for the filtering test anyway).

9. Configure the dhcp3 server to run at startup and configure the dhcp3 server to return addresses in the 192.168.217.1287 – 192.168.217.254 network by adding the following lines to /etc/dhcp3/dhcpd.conf:

```
option domain-name-servers 192.168.217.1;
subnet 192.168.217.0 netmask 255.255.255.0 {
range 192.168.217.128 192.168.217.254;
option routers 192.168.217.1;
}
```

10. Take a snapshot of the router image.

11. Create a new image for the base Windows install with 512M of RAM and 10G of on demand disk space.

12. Setup the image to have only one network interface, connected to the vmnet2 virtual lan.

13. Install and update Windows XP on the image.

14. Verify that the network connection through the router is working and is going through the router (verify that it works with the router running and does not work with the router paused).

15. Take a snapshot of the base Windows install.

16. Clone the base Windows install for the first tool image.

17. Startup the first tool image and install and sanity test the first tool on it.

18. Take a snapshot of the tool image with the tool installed.

19. Repeat steps 16 – 18 with each tool as well as for a vanilla install with no tool for the control tests.

1. Lab Testing Methods

The lab tests consist of two sets of tests – network condition tests and filter tests. The network condition tests rely on support built in to the Linux routing functionality and on one Windows tool to simulate various sorts of network degradations. The filtering tests rely on a combination of built in routing functionality, DNS configuration, and network snooping to recreate IP block, DNS block, and keyword block filtering.

Each set of tests consists of using each tool image to request each of the Alexa global top ten web sites. For each set of tests with each tool, the router is reconfigured to simulate either a network condition or a filter. For each network condition request, record the response time, response code, and any notes about anomalies in the response. For the filter requests, record only the response code and any notes.

The sequence of testing is:

1. Startup vmware image for a single tool.

2. Configure router for direct (control) connection.

3. Request each of the ten urls, recording time, code, and notes for each.

4. Configure the router to simulate 500ms latency.

5. Delete all browsing history and restart IE.

6. Request each of the ten urls, recording time, code, and notes for each.

7. Reset router to remove latency.

8. Repeat steps 4 – 6 with each of 1000ms latency, 10% packet loss, 25% packet loss, 28.8k bandwidth, and 128k bandwidth.

9. Configure router to filter by IP.

10. Delete all browsing history and restart IE.

11. Requests each of the ten urls, recording code and notes for each.

12. Configure the router to turn off IP filtering.

13. Repeat steps 9 – 12 with DNS and keyword filtering.

14. Repeat steps 1 – 12 with each tool image, including the control image.

A. URLs Tested

Note that for some of the following urls, sub-pages were used in place of the site home page. For example, for google, a specific search was used rather than the google search box home page.

http://yahoo.com

http://msn.com

http://www.google.com/search?q=filtering

http://youtube.com/watch?v=B8x0FY1rJjs

http://myspace.com

http://search.live.com/results.aspx?q=filtering

http://www.baidu.com/s?wd=filtering

http://orkut.com

http://www.qq.com

http://www.yahoo.co.jp

B. Network Conditions

Use the following commands to simulate and then remove the following network conditions. The first two conditions use the Linux network emulator module to simulate latency and packet loss. The final condition uses a free windows tool to simulate low bandwidth.

I. Latency

To simulate network latency, run the following commands on the router:

set 500ms: tc qdisc add dev eth0 root netem delay 500ms

unset: tc qdisc del dev eth0 root

set 1000ms: tc qdisc add dev eth0 root netem delay 1000ms

unset: tc qdisc del dev eth0 root

II. Packet Loss

To simulate packet loss, run the following commands on the router:

set 10%: tc qdisc add dev eth0 root netem loss 10%

unset: tc qdisc del dev eth0 root

set 25%: tc qdisc add dev eth0 root netem loss 25%

unset: tc qdisc del dev eth0 root

III. Bandwidth

To simulate reduced bandwidth, download and install on the host laptop the Traffic Shaper XP tool available for free at http://bandwidthcontroller.com/trafficShaperXp.html. Start the Traffic Shaper tool and create the following rules for the host laptop's outgoing network connection:

Throttle 128k Download

Throttle 128k Upload

Throttle 28.8k Download

Throttle 28.8k Upload

To run each bandwidth test with each tool, enable the two rules for the given bandwidth rate, enable the two tools for the rate (eg. the two 128k rules), run the tests, and then disable the rules.

Note that the Linux iptables routing software includes support for bandwidth limiting, but we were unable to get it to work reliably at such low bandwidths.

C. Filters

I. IP Block Filtering

To simulate IP block filtering, use the Linux iptables support for blocking specific IP addresses. For

convenience, it is best to put the following into a script so that the commands can easily be run repeatedly:

```
iptables -t nat -A PREROUTING -p tcp -d yahoo.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d msn.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d www.google.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d youtube.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d myspace.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d search.live.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d www.baidu.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d orkut.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d www.qq.com -j DNAT -i eth1 –to 192.168.217.1
iptables -t nat -A PREROUTING -p tcp -d www.yahoo.co.jp -j DNAT -i eth1 –to 192.168.217.1
```

To remove the IP filter, run the following:

```
iptables -t nat -F PREROUTING
```

II. DNS Block Filtering

To simulate DNS block filtering, you need to configure the bind server to refuse requests for the blocked domains. To configure the bind server, add the following line to the /etc/bind/named.conf.local:

```
//include /etc/bind/blocks.conf
```

Then create /etc/bind/blocks.conf with the following configuration:

```
zone “yahoo.com” {
type master;
file “/etc/bind/db.225”;
forward only;
forwarders {0.0.0.0 ; };
allow-query {1.1.1.1/32 ; };
}
```

Repeat the above lines for each of the domains to be blocked.

To turn on blocking for a given set of tests, uncomment the line in named.conf.local:

```
include /etc/bind/blocks.conf
```

and restart the bind server by running the following command:

```
/etc/init.d/bind9 restart
```

To turn off DNS filtering, re-comment the line in named.conf.local and restart the bind server again.

III. Keyword Block Filtering

To simulate a keyword block filter, you will not actually block any connections. Instead, you will install a snooper on the network that will look for keywords in the traffic and print a notice when the keyword has been found in the network traffic. The snooper has the advantage of letting you rerun the test while

capturing all of the connection data to trace the exact cause of the data leak.

First, install snort (apt-get install snort) and configure it to look for the keywords HTTP and each domain name in the network traffic. To configure snort, create /etc/snort/circumvention.conf with the following content:

```
alert tcp any any -> any any {msg:"HTTP"; content: " HTTP/1.";}
alert tcp any any -> any any {msg: "YAHOO"; content: "yahoo.com";}
alert tcp any any -> any any {msg: "MSN"; content: "msn.com";}
alert tcp any any -> any any {msg: "GOOGLE"; content: "google.com";}
alert tcp any any -> any any {msg: "YOUTUBE"; content: "youtube.com";}
alert tcp any any -> any any {msg: "MYSPACE"; content: "myspace.com";}
alert tcp any any -> any any {msg: "LIVE"; content: "live.com";}
alert tcp any any -> any any {msg: "BAIDU"; content: "baidu.com";}
alert tcp any any -> any any {msg: "ORKUT"; content: "orkut.com";}
alert tcp any any -> any any {msg: "QQ"; content: "qq.com";}
alert tcp any any -> any any {msg: "YAHOO JAPAN"; content: "yahoo.co";}
```

The above patterns are not comprehensive – some matchable keywords might pass through the filter undetected. But any unencrypted HTTP requests will be caught by the "HTTP/1." pattern, which is required in all HTTP requests. The domain requests should likewise catch most unencrypted traffic, since the domain will be included in image, javascript, and other locations.

To simulate a keyword filter, just run the following:

```
snort -i eth1 -c /etc/snort/circumvention.conf -A console -q
```

The above command will print out an alert if it matches any traffic on the network.

To investigate an alert more fully, rerun the request with the following command running on the router:

```
tcpdump -v -i eth1 -s 0 > dump.dat
```

and then open 'dump.dat' in a text editor to examine the network traffic.

2. In Country Testing Methods

For in country tests, use each tool and the control install to request each of the Alexa top ten sites for the tested country as well as twenty sites blocked by the country. For the full list of urls tested for each country, see the full test results in Appendix II.

The sequence of testing is:

1. Startup vmware image for a single tool.

2. Configure router for direct (control) connection.

3. Request each of the thirty urls, recording time, code, and notes for each.

4. Repeat steps 1 – 3 with each tool image, including the control image.

A. URLs Tested

The urls tested in each of the in country tests are listed in the full results appendix.

3. Tool Analysis Framework

After completing the lab tests, answer the following questions about each tool.

A. Summary

What does the tool do?

What method does it use?

What is the tool's stated purpose?

How does the tool handle hosting (central or peer)?

How well does it do it?

B. Lab Testing Results

What are the average response times for the lab tests? How much slower is the tool than the control test?

What were the results of the filtering tests?

C. In Country Testing Results

What are the average response times for the in country tests? How much slower is the tool than the control image?

Was the tool blocked in any countries?

Did the tool fail to return any specific pages?

Did the tool exhibit any other problems?

D. Availability

What is the cost of the tool?

How easy is the tool to download or otherwise obtain?

E. Install

Does the tool require client installation?

How easy is the client installation?

What platforms does the client run on?

What are the side effects of the installation?

F. User Interface

What is the user interface for the tool?

[include screenshot]

What user interface does the tool provide?

How easy is the user interface to learn?

How easy is the user interface to use?

Does the user interface make clear what the tool is doing?

What is the application's startup / switching time?

G. Server Component

Does the tool require a server component installation by a known peer?

How easy is the server install?

How expensive is the server to run?

How easy is the server to run?

Who hosts the servers?

H. Blocking Resistance

What steps does the tool take to avoid blockage?

Does the tool have flexibility to adjust to future blocking strategies?

I. Internationalization

What languages is the tool available in?

J. Internal Filtering

Does the tool itself engage in any filtering?

K. Security

What tradeoffs does the tool make between security and usability?

Does the tool filter out javascript?

Does the tool filter out active content?

Will actual usage decrease security of tool as designed?

How does the tool's security change as users increase or decrease?

L. Usage

What metrics are available about the usage of the tool?

M. Support

What support documentation is available?

What interactive support is available?

If support is community supported, how active is the community?

N. Development Activity

How old is project?

How active has development been recently?

Is active development planned for the next year?

O. Community

Does the project have an active user community?

Is the project connected to its use community?

P. Code Availability

Is the code freely available?

If not, is the code available for audit on request?

Q. Development Process

Is the design of the tool well documented?

What is the tool's testing and deployment process?

R. Evaluation

Given the answers to the above questions, evaluate the tool on the following criteria:

- Utility (Does it retrieve filtered web pages? How fast is it compared to unmediated browsing? How well does/will performance scale?)
- Usability (How easy it for novice users to learn?)
- Security (How well does the tool protect the user from surveillance at various points in the network?)
- Promotion and marketing of the tool (How well is knowledge of the tool pushed out to users in filtered countries?)
- Fiscal sustainability of the project (To what degree is the project on a path toward fiscal self sustainability?)
- Openness (Is the code open source? Are the developers approachable and forthright about their techniques and future plans?)