Leveraging Internet Services to Evade Censorship

Apostolis Zarras

Technical University of Munich
zarras@sec.in.tum.de

Abstract. Free and uncensored access to the Internet is an important right nowadays. Repressive regimes, however, prevent their citizens from freely using the Internet and utilize censorship to suppress unwanted content. To overcome such filtering, researchers introduced circumvention systems that avoid censorship by cloaking and redirecting the censored traffic through a legitimate channel. Sadly, this solution can raise alerts to censors, especially when it is mistakenly used. In this paper, we argue that relying on a *single* channel is not sufficient to evade censorship since the usage pattern of a circumvention system differs compared to a legitimate use of a service. To address this limitation of state-of-the-art systems, we introduce CAMOUFLAGE, an approach to combine multiple *non-blocked* communication protocols and dynamically switch among these tunnels. Each protocol is only used for a limited amount of time and the Internet connection is transparently routed through instances of different censorship circumvention systems. We prototype CAMOUFLAGE by using applications which are based on these protocols and also offer end-to-end encryption to prevent censors from distinguishing circumvention systems from regular services. We evaluate CAMOUFLAGE in countries that impose censorship and demonstrate that our approach can successfully bypass existing censorship systems while remaining undetected.

1 Introduction

The Internet has become the medium of choice for people to search for information, conduct business, and enjoy entertainment. It offers an abundance of information accessible to anyone with an Internet connection. Additionally, the explosion of social media plays a central role in shaping political debates, as for example the coordination of movements in "Arab Spring", where citizens used Facebook, Twitter, and YouTube to put pressure on their governments. However, free communication threatens repressive regimes as it, in many cases, exposes concealed truths and government corruption. Personal communication is then becoming subject to pervasive monitoring and surveillance, and various state and corporate actors are trying to block access to controversial information. In detail, regimes can trace, monitor, filter, and block data flows using sophisticated technologies such as IP address blocking, DNS hijacking, and deep packet inspection [12, 27].

With the use of censorship technologies to be more timely than ever, researchers developed a number of different systems to retain the freedom of the Internet [5,16,19], which are widely-known as *censorship circumvention systems* and most of the time try to deploy a redirection proxy that provides access to blocked websites. Nevertheless, censors can locate such proxies and instantly block them [20,33]. The root cause of most of the systems' identification is the differentiation they exhibit from regular Internet traffic. To overcome this limitation, *unobservable circumvention systems* were introduced. These systems try to impersonate popular applications to blend with the *allowed* Internet traffic [35,44,45]. Although these systems sound promising, they fail to raise the bar against censorship mostly because they only implement the imitating protocol partially and thus fall into discrepancies that censors can locate [25].

Consequently, unobservability by imitation is a fundamentally flawed approach [25]. For this reason, researchers introduced new systems that operate higher in the protocol stack. These systems avoid censorship by executing the actual protocol instead of trying to impersonate it [1,2,26,50] and thus can protect the users from various Internet restrictions. However, they suffer from being shut down if they got recognized, mostly due to their users' inexperience. In general, the average end-user is not familiar with the proper configuration and utilization of a circumvention system. Hence, the erroneous usage of such a system can create traffic that may appear suspicious to censors. Additionally, such systems support only one protocol and therefore are susceptible to loosing their functionality in case a government decides to completely block this protocol.

In this paper, we present Camouflage, a novel approach that protects users from Internet censorship, while requiring limited expertise from the users' side compared to existing systems. More specifically, Camouflage is a framework to which existing or future censorship-resistant systems can be plugged in, cooperate, and provide increased resistance against censorship. The main idea of our approach is to tunnel Internet traffic inside multiple non-blocked communication protocols and dynamically switch among them. Many of the existing systems can protect users from being subjects of censorship by tunneling the traffic through various protocols [1,2,26,50], yet they all work independently from each other. Thus, users who want to avoid censors' surveillance should install and configure as many of these systems on their computers, and manually rotate the forwarded traffic among them, which is likely an error-prone process. In our approach, the censorship-resistant tools can be attached to Camouflage as plugins and the framework itself decides when and for how long they will be used.

To demonstrate the functionality of our framework, we built a prototype implementation, which supports four different protocols used by four widely-used applications. To evaluate Camouflage in real-world, we chose countries that impose censorship and browsed the web with censored terms from inside these countries. The experimental results exhibit that our prototype can be successfully used for web browsing, while resisting censors' blocking efforts. To the best of our knowledge, our approach is the first attempt to combine a variety of different censorship circumvention systems under one solid framework.

In summary, we make the following main contributions:

– We propose CAMOUFLAGE, a novel approach for censorship circumvention that combines the advantages of existing systems while makes it easier for users to employ them. Its *plug-and-play* architecture can be used by existing or future censorship-resilient systems.
– We build a prototype based on widely-used applications and evaluate its performance and security. We show that the produced overhead is related to the implementation of each circumvention system.
– We evaluate our prototype in existing censored networks and show its ability to bypass censorship in real-world, concealing at the same time its presence from the censors.

2 Threat Model

Throughout this paper we use the following threat model. We assume that a user connects to the Internet through an Internet Service Provider (ISP) that utilizes some kind of censorship system. In fact, governments can control and regulate ISPs, which can be forced to monitor and block users' access to certain Internet destinations. For example, China can filter IP packets [9], or even can censor services such as blog platforms [52], chat programs [31], and search engines [51]. In our scenario, we assume that the user operates in an inhospitable network where ISPs can trace, monitor, filter, and block data flows using sophisticated technologies [12]. Additionally, we assume that the users are restricted from using proxies or other circumvention systems to evade censorship. Therefore, we take for granted that ISPs can identify and block the traffic that is forwarded through a censorship circumvention system using a variety of different features and strategies [22, 25] and notify the authorities for any incident. Then, the violators might face severe punishments varying from payment of fines to even imprisonment [46].

We also assume that regimes do not want to jeopardize the usability of the Internet due to political and economical implications. For instance, services such as email and VoIP constitute important parts of today's communications among businesses, which benefit from these services to reduce their operational costs. Furthermore, VoIP and chat communications are also widespread among individuals due to the level of convenience and flexibility they offer. Finally, another type of Internet services that is popular, mostly among young people, is online games, which are used for entertainment purposes. Thus, we speculate that censorship regulations do not interfere with fundamental Internet services such as email communications, VoIP, file sharing, and even entertainment services including online games. In the extreme case in which a censor might decide to block some of these services, we believe that this decision will not affect all of them, but only a fraction of the services.

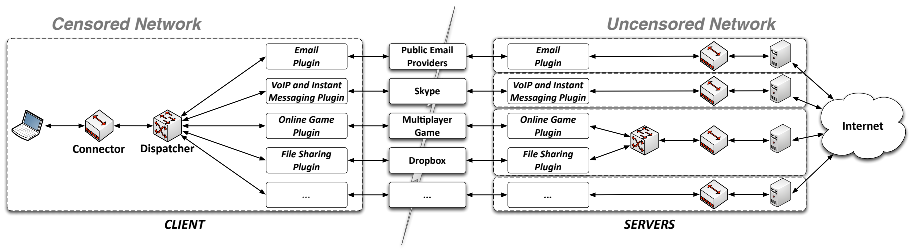

Figure 1: Abstract architecture of Camouflage.

3 System Design

Camouflage acts as a pluggable framework for different circumvention systems that evade censorship by leveraging existing protocols and services. Thus, in order to gain access to the framework, users need to install the Camouflage *client* on their computers along with the appropriate plugins (i.e., circumvention systems). On the other hand, administrators should install the Camouflage *server* as well as the supported plugins. Note that a server can support multiple circumvention systems that operate in parallel and are synchronized. Both clients and servers consist of key components necessary for converting and forwarding the traffic. In the remainder of this section, we introduce the main components of Camouflage and describe how do they contribute to the design and operation of a secure and easy-to-use framework.

3.1 Abstract Architecture

Camouflage is composed of two key components: a client and a server. The client usually runs in a censored environment in which all the communications are monitored by censors, while the server operates in an uncensored and secure environment where the network traffic flows unrestricted. In addition, our framework transfers all the data transparently and thus any software which can be configured to use a proxy, such as a web browser, is able to use the client. On the server side, we set up a proxy (HTTP or SOCKS), which is responsible to communicate with the outside world and access the censored content. Figure 1 illustrates the framework's abstract architecture.

The client consists of the following components: a connector, a dispatcher, and different plugins. The connector receives the traffic from the user's application (e.g., the web browser), transforms it from multiple connections' traffic to a serialized data stream by adding suitable headers, and then encrypts it. Next, it forwards the traffic to the dispatcher, which decides through which plugin the data stream will be transferred. Finally, the plugin running in the client's side is responsible for transmitting the data stream. Accordingly, when a plugin receives an encrypted data stream transfers it to the connector through the dispatcher. In this component, the data stream is decrypted, is split into multiple connections, and the traffic is forwarded to the appropriate application.

The server, on the other hand, consists of the proxy, the connector and the supported plugin. When encrypted data is received, the plugin transfers it to the connector. The server's connector operates in a similar way to the one presented for the client. More specifically, when a data stream is received, it checks the connection identifier and forwards the data through the corresponding connection with the proxy, or opens a new connection if the identifier is currently unused (i.e., the connection was newly established on the client's side). Similar, when the server transmits data back to the client, the proxy sends the data sequences to the connector in order to transform them into an encrypted data stream and then the plugin transfers it through the suitable channel. As we previously mentioned, a server can support more than one circumvention system. In this case, the existence of a dispatcher is necessary in order to combine the different plugins and forward the traffic to the same proxy.

3.2 Connector

Modern applications, such as web browsers, use HTTP/1.1 protocol that allows multiple network connections to run in parallel. In detail, when a user visits a web page that has many different objects on it (e.g., images, JavaScript files, frames, data feeds, etc.), the browser tries to download several of them at once in different parallel sessions to obtain better performance. Most HTTP servers and browsers use an HTTP protocol feature called *keep-alive* that does not close the TCP connection when the client is done with it; the connection closes either after an idle connection timeout or after a maximum number of allowed requests. This makes sense since opening a remote connection is expensive due to the three-way TCP handshake, so it is faster to open one connection and then download n items compared to open and close a connection n times. However, the plugins provide only one data channel for forwarding the data. To overcome this obstacle, we need a middleware that combines the multiple connections generated from applications with the single data channel provided by the plugins. In Camouflage, the connector plays the role of this middleware.

In detail, the connector bundles the traffic from all connections to one data stream by adding a small header in front of the payload. This header contains the identifier of the connection and the payload's length. Using the identifiers, the connector can forward the data to the correct connection within the user's application, while the length of the payload ensures that the appropriate amount of data will be read. The transformed data stream is then encrypted and forwarded to plugins. The connector also receives a similar data stream back from the plugins with the contents of the censored web pages. Additionally, it ensures that the state of each connection is consistent at both client and server side. Especially if one side closes a connection, this information is transmitted to the other side which in turn closes the connection as well.

3.3 Dispatcher

The dispatcher acts as an intermediate component between the connector and the actual plugins. More precisely, the use of the dispatcher is twofold: (*i*) forward the traffic from a client to a server and vice-versa, and (*ii*) schedule the orchestration of activities by planning and monitoring the plugins. For the first part, the dispatcher forwards the outgoing traffic from the connector to a server over the currently active plugins. Accordingly, it receives the incoming traffic from a server and delivers it to the connector. To this end, the dispatcher should know which plugins are supported by the current instance of CAMOUFLAGE. If a connection to a plugin is not feasible anymore, the dispatcher will temporarily disable this plugin and will try to connect with an alternative one. Nevertheless, the plugin will be enabled again after a timeout period.

For the second part, the dispatcher administers the plugins by implementing the schedule in which a plugin is selected. The implementation of the schedule could be created either manually by the user or automatically by integrating the manifest files of plugins. The former solution is recommended for advanced users, while inexperienced ones can always use the latter. The dispatcher can decide at any time if it wants to execute the plugins on parallel, or one at a time. This way, makes it difficult for the censors to create detection patterns.

3.4 Plugins

The data channels (circumvention systems) that used for tunneling the traffic are the lifeblood of CAMOUFLAGE. Each circumvention system is implemented in our framework as a standalone plugin. CAMOUFLAGE supports an abstract architecture, in which new circumvention systems can be easily plugged in. It is mandatory that the implementation of each system to leverage an actual protocol, or service, and not try to imitate it. Additionally, each plugin must accompanied by a manifest file. This file contains information such as the plugin's unique identifier, the recommended operation time used by the dispatcher, the suggested timeout period after which the dispatcher will try to reconnect to the plugin if a previous connection attempt failed, and the list of the required software. Both the usage time duration and the timeout period are only recommended values by developers and can be voluntarily modified by users.

Each plugin is responsible for concealing its tunneled data stream. Therefore, it is highly recommended the utilization of a two-layer encryption strategy. First, each plugin should leverage applications that use encrypted communications by default. This will prevent regimes that monitor the network traffic to have a direct access to unencrypted data. However, this may not be always the case. For instance, a company that wants to operate in a country that imposes censorship must accept the country's requirements in order to be allowed to enter the country's market. These requirements could permit, among the others, the government's censors to have access to unencrypted network traffic. Thus, no matter how strong is the encryption that the application uses, the plugin should implement a second layer of encryption as well. More specifically, it must encrypt the data stream before being encrypted by the application itself. This way, the censors that monitor the network traffic will be prevented from accessing the plaintext data if an agreement with the application provider is established. Overall, the two layers of encryption ensure that even if the application gets *compromised*, the content of the data stream will remain hidden.

4 Circumvention Systems

As we mentioned, the lifeblood of our system is the different plugins it supports and each of them implements a circumvention system. To prove the feasibility of our approach, we implemented a prototype of Camouflage using four different circumvention systems that are attached to our framework as plugins. These plugins leverage four different services and protocols to evade censorship and do not rely on emulation or any other imitation technique, but each of them utilizes the actual implementation of a protocol. To test each service, we selected an application that supports a utilized protocol and implemented the censorship-resistant system on top of this application. In brief, we chose the following applications:

1. *Public email providers:* to implement a circumvention system based on SMTP protocol.
2. *Skype:* to leverage VoIP and instant messaging communication protocols.
3. *Runes of Magic:* to benefit from communication protocols in multiplayer online game platforms.
4. *Dropbox:* to conceal traffic within a file sharing service.

Note that these applications are selected only as proof-of-concept scenarios and therefore the circumvention systems can operate in the same way with differently selected applications (e.g., utilization of *Google Voice* as VoIP application). Same principle applies in the case that an application is forbidden in a country. It is worth to mention here that some of these applications were proposed in previous works [26,50], however, it was not possible to find their actual implementations and in order to evaluate our prototype we had to design and implement them from scratch. In addition, for the purpose of this paper, the communication between the client and the server is already initialized, which means that there is no need for any kind of registration. Nevertheless, sophisticated registration strategies could be applied in real-world circumvention systems to prevent attacks, such as denial-of-service against the system, but these strategies are outside the scope of this paper. In general, the framework itself allows developers to design their own registration strategies without any restrictions.

4.1 Email

Electronic mail (email) is a widely-used method of exchanging digital messages from an author to one or more recipients. Modern email systems are based on a *store-and-forward* architecture, which permits servers to accept, forward, deliver, and store messages. The wide acceptance of email allows us to create a circumvention system that utilizes the email delivery system to evade censorship. More precisely, the system uses publicly available email providers to hide the data stream inside email messages. Private email providers could be used as well to perform this task, but we believe they are more prone to manipulation by repressive regimes compared to public email services provided by international corporations. We assume that censors do not have access to the users' mailboxes hosted outside of the regime's geographical boundaries. In contrast, it is easier for a censor to access mailboxes of email providers that are hosted inside the regime's borders. Moreover, we consider as suitable email providers those which by default offer email encryption. This way, an encrypted email will not be considered suspicious by censorship authorities, if all the emails sent by this email provider are encrypted as well.

With regards to the design of the email circumvention system, both client and server plugins share the same mailbox that is hosted at a public service. The email servers of this service must reside outside the censors' jurisdiction (e.g., Gmail, Yahoo Mail, Outlook.com). The client communicates with the server, and vice-versa, by sending emails to the same mailbox. The data stream is divided into blocks, which are encoded in *Base64* format. Additionally, we use encryption to protect users from revealing the content of the visited web pages to a compromised email provider. The body of the email contains the data stream, while the header contains a message counter to retain the correct sequence of data blocks and a flag that indicates the sender of this message. Both client and server implementations use the *IMAP IDLE* feature to get notified on new emails and recreate the data stream from the received messages. This feature allows IMAP email users to immediately receive any mailbox changes without having to automatically and repeatedly ask the server for new messages [28].

4.2 VoIP and Instant Messaging

Voice over Internet Protocol (VoIP) is a technology used for the delivery of voice communications and multimedia sessions over Internet Protocol (IP) networks. Instant messaging, on the other hand, is a type of online chat that offers real-time text transmission over the Internet. Short messages are typically transmitted bi-directionally between two parties, where each party composes a sentence and sends it to the other. Modern applications that support VoIP often support instant messaging as well. One such application is Skype, in which users communicate with each other both with voice and text. For the prototype of this circumvention system we use Skype and implement it to transmit data either as audio or as encrypted text messages. However, any other VoIP software will work in a similar way. In the following paragraphs we analyze the operation of each service in more detail.

Nowadays, many companies switch from traditional telephony to VoIP to reduce their telephone costs [10, 17]. Similarly, individuals communicate with each other using publicly available and usually free VoIP services. Hence, VoIP is difficult to be manipulated or even blocked by censorship authorities. We exploit this censorship weakness and create a system that transfers uncensored data over VoIP. More specifically, our system modulates binary data into audio signals, which are transmitted over the voice channel of a VoIP software. This concept is similar to the operation of modems (i.e., a device that modulates an analog carrier signal to encode digital information and also demodulates such a carrier signal to decode the transmitted information). The goal is to produce a signal, which can be transmitted and decoded, to reproduce the original digital data. For our prototype we used a 1200 baud Audio Frequency Shift Keying modem without sophisticated error correction. However, in a real-world application a more advanced approach should be applied.

We mentioned that for the prototype implementation of the system we selected Skype. Therefore, the data stream is converted to audio signals and transmitted from one edge to the other using *Skype call*. For sending the generated audio signal over Skype we use a virtual audio device. This device is installed as a driver and behaves like a real sound card. In Skype it can be configured as the microphone device. Additionally, a second virtual audio device is set as Skype's output and all signals pointing to this device are recorded by the system, and eventually demodulated back to the initial data stream.

Alongside with VoIP services, some companies leverage instant messaging as a complementary method to the customer support services they offer. In addition, instant messaging is one of the most popular Internet activities among individuals. According to studies, a vast majority of the Internet's population uses instant messaging as its main communication tool [29]. Consequently, if a repressive regime blocks a major application such as Skype, which provides instant messaging communication services to its users, this will cause a severe impact to a significant portion of the regime's inhabitants and companies. Hence, in some cases is preferable for a repressive regime to monitor this means of communication than completely forbid it. With this in mind, we created a circumvention system that utilizes the text channels of instant messaging services. To our benefit, Skype automatically encrypts the transmitted messages, fact that offers us a two-layer encryption similar to email circumvention system. In general, we used a similar approach to the one presented for the email circumvention system.

4.3 Online Gaming

Online games are video games which are played over the Internet. These games are divided in single-player where input from only one player is expected throughout the course of the gaming session and multi-player that allow more than one person to play in the same game environment at the same time. The latter provide their users (players) with a form of social communication channel in which the players can interact with each other. These communication services in multi-player games can be used as covert channels for the secret transfer of data streams. Modern multi-player games offer various communication channels such as public chat rooms, private instant messaging, and voice chat. We leverage the services offered by multi-player games to create a circumvention system.

We prototype a circumvention system, by using the voice chat communication channel of *Runes of Magic*, a massively multiplayer online role-playing game. We create two characters (avatars) which are connected to the same server. The client owns the first avatar, while the server possesses the second. The avatars create a voice chat room where they communicate. We utilize a similar approach to the one presented for the VoIP circumvention system. More accurately, we convert the digital data stream to audio signals and transmit them from one side to the other through the voice chat room. The other side demodulates the signals back to their original form and processes the requests.

This approach is only a proof-of-concept. We want to show that a strategy that leverages online games to evade censorship is feasible. Thus, we only use the voice chat rooms to transmit the data, which imposes limitations in the amount of data being transferred. To increase the amount of transmitted data, we can use in parallel approaches that benefit, for example, from the movements of avatars to transmit information over the online game engine [47, 48].

4.4 File Sharing

File sharing is a private or public distribution of data or resources, such as documents, multimedia, graphics, computer programs, images, or e-books, in a network with different levels of sharing privileges. More specifically, it allows to a number of people to simultaneously modify the same files. File sharing is known for quite a long period. It is massively used, from companies and universities, among people who work on the same projects and need to share and modify the same documents and data. Recently, as social media has become increasingly popular with hundreds of millions of users, file sharing became attractive among individuals as well. Nowadays, people use it to upload and share pictures and videos among friends. File sharing is so massively used in today's world, which offers a unique opportunity for a circumvention system based on this service.

We create a circumvention system on top of Dropbox. Dropbox is a personal cloud storage service frequently used for file sharing. Our censorship-resistant system uses the official Dropbox client to create a private communication channel between the client and the server. After configuring and starting Dropbox on both sides, the tool uses a shared folder for data exchange. The sent data is split into blocks which are then stored as encrypted files in the shared folder. The folder is automatically synchronized with the other party. Both plugins can monitor the folder for new files, read the data from them, and recreate the data stream. For additional security we can apply solutions that revoke the access to files after a certain period of time [6, 21, 37, 49].

5 Evaluation

We implemented a prototype of CAMOUFLAGE and evaluated the effectiveness of our approach. We first measured its performance and then studied the users' behavior to properly configure the plugins. Finally, we evaluated CAMOUFLAGE's ability to evade censorship on countries that impose censorship to their citizens.

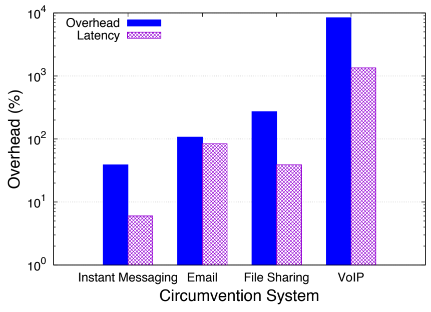

Figure 2: Overhead and latency in loading a page with different plugins.

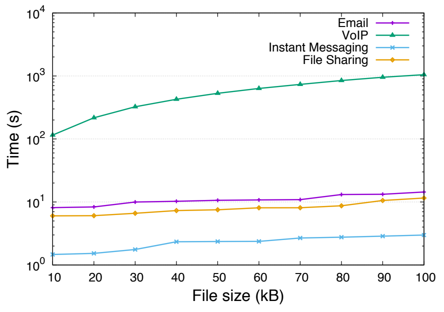

Figure 3: Downloading time for files size from 10 to 100 kB.

5.1 Performance

Using a circumvention system is the only way to access a censored web page, especially if services such as Virtual Private Networks (VPNs) are forbidden. Thus, to show that a circumvention system must only be used for accessing censored web pages, and not for web surfing, we measured its performance. Therefore, all the performance experiments were conducted by using broadband connections for accessing the Internet on both client and server side. We believe that individuals can contribute to the Internet's freedom by hosting at least one circumvention system (Tor [16] uses a similar infrastructure). Hence, we chose to use DSL instead of a university's connection to make our experiments more realistic. For instance, individuals who want to contribute in the fight against censorship can run a Camouflage server during the night when their bandwidth is usually idle. During our experiments, the utilized DSL connections remained idle and the only traffic sent over the network was the traffic which was created by the circumvention systems. Obviously, using broadband connections had a huge impact in the performance of Camouflage, compared to previous works [26], but at the same time it provided us with realistic results instead of the ideal results we would have gotten in a sterile laboratory environment.

In our first experiment, we measured the overhead and the latency added by each circumvention system when a web page is downloaded. First, we visited the main page of popular websites with a browser without using any intermediate channel and captured all the incoming traffic which constitutes the bottom line of the experiment. Then, we visited the same web pages with different circumvention systems. Between the measurements we cleared the browser's cache so all the contents of the web pages were loaded directly from the network. Figure 2 shows the average values for the overhead and the latency of each system. As we can see, instant messaging increased the incoming data only by 39% compared to VoIP, which increased the traffic by a factor of 84. This is caused because VoIP can conceal a smaller amount of information in its audio signals than the transferred information through raw text. Regarding email, it increased the overhead by 107%, while file sharing increased it by 272%.

The latency on the other hand is caused by different factors for the various systems. For instant messaging, the latency is low because this service is designed to be real-time and transferring data this way does not create much overhead. In the case of the email, the latency is mainly caused by the time email servers spend for processing the mails; the traffic overhead is considered negligible here. Same principle applies for the file sharing. The data is first sent to a file sharing server, processed there, and then synchronized with the file sharing client. Finally, the latency of the VoIP system is clearly caused by its limited bandwidth.

Expanding the previous experiment, we conducted downloads of files of various sizes and measured the time required for a complete download. The experiment used the same configurations described earlier. Figure 3 illustrates the results. The outcomes of this experiment help us to create a version of our framework in which all the supported circumvention systems can operate in parallel. In detail, we notice that a circumvention system that uses instant messaging is the most effective approach no matter what the size of the downloaded file is. Therefore, this approach could be used for web elements that require more bandwidth such as videos and high-resolution images. Email and file sharing systems behave in similar way and the offered bandwidth by each system is rather close to the other. Thus, these systems could be used to download medium and low quality images and medium size Flash applications. Finally, VoIP systems offer a rather small bandwidth for transferring data. Hence, VoIP censorship-resistant tools can be used to transmit a limited amount of data, such as a text entry in a microblogging service like Twitter, or to download the mobile versions of the websites. Although, in theory all circumvention systems can be used for all the tasks, in practice is not always feasible.

5.2 Traffic Patterns

Camouflage offers unobservable connections. By unobservability we mean the ability of a circumvention system to hide its existence from censorship authorities. In other words, the censors should not be able to identify whether a user is using a circumvention system. This is essential for the user's safety because authorities can prohibit the use of technologies that evade censorship. To prevent a censor from detecting Camouflage, the behavior of the protocols used as tunnels should be as close as possible to a user's normal use of those protocols. For this, we monitored the users' behavior when they utilize the proposed applications. Additionally, we considered reports that studied traffic characteristics [8, 40, 41]. The outcomes allowed us to create generic traffic patterns that match the behavior of the majority of users. Table 1 depicts an overview of our exported results. Keep in mind that these results are affected by many factors, such as the age and cultural influences, and may look different in separate regions.

Application	**Duration (min)**
Email	1 – 3
VoIP	20 – 30
Instant Messaging	15 – 20
File Sharing	5 – 10

Table 1: Suggested time values for different applications.

Unfortunately, there exist individuals whose behaviors do not match with our proposed patterns. This allows to a well-trained censor to detect these discrepancies. With that in mind, we designed our prototype to liberally allow its users to configure it based on their demands. More precisely, during the installation process of a circumvention system, users get informed about the typically-used traffic pattern and asked for any modifications. Note that users are not bind to their initial decisions and can modify these traffic patterns any time they want. However, we recommend only the advanced users to manually modify these values because a misconfigured system can cause exactly the opposite results and the existence of Camouflage to be revealed to censors.

Nevertheless, our experiments showed that is not always feasible for users to know their unique traffic patterns as they may do not have neither the experience nor the suitable tools to measure it. Simultaneously, users that use the very same services leveraged by our plugins in their everyday lives can raise suspicions when these services used with and without Camouflage. This results in two different traffic patterns which can be observed by anomaly-based detection systems. To overcome this limitation we enhanced Camouflage with the ability to monitor and decipher the network traffic patterns of a user, when the plugins are not used. This way, the generated network patterns will be unique and based on user's typical behavior. Consequently, a censor will not be able to distinguish between the real traffic and the artificial one, and thus will not be able to detect the existence of our framework, as the results of our experiments revealed. More precisely, the traffic patterns with and without the use of Camouflage looked almost identical which makes it impossible for censors to spot the differences.

5.3 Real-World Deployment

To explore how Camouflage performs in a real-world scenario, we evaluated it in countries that impose censorship to their citizens. Therefore, we acquired access to servers hosted in these countries and imposed to the same censorship as these countries' inhabitants. Then, we tried to access forbidden websites, which are usual websites that contain known forbidden keywords. We repeated our experiments in different time periods to capture any possible changes in the detection capabilities of the censors. Our early results demonstrate that our framework can successfully evade censorship, while it remains undetectable for a large period. In detail, without Camouflage it was not possible to render a plethora of web pages that contained one or more forbidden keywords, which became possible once we utilized our framework. To be certain that our results are accurate, we repeated these experiments for a period of one month. During this period we were able to evade censorship that was imposed in different countries.

A perfect example of such a country is China. It is well known that China has the world's most complex Internet censorship system [23]. However, its censors are very prudent to perform DNS hijacking nowadays due to the risk of affecting the network in other countries [34]. Chinese censors impose strict restrictions on international Internet traffic and the most effective filtering mechanism is the keyword filtering. These factors make China an ideal candidate to evaluate

Camouflage. For this purpose, we used a list of known forbidden keywords. We ran our experiments in daily basis for a period of one month and monitored if during this period (*i*) we were able to access censored web pages and (*ii*) Camouflage got detected and its services were banned. The outcomes of this experiment revealed that with the utilization of Camouflage we could access web pages that otherwise would not be possible. Additionally, during the period of our experiment we were able to continuously access these forbidden web pages which shows that our framework was not detected by censors. Therefore, we believe that Camouflage can assure unobservability as it blends with the real network traffic. It is worth to mention that not all the applications were available in China, for instance, we could not use Dropbox. Therefore, we performed our experiments only with the allowed plugins. Nevertheless, even if an application is not available to a country, it can easily be replaced by another. Therefore, by using more plugins we increase our chances to access the data we want in a heavy censored environment.

Overall, Camouflage was able to evade censorship when applied. However, the main goal of the framework is to provide access to censored web pages. Thus, it should explicitly be used only for this specific scenario and not for everyday tasks such as web surfing, e-radio listening, or any other activities that require a high bandwidth connection and could raise suspicions due to highly-produced network traffic over services that are not designed to produce so.

6 Related Work

Circumvention systems try to ensure anonymity to their users. Anonymity is an old idea. Chaum proposed a technique based on public key cryptography that allows an electronic mail system to hide both the participants and the content of the communication [7]. There exist systems that provide anonymity by following a high-latency network design [13,24]. These systems can resist against strong adversaries, but introduce too much lag for interactive tasks such as web browsing. On the opposite side, systems with low-latency network design [3,4,11] can anonymize interactive network traffic, but it is difficult to prevent an attacker that eavesdrops both ends of the communication from correlating the timing and volume of traffic [38]. Finally, there is the peer-to-peer network design in which the participants both generate and relay traffic for others [30,39].

On the other hand, the oldest technique to evade censorship and surveillance is the use of open proxy servers [18,42]. Toward this direction, Infranet [19] improves the proxies infrastructure by leveraging a tunnel protocol that provides a covert communication channel between the clients and the servers. Similarly, Collage [5] uses user-generated content on social-networking and image-sharing websites such as Facebook and Flickr to embed hidden messages into cover traffic. To this end, researchers presented an obfuscation-based approach that enables users to follow privacy-sensitive channels, while makes it difficult for the censors to discover the users' actual interests [36]. However, these designs are susceptible to inside attacks where censors pretend to be ordinary users to locate and block the infrastructure of censorship resilient systems [20]. Over the years, researchers have designed better proxy distribution strategies that protect proxies from Sybil attacks [20, 33]. Additionally, reputation systems might be used to detect censors who have infiltrated in the proxies' network [43]. These strategies are adequate against individual users, but are insufficient in thwarting censors that rule a significant amount of untrustworthy users.

One of the most effective circumvention tools is Tor [16], which is a circuit-based anonymous communication system that uses encryption to conceal the network packets. More specifically, it interposes at least three relays between each user and the website the user visits. The transferred packets through these relays are encrypted, and each relay can decrypt only the necessary information that leads the packet to the next relay. Although the relationship between the user and the visited website through Tor is secure, repressive governments can block the Tor itself [14]. To make Tor resilient to such attacks, developers have proposed a centralized discovery service to disseminate a restricted set of relay identities to requesting users [15]. Obfsproxy [32] is the first Tor pluggable transport, which adds an additional layer of encryption to Tor's traffic to obfuscate its identifiers. However, albeit the developers' modifications, the problem is that the traffic generated by Tor remains recognizable by its characteristic patterns and content signatures.

Unobservable circumvention systems, on the other hand, instead of encrypting the web content and access it through proxies, imitate applications and blend with the authorized Internet traffic. SkypeMorph [35] is a system designed to encapsulate the Tor's traffic into a connection that resembles Skype video traffic. CensorSpoofer [44] is a framework for censorship-resistant web browsing that exploits the asymmetric nature of web browsing traffic. StegoTorus [45] is a tool that comprehensively disguises Tor from protocol analysis. Although these approaches sound promising, Houmansadr et al. [25] demonstrate that these systems fail to achieve unobservability because they implement only partially the imitating protocol and fall into discrepancies.

An alternative to unobservable circumvention systems by imitation is to run the actual protocols and tunnel the hidden content inside their traffic. FOE [2] and MailMyWeb [1] are two systems that can download a requested website and send it as an email attachment to the requesting user. These systems can evade censorship, however, the users cannot interact with the actual website and they can only leverage these systems for accessing static websites. SWEET [50] encapsulates a censored user's traffic inside email messages. In detail, the client tunnels its network traffic inside a series of email messages that are changed between the client and an email server operated by SWEET's server. The server acts as an Internet proxy by forwarding the encapsulated traffic to the requested blocked destinations. FreeWave [26] operates by tunneling Internet traffic inside non-blocked VoIP communications by modulating them into acoustic signals that are carried over VoIP connections. These systems appear to work flawless. Nevertheless, their main limitation is that they only support one protocol and thus their overuse by users can trigger alerts on censors.

7 Conclusions

In this paper, we presented CAMOUFLAGE, a novel approach that protects users from Internet censorship. The key idea of CAMOUFLAGE is a framework where different circumvention systems can be plugged in and help users to access an uncensored Internet. The framework operates one layer below the circumvention system and thus the design and the implementation of each system lies in the hands of each developer. To demonstrate the feasibility of our approach, we built a proof-of-concept prototype on widely-used applications and evaluate its performance and security. We showed that different systems can co-exist with each other, while a central framework can synchronize them. Finally, we evaluated CAMOUFLAGE in countries that impose censorship. The outcomes of our experiments revealed that by using CAMOUFLAGE is possible to access an uncensored Internet while at the same time the existence of our framework remained hidden from the deployed censors.

Acknowledgments

The research was supported by the German Federal Ministry of Education and Research under grant 16KIS0328 (IUNO).

References

1. MailMyWeb. `http://www.mailmyweb.com`, Jun. 2013.
2. The Foe Project. `https://code.google.com/p/foe-project`, Nov. 2013.
3. O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. In *Designing Privacy Enhancing Technologies*, 2001.
4. J. Boyan. The Anonymizer – Protecting User Privacy on the Web. *Computer-Mediated Communication Magazine*, 4(9), 1997.
5. S. Burnett, N. Feamster, and S. Vempala. Chipping Away at Censorship Firewalls With User-Generated Content. In *USENIX Security Symposium*, 2010.
6. C. Castelluccia, E. De Cristofaro, A. Francillon, and M.-A. Kaafar. EphPub: Toward Robust Ephemeral Publishing. 2011.
7. D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
8. M. Chui, J. Manyika, J. Bughin, R. Dobbs, C. Roxburgh, H. Sarrazin, and M. Westergren. *The Social Economy: Unlocking Value and Productivity Through Social Technologies*. McKinsey, 2012.
9. J. R. Crandall, D. Zinn, M. Byrd, E. T. Barr, and R. East. ConceptDoppler: A Weather Tracker for Internet Censorship. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.
10. J. Crispin. The Importance of VoIP and a Business Continuity Plan for Business Survival. `http://www.tech2date.com/the-importance-of-voip-and-a-business-continuity-plan-for-business-survival.html`, Jun. 2011.
11. W. Dai. Pipenet 1.1. *Usenet post*, 1996.

12. A. Dainotti, C. Squarcella, E. Aben, K. C. Claffy, M. Chiesa, M. Russo, and A. Pescapé. Analysis of Country-Wide Internet Outages Caused by Censorship. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2011.
13. G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE Symposium on Security and Privacy*, 2003.
14. R. Dingledine. Tor and Circumvention: Lessons Learned. In *International Cryptology Conference (CRYPTO)*, 2011.
15. R. Dingledine and N. Mathewson. Design of a Blocking-Resistant Anonymity System. *The Tor Project, Technical Report*, 11:15–16, 2006.
16. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, 2004.
17. T. L. Douglas. The Importance of VoIP. `http://ezinearticles.com/?The-Importance-of-VoIP&id=4278231`, May 2010.
18. Dynamic Internet Technology. Dynaweb. `http://www.dit-inc.us/dynaweb`, Nov. 2013.
19. N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. R. Karger. Infranet: Circumventing Web Censorship and Surveillance. In *USENIX Security Symposium*, 2002.
20. N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger. Thwarting Web Censorship With Untrusted Messenger Discovery. In *Privacy Enhancing Technologies Symposium (PETS)*, 2003.
21. R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy. Vanish: Increasing Data Privacy with Self-Destructing Data. In *USENIX Security Symposium*, 2009.
22. J. Geddes, M. Schuchard, and N. Hopper. Cover Your ACKs: Pitfalls of Covert Channel Censorship Circumvention. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.
23. Global Internet Freedom Consortium (GIFC). The Great Firewall Revealed. `http://www.internetfreedom.org/files/WhitePaper/ChinaGreatFirewallRevealed.pdf`, Dec. 2002.
24. C. Gulcu and G. Tsudik. Mixing E-Mail With Babel. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 1996.
25. A. Houmansadr, C. Brubaker, and V. Shmatikov. The Parrot Is Dead: Observing Unobservable Network Communications. In *IEEE Symposium on Security and Privacy*, 2013.
26. A. Houmansadr, T. Riedl, N. Borisov, and A. Singer. I Want My Voice to Be Heard: IP Over Voice-Over-Ip for Unobservable Censorship Circumvention. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2013.
27. C. S. Leberknight, M. Chiang, H. V. Poor, and F. Wong. A Taxonomy of Internet Censorship and Anti-Censorship. `http://www.princeton.edu/~chiangm/anticensorship.pdf`, Dec. 2012.
28. B. Leiba. RFC 2177: IMAP4 IDLE Command, Jun 1997.
29. J. Leskovec and E. Horvitz. Planetary-Scale Views on a Large Instant-Messaging Network. In *International Conference on World Wide Web (WWW)*, 2008.
30. B. N. Levine and C. Shields. Hordes: A Multicast Based Protocol for Anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.
31. R. MacKinnon. *Race to the Bottom–Corporate Complicity in Chinese Internet Censorship*. Human Rights Watch (HRW), 2009.
32. N. Mathewson. The Tor Project – A Simple Obfuscating Proxy. `https://gitweb.torproject.org/obfsproxy.git`, Nov. 2013.

33. D. McCoy, J. A. Morales, and K. Levchenko. Proximax: A Measurement Based System for Proxies Dissemination. In *International Conference on Financial Cryptography and Data Security (FC)*, 2011.
34. R. McMillan. China's Great Firewall Spreads Overseas. `http://www.networkworld.com/news/2010/032510-chinas-great-firewall-spreads.html`, Mar. 2010.
35. H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.
36. P. Papadopoulos, A. Papadogiannakis, M. Polychronakis, A. Zarras, T. Holz, and E. P. Markatos. K-Subscription: Privacy-Preserving Microblogging Browsing Through Obfuscation. In *Annual Computer Security Applications Conference (ACSAC)*, 2013.
37. R. Perlman. The Ephemerizer: Making Data Disappear. *Journal of Information System Security (JISSec)*, 1:51–68, 2005.
38. A. Serjantov and P. Sewell. Passive Attack Analysis for Connection-Based Anonymity Systems. In *European Symposium on Research in Computer Security (ESORICS)*, 2003.
39. R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P^5: A Protocol for Scalable Anonymous Communication. In *IEEE Symposium on Security and Privacy*, 2002.
40. Statista. Daily Time Spent Playing Video Games Per Capita in the United States. `http://www.statista.com/statistics/186960/time-spent-with-videogames-in-the-us-since-2002`, Sep 2014.
41. Statistic Brain. Skype Statistics. `http://www.statisticbrain.com/skype-statistics`, Sep 2012.
42. UltraReach Internet Corporation. Ultrasurf. `http://www.ultrasurf.us`, Nov. 2013.
43. K. Walsh and E. G. Sirer. Experience With an Object Reputation System for Peer-To-Peer Filesharing. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2006.
44. Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov. CensorSpoofer: Asymmetric Communication Using IP Spoofing for Censorship-Resistant Web Browsing. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.
45. Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh. StegoTorus: A Camouflage Proxy for the Tor Anonymity System. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.
46. B. Wilkins. 25 Shocking Facts About Chinese Censorship. `http://www.onlinecollege.org/2009/07/05/25-shocking-facts-about-chinese-censorship`, Jul. 2009.
47. S. Zander, G. Armitage, and P. Branch. Covert Channels in Multiplayer First Person Shooter Online Games. In *IEEE Conference on Local Computer Networks (LCN)*, 2008.
48. S. Zander, G. Armitage, and P. Branch. Reliable Transmission Over Covert Channels in First Person Shooter Multiplayer Games. In *IEEE Conference on Local Computer Networks (LCN)*, 2009.
49. A. Zarras, K. Kohls, M. Dürmuth, and C. Pöpper. Neuralyzer: Flexible Expiration Times for the Revocation of Online Data. In *ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2016.

50. W. Zhou, A. Houmansadr, M. Caesar, and N. Borisov. SWEET: Serving the Web by Exploiting Email Tunnels. In *Privacy Enhancing Technologies Symposium (PETS)*, 2013.
51. T. Zhu, C. Bronk, and D. S. Wallach. An Analysis of Chinese Search Engine Filtering. *arXiv preprint arXiv:1107.3794*, 2011.
52. T. Zhu, D. Phipps, A. Pridgen, J. R. Crandall, and D. S. Wallach. The Velocity of Censorship: High-Fidelity Detection of Microblog Post Deletions. In *USENIX Security Symposium*, 2013.