2011 IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing

Camouflaged Private Communication

Amir Herzberg*, Ely Porat*, Nir Soffer† and Erez Waisbard*
* Department of Computer Science, Bar-Ilan University, Ramat-Gan, 52900, Israel
Email: {herzbea,porately,waisbard}@cs.biu.ac.il
† Independent Consultant, Givataim, 53600, Israel
Email: nirsof@gmail.com

***Abstract*—Communication such as web browsing is often monitored and restricted by organizations and governments. Users who wish to bypass the monitoring and restrictions often relay their (encrypted) communication via proxy servers or anonymizing networks such as Tor. While this type of solution allows users to hide the content of their communication and often bypass the restrictions, the mere use of a proxy may alert authorities, placing the users at risk. We introduce a simple and efficient construction, leveraging general purpose services, which allows users to hide their communication from active adversaries without relying on prior trust. Finally, we implement a proof of concept showing that camouflaged web browsing is possible using legitimate services such as Gmail without requiring software installation on the client side.**

I. Introduction

Many organizations and governments monitor the traffic of their users and block access to certain web sites. This is often done with good intentions, e.g. in the case of an organization that wishes to protect its employees from web sites containing viruses or malware. However it is also done by governments who wish to block access to sites that do not align with the government's view.

This type of filtering has its limitations and users are able to bypass such restrictions by using a proxy server to relay the communication (either by setting a dedicated server or by using public proxies). However, we note that simply relaying the traffic through a proxy server is not enough to protect users as monitoring tools can perform deep packet inspection and detect which sites are being accessed. While this can be solved by an encrypted channel, firewalls can still block the communication based on the destination address (e.g. when using a known anonymizing server).

Another approach for bypassing the restrictions is by using a distributed anonymizing network such as Tor[3]. Tor works by having users run an onion proxy on their machine. The Tor software sets a virtual circuit, using multi-layer encryption with each router in the route peeling one layer of encryption until the cleartext packet is forwarded on to its original destination. Viewed from the destination, the traffic appears to originate at the Tor exit node. While Tor routers are dynamic and traffic is encrypted, the protocol does not try to hide its usage and communication to Tor routers is easily detected, for example, by observing the certificate that they exchange when setting the SSL session [5]. In addition, firewalls can block the initial access to a directory server[9] and may also block encrypted communication except to a small list of approved sites.

The methods above provide anonymity, but do not hide the fact that the user is using anonymized communication, which may be enough to alert the authorities. A method for communicating with a proxy server in a way that hides the request in plain HTML was suggested by Feamster et.al[6]. They suggested an architecture, called Infranet, that creates a covert tunnel between a requester and a responder which looks like a normal communication between a user and a web server to a passive eavesdropper. Infranet requesters install a special software that is configured with IP addresses and public keys of responders, encode the requests using the keyed encoding function and get the hidden reply in a jpeg using steganography. This works well as long as the censor is unaware of the IP addresses of the responders, however, a censor with access to the requester's software can learn these IP addresses and then either log the IP addresses of user that connect to them or block them. Learning the identity of proxy servers and then blocking them, was done by Wikipedia who blocked all the Tor relays in an attempt to prevent Tor users from anonymously defacing Wikipedia pages.

In this work we suggest a new approach to private communication that avoids the above shortcomings. We introduce the notion of *Camouflaged Private Communication*, a method for hiding one communication session within another using a general purpose service such as Gmail. Leveraging the built-in features of such services allows us to implement secure communication without requiring installation of any software on the client side. This makes our solution suitable also for people who access the internet from public locations such as internet cafes.

The main application we see for this concept is *Camouflaged Browsing* that hides the fact that browsing takes place. We implement a camouflaged browsing proof of concept using Gmail infrastructure by sending a requested URL and receiving a reply with the content of the requested page. Since the communication is encrypted, to an eavesdropper the communication appears to be a typical communication with the Gmail server.

The main idea behind our construction is to have parties communicate through a trusted general purpose service, such as web mail. In the case of Gmail, all communication between the parties and the Gmail server is encrypted and relayed through the Gmail server. Thus, an eavesdropper cannot learn

978-0-7695-4578-3/11 $26.00 © 2011 IEEE
DOI

the usernames of the communicating parties and the communicating parties are not aware of the IP addresses of each other. In addition to the built-in anonymity, using a central service gives us an easy way to detect who is currently connected to the service (which is typically one of the hardest things to implement securely in a distributed system). Using this technique we are able to efficiently achieve a secure solution, but we rely on the general purpose service not to reveal the private information to the censor. In section IV we discuss why we believe that this is a reasonable assumption and why we think it is more practical for a large scale system than prior assumptions that were made by previously suggested solutions.

II. Camouflaged Browsing

In this section we describe the notion of *Camouflaged Browsing*. Loosely speaking camouflaged browsing refers to a web browsing session that is hidden within another communication. The visible communication is considered as 'allowed' while the 'hidden' browsing communication goes unnoticed by anyone who monitors this communication. We model the scenario using two domains: one is the 'restricted' domain $\mathcal{R}$ and the other one is the 'free' domain $\mathcal{F}$. Parties communicate over an open network with unauthenticated channels and all the communication between $\mathcal{R}$ and $\mathcal{F}$ goes through a censor $\mathcal{A}$. A camouflaged browsing session takes place between a camouflaged client $\mathcal{C} \in \mathcal{R}$ and a camouflaged server $\mathcal{S} \in \mathcal{F}$, where $\mathcal{F}$ is also communicating with the web site $\mathcal{W}$ per $\mathcal{C}$'s request. Both $\mathcal{C}$ and $\mathcal{S}$ communicate using a legacy infrastructure that is assumed to be honest (e.g. the Gmail service that we use in Section III). Let S_1 be a camouflaged browsing session and let S_2 be a typical session between two legacy service's users. We say that a camouflaged browsing scheme is secure if no censor is able to tell which of the sessions is a camouflaged browsing session.

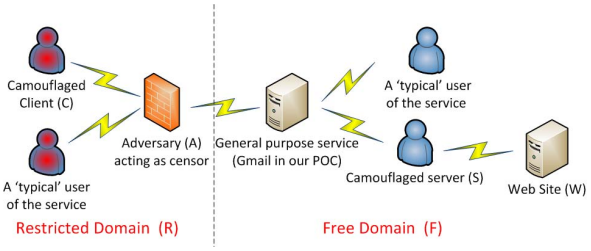

Fig. 1. Camouflaged Browsing

III. Gmail Based Implementation

We now turn to describe an implementation that leverages Gmail's infrastructure. We use Gmail not only because of its convenient infrastructure, but also since this is an encrypted service that is rarely blocked. There are three main features that are available to Gmail users when they log into their accounts using the web interface: email, chat and the ability to see which of their Gmail contacts is currently online, all protected by default using HTTPS.

A typical usage by the camouflaged client would be to log into his camouflaged Gmail account using the web interface, pick one of the camouflaged servers that is currently online and send it the desired URL. Upon receiving a URL from a Gmail user, the camouflaged server, would fetch the content of the requested web site, pack it and send it as email to the initiating client of the session. The camouflaged client would locally open the email containing the web page. Intuitively, if everything is done on the client side using the web interface, then all an eavesdropper sees are a few encrypted connections to the Gmail server and cannot tell that the reading of an email is really viewing a remote web page.

A. Components

- **Directory Server:** One of most challenging tasks in anonymous routing is finding which anonymous router is currently connected. When implementing camouflaged browsing using Gmail we use the contact list as a directory server. Naturally, users should create new usernames that are not linked to their real identity before using this system. In addition, in order to select a camouflaged server, the usernames that represent these servers should first appear on its contact list. In Gmail this is done by adding them and that means that we need someone trusted to manage and publish the usernames of the servers (e.g. an organization like the EFF). If someone wishes to act both as a client and sometimes as a server[1], he should create two separate usernames, one representing himself as client and one as a server.
- **Camouflaged Client:** As we noted from the start, our solution can work without *any* custom software installation on the client side. The initiating user can simply log into the Gmail account, view who is online and send the requested URL using the built-in chat. The reply can be encoded as a multipart MIME message that can be displayed directly in the web interface or by another mail client.

  However, the above method may not be the most convenient, especially if one wishes to do more than read a single static page. If one wants to follow links in the page then the most seamless approach on the client side would be by directing all HTTP requests to a local proxy that runs on the user's machine and redirects all HTTP requests using camouflaged browsing. To accomplish this, the local proxy logs into the Gmail account and sends the URL to one of the usernames that acts as servers for camouflaged browsing. This local proxy also monitors incoming emails for the reply. Once the reply arrives the proxy extracts the information and displays it.
- **Camouflaged Server:** The server needs to listen to incoming URLs that may arrive either by email or by chat. Once it receives the URL, it downloads the web page and all the

[1]For example a user with a laptop that is acting as a client when he is behind the company's firewall during the day and as a server at night when he connect his laptop at home.

files that are necessary to properly display the page and pack them for delivery. For viewing the page in a browser on the client side, the server can create an archive with the page files, to be extracted in the client side. For viewing directly in a mail client, the server can create a multipart MIME message and embed the necessary files as MIME parts in the message. Finally the server sends the reply to the client.

The complete details of the implementation appear in the full version of this paper. The full version, the source code and the username of our camouflaged server are available on our web site: http://nirs.freeshell.org/camouflage-browsing/.

B. Performance

In this section we analyze the performance of our Camouflaged Browsing implementation. The test machine is connected to the internet through ADSL connection with a download bandwidth of 1269KB/s and upload bandwidth of 103KB/s.

We have tested 4 different URLs (with 100 measurements for each) of different sizes. The time it took to open these sites with the web browser ranged from 1.78 seconds to 5.54 seconds (depending on the amount of content the site displays). Using camouflaged browsing the time to open these web sites ranged from 16.8 seconds to 35.32 seconds.

As can be seen in figure 2, most of the time is spent downloading the page and sending the response to the other user. For downloading the page we used Wget[11] which downloads the page resources sequentially. Future implementations can speed up the download process by downloading files concurrently (as done by browsers).

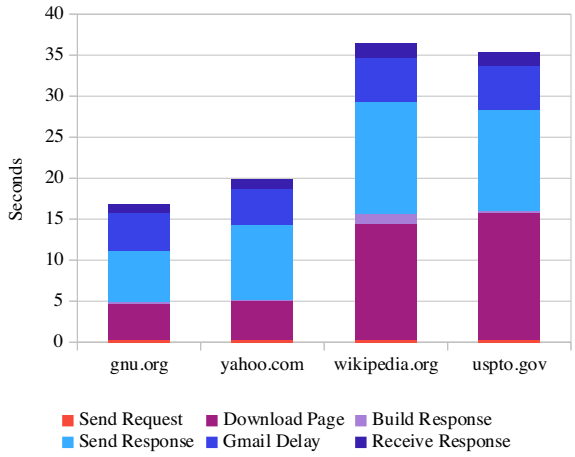

Fig. 2. Camouflaged timing analysis

IV. Security

In this section we analyze the security of the scheme presented in Section III. Throughout this analysis we assume that the Gmail server is honest. Since the Gmail server knows the IP addresses of all who communicate with it, then there is no anonymity in case it is compromised.

A. Leakage of Browsing Pattern

In [4] Chen et. al. showed that web applications leak information even if they are encrypted. This is done by looking at the communication pattern to infer the current state of the application and the inputs of the user. For example, they show how to infer the entered text based on the auto-completion suggestion.

Monitoring the encrypted communication between the initiating client and the Gmail server, the adversary would look for a special communication pattern. In the simplified form that we discussed so far, the camouflaged client is sending a short URL and soon after he receives an email. If the user automatically opens the email, then we get a noticeable pattern of sending a short encrypted message to the Gmail server, which is followed by getting a short notification and then opening of a web page.

In order to prevent this attack we need to break the pattern. One suggestion to cope with this is to delay the opening of the message and artificially create a chat session pattern to hide the short incoming mail notification. Recall that all the communication is encrypted so exchanging messages between the client and the server (each of different length and after a short random delay) creates a pattern of a real conversation between two users. For an adversary that monitors the traffic on camouflaged client side, the encrypted incoming mail notification appears just like a chat message. Once the reply is in the inbox, the user can choose how long to wait before opening it. Naturally adding delays and changing the user experience goes against the seamless browsing experience we wanted to achieve, but as these are configurable, we leave it for the user to set these parameters to achieve the best tradeoff between performance and privacy.

Other methods for avoiding pattern detection can include breaking of the message into smaller messages that would be combined on the client side. If one is only interested in the text of the page, the reply email can be much smaller (this would also shorten the time it takes the server to get the content of the page) and may be be transferred in chunks using XMPP.

We further note that patterns are detectable over time. If the user only uses this technique once in a while then there would be no noticeable pattern. However, if it is massively used to open every web page then it would result in a very distinguishable pattern.

B. Corrupted Server

The fact that communication is encrypted and that we are able to prevent the pattern detection does not guarantee that camouflaged browsing cannot be detected by an adversary that is controlling a camouflaged server. There are timing attacks on Tor [7] that aim to learn the identity of the communicating party by observing latency patterns in the communication. While these attacks were tailored to Tor like systems and are

not applicable 'as is' to our system, the ideas in[7] could be extended to use patterns that are noticeable in our system.

One way we suggest to cope with such an attack is by disconnecting the client while waiting for the reply. This allows the Gmail server to 'absorb' any reply pattern. When the client finally reconnects the censor always sees a consistent view of a client downloading the encrypted email headers. Using this method the client is also able to spot unusual patterns in its mailbox and blacklist the servers who originated it.

Another way to avoid this type of attack is by avoiding these 'bad' servers. This can be done for example in the case that the camouflaged client knows the identity of a trustworthy server[2]. If the camouflaged server is chosen at random, then the probability of this attack to work is proportional to the number of 'bad' servers.

The one thing that the adversary does learn when controlling the camouflaged server is the username of the client, but since this is a pseudonym, anonymity is preserved.

C. Secrecy of IP Addresses

That fact that we use the Google server to relay all the communication ensures that the IP addresses of camouflaged clients and servers are known only to the Google server. The only thing that the client and server know about each other is the Gmail username and that is the most that an adversary can learn. We note that this particular feature is unique to Gmail and other web based mail services often disclose the IP address from which a user connected to the mail service.

D. Authenticating the Gmail Server

Recall that our underlying assumption is that the Gmail server is honest and that all communication to it going through a secure authenticated channel using HTTPS. This holds as long as no one can impersonate the server. A Recent work by Soghoian and Stamm [8] discussed the possibility of government agencies to compel certificate authority to issue a false SSL certificate that can be used to hijack individual's secure web based communication. They proposed using a browser add-on to notice such an event that can be useful in the present context.

E. Coercing Google

So far we described our scheme under the assumption that the Gmail server is secure, however one should not build the entire security on the foundation of a sole service provider. Clearly, the ideas that we presented are not limited to Gmail and this work should be extended to support similar infrastructure such as the ones provided by Yahoo, Microsoft and Skype. This would ensure that we do not rely on a single service provider that may be coerced to collaborate with the government. The more services we can use, the smaller the chances they would be blocked.

[2]An organization like the EFF may choose to post a list of trusted usernames on its site.

While companies such as Google may be willing to disclose the identity of individuals in response to a court order, they are unlikely to be willing to actively monitor and disclose the identities of all the users who communicate with camouflaged servers. Since our scheme does not allow a censor to learn the identity of the communicating parties from monitoring traffic, the censor is unable to track these users and then produce individual orders for disclosure.

We stress that some trust is required in order to allow secure communication between clients and servers. We believe that *for a large scale system*, trusting a general purpose service such as Gmail is more practical than establishing this trust out-of-band or assuming that the censor cannot play the part of the client and obtain the IP address of the servers.

V. Future Work

We have introduced the concept of *Camouflaged Private Communication* and presented an efficient and secure scheme for *Camouflaged Browsing* that is based on Gmail, but as noted before, the implementations should also support similar infrastructure such as the ones provided by Yahoo, Microsoft and Skype. In addition to improving security, implementing camouflaged communication using various services would also improve the usability.

Our proof of concept made use of the encrypted connections that Gmail offers. However, it can be extended to use infrastructures that do not offer encrypted connections by combining ideas from public key cryptography along with steganography [1], similarly to what was done in [2], [6]. This would open the door to a wide range of infrastructures such as Facebook and other social networks and online gaming sites (where lots of data can be hidden in the massive data they exchange).

References

[1] L. von Ahn, N. J. Hopper: Public-Key Steganography. In *EUROCRYPT 2004*. pages 323–341

[2] S. Burnett, N. Feamster, S. Vempala: Chipping Away at Censorship Firewalls with User-Generated Content. In *USENIX Security Symposium 2010*. pages 463–468

[3] R. Dingledine, N. Mathewson and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium 2004*, pages 303–320.

[4] S. Chen, R. Wang, X. Wang, K. Zhang. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In *IEEE Symposium on Security and Privacy 2010*, pages 191–206

[5] Detecting Tor on your Network.

[6] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, D. Karger: Infranet: Circumventing Web Censorship and Surveillance. In *USENIX Security Symposium 2002*, pages 247–262 Available at http://blog.vorant.com/2005/01/detecting-tor-on-your-network.html

[7] N. Hopper, E. Y. Vasserman, E. Chan-Tin: How much anonymity does network latency leak? In *ACM Transactions on Information and System Security 2010* Volume 13(2)

[8] C. Soghoian, S. Stamm: Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL (2010). Available at SSRN: http://ssrn.com/abstract=1591033

[9] Tor partially blocked in China. In*Tor Blog*. Available at: https://blog.torproject.org/blog/tor-partially-blocked-china

[10] D. Wendlandt, D. G. Andersen, A. Perrig: Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *USENIX Annual Technical Conference 2008*. pages 321–334

[11] GNU Wget. Available at http://www.gnu.org/software/wget/