vpwns: Virtual Pwned Networks

Jacob Appelbaum
Security and Privacy Research Lab
University of Washington & The Tor Project
ssladmin@uw.edu

Marsh Ray
PhoneFactor, Inc.
marsh@extendedsubset.com

Karl Koscher
Security and Privacy Research Lab
University of Washington
supersat@uw.edu

Ian Finder
Security and Privacy Research Lab
University of Washington
finder@uw.edu

Abstract

User-accessed Virtual Private Network systems allow authorized users remote access to protected or otherwise privileged networks while avoiding dependence on ISPs along the route for data confidentiality and integrity. This direct expression of the internet's end-to-end principle of security is generally accepted as a highly successful design.

VPN services and technology advertising censorship circumvention, resistance to data retention, and anonymity as features are proliferating rapidly. But it is unclear that these security properties were included in the original design requirements of VPN protocols and product implementations. Experience with dedicated anonymity networks (e.g., Tor) shows that strong anonymity is not achieved by accident. The 'P' in VPN notwithstanding, not all privacy methods are equal or strongly anonymizing, which opens opportunities for attackers when VPN-based systems are used for anonymity or even simple censorship circumvention.

This paper evaluates VPN anonymity, security and privacy features including identity, geographic location, confidentiality of communications, and generalized security issues such as reachability and prevention of network tampering. We find many popular VPN products are susceptible to a variety of practical user deanonymization attacks. Weaknesses stem from lack of security analysis of the composition of VPNs, applications, and the TCP/IP stack on each respective operating system. Although we describe some potential mitigations for vendors, the primary goal of this paper is to raise awareness of the inherent risks which come from repurposing off-the-shelf VPN systems to provide strong anonymity.

1 Introduction

Virtual Private Network systems (VPNs), although originally conceived as a tool to enable private networks to participate in the economies of scale of the public internet, today are often used for purposes beyond merely connecting private networks together. The availability of mature VPN implementations and the encryption and access control they provide would seem to make VPNs an attractive option for systems that provide user anonymity or resist censorship. Nevertheless, many anonymity and censorship circumvention systems that are built on top of VPNs are easily subverted by active (and sometimes passive) attacks. Even with the best cryptography and careful coding practices, the security of the VPN may be bypassed entirely.

The anonymity community often ignores VPN-based solutions, considering them obviously flawed against strong attackers. Nevertheless, these solutions are routinely employed by users who believe the claims of vendors.

Whenever a tool is pressed into service to provide data security properties for which it was not originally designed and tested, the potential for subtle security flaws greatly increases. In the particular case of a VPN used as an anonymizing service, the issues seem to arise primarily from the conventional relationship the VPN client software has with the endpoint system's routing table. After all, to the kernel it is "simply another network", so the most common VPN implementation technique is for the active VPN connection to appear to the system as another virtual network adapter. Consequently, enforcement and application of many VPN security properties depend greatly on the local routing table.

Applications generally prefer to remain unaware of network state changes, at best they might implement some notion of an "offline" mode. Because monitoring the changes in the lower layers of the network stack is not an important goal in computing for most users, user interfaces tend to minimize such details as long as basic connectivity is working. But these networking subsystem properties that are unimportant details under normal circumstances silently become security-critical consider-

ations once the user begins relying upon the routing table to ensure their anonymity.

2 VPN Security Mechanisms

Routing table based network security is normally a perfectly acceptable architecture for classic VPN deployment scenarios: if the routing table is wrong, the packet simply cannot be delivered. Even when an unencrypted packet does manage to escape via a physical interface, usually it bears an RFC 1918 [3] private use address on either the source or destination so the packet is not likely to make it very far outside of the trusted network. Those who depend on such systems will eventually notice the failed connections, complain to their network administrator and the misconfiguration will be resolved.

But when the goal of the system is to provide strong user anonymity, the requirements become much more stringent. Even a single leaked DNS query or TCP SYN packet may be enough to reveal the user's identity entirely and subject them to consequences much greater than those of a failed connection. Under these new requirements, the method of securing traffic via the endpoint system's routing table is insufficient. It proves vulnerable to a number of generic problems that have the effect of expanding the user's attack surface dramatically.

3 Security Claims and Properties

Many VPN providers or products seem to overpromise in terms of where their products and tools work, making extremely bold claims about privacy, security, and anonymity without having had their claims evaluated to the standards found in the anonymity community.

For example, AnchorFree's Hotspot Shield website [7] claims the following:

- "VPN encrypts all traffic."
- "Protect yourself in Wi-Fi hotspots."
- "Hide your IP and ensure anonymous browsing."
- "Protect yourself from snoopers at Wi-Fi hotspots, hotels, airports, corporate offices."

We also find that Private Tunnel [16] makes similar claims:

- "Preventing anyone from viewing or snooping your data exchange across the Internet"
- "Preventing anyone from seeing your public IP address"

These claims are unreasonably absolute and they specifically fail to disclose the privileges afforded to the service operators by the design of the system as a whole.

3.1 Use cases

Many VPN providers or products seem to promise perfect privacy and security. They rarely define cohesive threat models or explain details about the security evaluations they expect users to make. It is exceedingly rare to find a provider or product development team that discloses such threat model discussions or evaluations openly. In many cases the VPN vendors do not properly deploy SSL/TLS for their general websites or download sites.

We consider users to belong to four primary use cases:

- Users who require access to internal or otherwise protected resources
- Users who wish to avoid Firesheep [11] or other small attackers (e.g., cafs Wi-Fi sniffing)
- Users who wish to access the wider internet without censorship or surveillance
- Users who wish to reposition themselves into different legal frameworks (e.g., geographically limited content).

4 Security Implementation Realities

The security realities of VPN deployments diverge significantly from the claims made by vendors.

When a client connects to or disconnects from a VPN service, a significant amount of reconfiguration must be performed on the client network stack. In order for effective anonymity to be achieved most of this reconfiguration must complete successfully. Device drivers must be loaded and unloaded, routing tables and name resolution settings must be adjusted, some existing connections must be dropped, and some applications restarted. To this end, many VPN systems provide custom client software. Some VPN client packages provide rich functionality, allowing desktop settings to be reconfigured or even arbitrary scripts pushed to the client to be executed with Administrator privileges. For example, a corporate VPN, may require a qualifying anti-malware package be installed and client OS updates be applied before allowing clients access to the network.

The term 'split tunnel' refers to the technique of passing some, but not all, net traffic over the VPN. This ability is sometimes very useful or even necessary for proper operation. For example, most users prefer not to lose access to local resources (e.g., printers and other network-attached peripherals) while connected to the VPN. But how does the VPN implementation know just where to split the tunnel, i.e., how wide should the non-VPN routing table entry be? It is not sufficient to simply retain the route associated with the physical adapter because it is not known whether it represents a trusted home LAN, or a hostile airport Wi-Fi.

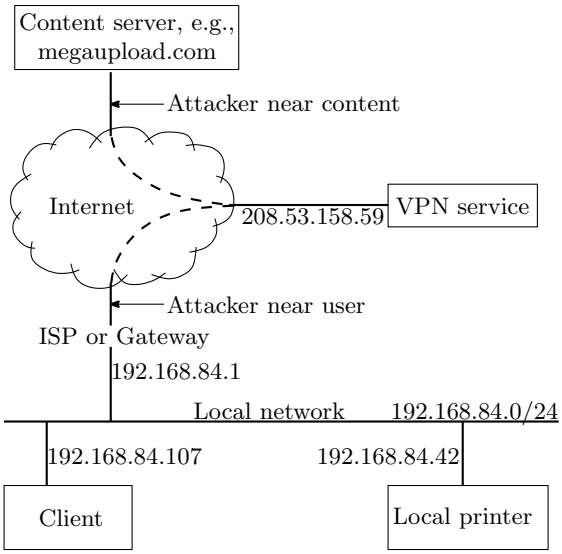

Figure 1: Attack scenarios corresponding to the routing table of Listing 1.

While the issue of split tunneling is known to VPN-savvy users and administrators, many assume that software that installs a default route on the client will resolve all the related architectural issues. However, under the more stringent threat model implied by even the weakest of anonymity requirements, split tunneling represents an architectural weakness that may be mitigated only by considering the entire context of the networking stack. To this end, some VPN systems will push a more-specific route. This still leaves a split through which traffic may leave the client without the protection of the VPN, it's simply a smaller split.

There are issues at the service level too. We find that many vendors claim absolute anonymity on the same pages where they explain how they log extensive user data, enough to expose their users to retaliation if those vendors become motivated to do so for legal, political [1], or economic reasons. While some companies may reject data retention, it may not be a matter of choice because of the overall architecture. Country-wide dragnets on data may effectively create those logs which VPN providers refuse to create themselves.

4.1 Security through routing tables

The security of most popular VPNs is a matter of IP routing as much as cryptography. Virtual devices are often created by VPN software, such as OpenVPN, and all traffic passed through those devices is encrypted. However, we found that VPNs generally do not use kernel or other packet filtering to ensure that only the VPN software itself sends traffic out of the actual network adapters. Thus the choice of which packets are encrypted or not is actually made by the OS in a manner that may be less than fully consistent with the security goals, which are merely implicit.

```
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      10.36.13.4      10.36.13.4      1
          0.0.0.0          0.0.0.0    192.168.84.1  192.168.84.107     21
       10.36.13.4  255.255.255.255      127.0.0.1       127.0.0.1     50
  10.255.255.255  255.255.255.255      10.36.13.4      10.36.13.4     50
        127.0.0.0        255.0.0.0      127.0.0.1       127.0.0.1      1
      192.168.84.0  255.255.255.0  192.168.84.107  192.168.84.107     20
    192.168.84.107  255.255.255.255      127.0.0.1       127.0.0.1     20
    192.168.84.255  255.255.255.255  192.168.84.107  192.168.84.107     20
     208.53.158.59  255.255.255.255    192.168.84.1  192.168.84.107     20
255.255.255.255  255.255.255.255      10.36.13.4      10.36.13.4      1
255.255.255.255  255.255.255.255  192.168.84.107  192.168.84.107      1
```

Listing 1: Routing table for the Perfect Privacy PPTP service on Windows XP. The client's IP address on the local network is 192.168.84.107 while the remote VPN server is 208.53.158.59. Some redundant entries elided for space.

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.27.0.1      0.0.0.0         UG    0      0        0 tun0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth1
172.27.0.0      0.0.0.0         255.255.252.0   U     0      0        0 tun0
192.168.2.0     0.0.0.0         255.255.255.0   U     1      0        0 eth1
198.252.153.26  192.168.2.1     255.255.255.255 UGH   0      0        0 eth1
```

Listing 2: Example OpenVPN routing table for riseup.net on GNU/Linux. The client's IP address on the local network is 192.168.2.1 while the remote VPN server is 198.252.153.26.

The client computer sends data through different interfaces by consulting its local routing table. For this architecture to function at all, the client must be able to send packets to the remote server peer via the gateway that would have handled the default route were the VPN not connected.

This is seen in the routing table presented in Listing 1. The previous default route (*0.0.0.0*) is overridden by specifying a lower cost metric for the VPN service. Note that even though the VPN-installed split tunnel is as narrow as possible, a single IP address *208.53.158.59*, all the existing routing table entries have been retained. A remote attacker need only to induce a client application to attempt to connect to that IP or any of the local network in order to result in a packet being sent from the client outside the VPN, potentially de-anonymizing the user.

We present two representative samples of routing tables found in the wild from Perfect Privacy [2] and RiseUp! [4]. Similar issues as in Listing 1 are also present in Listing 2.

A much better architecture for systems needing to avoid such breakout or bypass bugs is to require explicit proxy configuration in applications themselves. When the configured proxy is unavailable these kinds of systems are much more likely to "fail closed", i.e., fail into a secure configuration rather than assume a working but insecure configuration. Explicitly configuring applications

is less convenient, but having an explicit rather than implicit security feature also engages the user in actively managing their defense in depth strategy. This is the model of Torbutton [15] and TorBirdy [19, 8, 13] and it provides applications with contextual security information that is otherwise missing. Because the failure modes of privacy systems are both subtle and serious, this degree of contextual awareness of applications and users is essential and largely missing when the data is injected into the general purpose networking stack.

5 Attack Scenarios

Scenarios are categorized here according to on which side of the VPN service the attacker is positioned (near user or near content) and whether or not the attacker must modify the network data or may simply observe it (active or passive). Other combinations are possible. For example, an attacker near an unsecured Wi-Fi hotspot could passively monitor users' traffic in coordination with sending packets to a content server bearing a forged source IP address of the VPN service.

An additional scenario, that of the compromised VPN service provider, was deemed distinctive enough to merit its own category. As it is the union of all possible capabilities for a network-based attacker, it represents the worst-case scenario.

5.1 Passive attacker near content

A passive eavesdropper near the content provider has the ability to observe the network traffic passing between a set of websites and set of users. However, they lack the ability to see the traffic at the level of any particular user's ISP. Typically a user will have 'opted in' to interacting with at least one of these sites. However, even minimal interaction with a major commercial web site tends to spawn a significant number of child requests to third-party sites with which there is some type of data or revenue sharing agreement in place (e.g., google-analytics.com).

The goal of such an attacker is to learn something about the identity of some or all of the sites' users, or at least correlate some of the traffic across contexts.

This scenario could arise in several ways. The hosting provider or the website itself could be compromised. The hosting provider or their ISP could be compelled to disclose their traffic due to an adverse legal environment. A national telecommunications company or firewall system could monitor the traffic for an entire nation of users. But fundamentally this situation is the norm that naturally arises whenever the business interests of a website trading in personal data diverges from the privacy expectations of that site's users. However, the traffic such a site can observe is usually restricted to its own and that of its partners.

Other studies have shown passive browser fingerprinting to be effective at correlating user identities. [9] VPN-based systems in which a user shares the same browser with non-anonymous web surfing are nearly certain to transfer at least one cookie or other session identifier via the VPN session, which is enough for such an observer to de-anonymize the user via correlation with their non-VPN identity.

5.2 Active attacker near content

This type of attacker has all the capabilities of the previous attacker with the additional ability to inject and/or modify traffic.

Such an attacker is in an excellent position to intercept a content request from an anonymity-seeking browser user and modify it according to his needs. In theory, websites can protect their users with consistent use of SSL/TLS HTTPS. However in practice, finding sites and users that are perfectly consistent in their avoidance of plain HTTP connections is extremely rare [17]. A single request for any resource type (except perhaps images) is all that is needed to inject arbitrary script into the browser and have it execute with the origin of the requested site.

From that point, the user's browser is largely under control of the attacker who can induce it to request any URL. The full range of options available to an attacker having such script injection capabilities are well known and to the security of the web application itself it represents a fundamental compromise.

Even in the case of the most restrictive possible routing table the non-VPN routing table entry must contain at least one address: that of the VPN server itself. By requesting a resource from this address, the attacker can induce the user's browser to attempt to establish a TCP connection in the clear to this address on all but a few well-known port numbers which are blocked by the browser.

We are pointing out for the first time, to our knowledge, that this capability can also be leveraged by an attacker to cause the VPN to leak packets which, under the right cicumstances, could be sufficient to deanonymize or to leverage and to increase specific attack surfaces of the VPN user. For example, requests to any resources on the correct side of the split tunnel have the potential to access the user's local systems and techniques for scanning internal systems using ordinary HTML-originated requests or active content like Javascript, Flash, and Java, are well developed. If an internal scan discovers even a single server (e.g., an organizational intranet server or router configuration page) the content it hosts is likely to reveal information about the location of the user. Modern browsers make some attempts to prevent the access of

internal resources from external web pages, but the effectiveness of such mitigations is invariably limited by the very same challenge of programmatically distinguishing "internal" from "external" addresses as is faced by the VPN's split tunnel autoconfiguration process.

During development of this paper, the authors conceived of several other attacks of this type. However, none of the VPN systems tested held up against far simpler methods and they were not investigated further. For example, none of the VPNs tested adjusted the IPv6 routing table as part of connection set up. If a client has a working connection to the public IPv6 net, an active attacker can simply induce the client to request a resource he provides via IPv6 only. When the client request arrives at the attacker's server, the attacker learns the client's IPv6 address. As client IPv6 addresses are often constructed from physical adapter MAC addresses [10], such an attack could be particularly effective.

As a further example of the flawed architecture, consider a VPN that provides clients with recursive DNS resolvers on the same IP address as the VPN server itself. The split tunnel architecture ensures that the DNS packets, usually without DNSSEC, flow outside of the VPN tunnel's protection. Thus even if all non-DNS connections are protected, an attacker may control many, if not all of those connections with well known DNS attacks.

We further considered many variations on tagging that leverage XMPP, X.509 certificate OCSP server fields, fully protected web browser DNS prefetching, simple image tags, CSS, Javascript, Java applets and more.

5.3 Passive attacker near targeted user

VPNs are often used to allow remote users secure remote access to protected systems (e.g. a corporate network). A common scenario that comes to mind (and is perhaps borne out by reality) places the attacker on the same airport, hotel, or coffee shop Wi-Fi as the target user. Similarly, users under repressive political regimes typically face the challenge of being in an adversarial relationship [14] with their own direct connectivity provider (typically a nationalized telecommunications company or ISP) when it comes to their anonymity. In the United States, many users of peer-to-peer file sharing applications have few alternatives for high-speed internet other than to purchase service from the same "vertically integrated" content industry that aggressively pursues legal action against file sharers (sometimes erroneously).

The traditional view of the attack realities has envisioned attackers near the end user. Yet, even among VPN systems that provide custom client software, there are few examples of special features in place designed to defend the client endpoint from a hostile local networking environment. Features directly in support of the end-user's goals of privacy (as opposed to the server administrator's) from the local operator appear to lose consistently in the ever-present tradeoff between security and functionality, e.g., split tunneling.

Consequently, for the de-anonymizing attacker, it is still quite advantageous to be close to his target.

Clearly the user's ISP will be able to observe their connection to the VPN. A dedicated VPN protocol will stand out as it runs on its own TCP port (or in the case of PPTP, a dedicated non-TCP/UDP protocol on top of IP). Typically the username is sent in the clear. Likewise, IPsec based services will show distinctive features in packet headers. SSL/TLS based VPNs seem the most likely to blend in with ordinary traffic, but unless they are designed specifically for censorship-resistance they too will tend to have distinctive features which readily permit identification by traffic analysis and minimally-deep packet inspection as is often seen with the Tor [22] network's ongoing arms race with censors.

Perhaps the simplest de-anonymization attack an ISP can perform is to look up the customer account for which the client IP address is currently assigned. Indeed, anti-anonymity regulations (proposed or enacted) typically require retention of this IP address-to-customer association data. [5] To prevent users from communicating anonymously some countries even go as far as requiring photo-ID registration before accessing the internet in public cyber-cafs, if not outlawing encryption entirely.

It is an interesting debate whether or not the effectiveness of such an obvious technique is truly a weakness of VPN based anonymity solutions. This IP address-to-customer association data is nearly fundamental to the provisioning of services by the ISP to the customer and most other non-VPN type systems have the same limitations. But the position of the authors is that the security of such a system is only as good as that which its real world users are able to reliably obtain from it. Therefore, those of us who appreciate the inherent limitations of the technology must be careful to evaluate it from the perspective of its users, who are counting on it to secure their anonymity based largely on its marketing representations.

Nevertheless, any functional VPN should not allow the attacker to learn much about the protected resources the targeted user is actually accessing, although traffic analysis alone can be astonishingly effective. [23, 12] Furthermore, it requires the ISP to learn about the IP address (or historical traffic patterns) via some other channel before having a reason to target the user, at least in societies where VPNs and anonymizing services are not inherently illegal.

Another effective user-side attack is simply close observation of every packet sent by the client IP, particularly any which emerge outside of the encrypted VPN tunnel. As former NSA scientist Robert Morris Sr. once

famously [18] said *"Rule 1 of cryptanalysis: check for plaintext"*. The complex client configuration requirements (as described in Section 4) for secure VPN operation ensure ample opportunity for plaintext.

5.4 Active attacker near targeted user

Applications in use on a client computer generally have no knowledge of the VPN. While useful because it avoids per-application configuration or user education, it also means that that applications lack contextual information that can be essential for security, privacy and anonymity. If an attacker were simply to deny all traffic to the VPN host by way of Deep Packet Inspection, it may cause the user to disable or restart the VPN client, or the VPN connection may even restart itself with a watchdog timer of some kind. Until the VPN reconnection is complete, the client's routing table momentarily assumes an unsecured default (or even unpredictable) state. Applications the user expects to be secure now simply connect directly.

5.5 Compromised VPN service provider

We must not overlook the obvious and technically uninteresting attacks. A user's anonymity may be compromised by their VPN provider itself being subject to attacks, an accidental data breach, or coercively compelled disclosure.

Malicious VPN services could be deployed as part of an elaborate social engineering project by the very same adversaries motivating their users' attempts at anonymity. VPN clients inevitably require some degree of installation and configuration by the end users, who will need localized client software and supporting documentation. Those in smaller geographic regions are likely to find a very limited set of choices available. Given the relatively low barriers to entry in becoming a VPN service provider, if this is not already a common attack vector it is likely just a matter of time. There are rumors that the Iranian government may already be engaged in this activity [6] and there is strong evidence that the Syrian government or pro-governmental forces have backdoored commonly available client side software [14] as a method of targeting users.

6 Examples of Technical Countermeasures

It is possible to attempt to mitigate some of the architectural issues in a platform-specific manner. As an example we find that with OpenVPN on GNU/Linux or another TUN/TAP device-based VPN, it is possible to use Netfilter [20] and iptables [21] to ensure that the Linux kernel only allows the VPN software to write packets to the network device and stops unprotected packets from leaving the physical device unless the VPN is sending them.

Operating systems should provide contextual information about the state of the network and track applications that cross-contaminate data flows. A standard method should be developed to ensure that a given VPN will fail closed if it accepts a default route. OS vendors should ensure that only the VPN client software in userspace or in the kernel is able to write to the physical network. This should ensure that entire classes of accidental data leakage are simply blocked with a defense in depth strategy.

Users should beware of the pitfalls outlined in this paper as well as educating themselves about the vendors who provide access or client software. In some cases a VPN might be a perfectly reasonable choice and in others, a system designed for specific risks or threats is a better choice.

7 Conclusion

Providing strong anonymity and censorship resistance in an environment of targeted attacks by intelligent adversaries presents a unique and difficult challenge, one for which the existing data security toolbox is incomplete. Unlike securing data transport in bulk, guarding against all forms of potential information leakage leaves essentially zero margin for error. In order to be effective, anonymity systems need to be designed top-to-bottom for this task. These will remain special-purpose systems for the foreseeable future.

8 Acknowledgements

We would like to thank the University of Washington Security and Privacy Research Lab and other anonymous cypherpunks who contributed valuable feedback. Additionally, we'd like to thank Arturo Filastò from the Tor Project, Elijah Sparrow from Riseup Labs, Zachary Weinberg, Seda Gurses, and Alice Allen.

References

[1] Hidemyass lulzsec fiasco. `http://blog.hidemyass.com/2011/09/23/lulzsec-fiasco/`. [Online; accessed 06-July-2012].

[2] Perfect privacy. `https://www.perfect-privacy.com/about.html`. [Online; accessed 06-July-2012].

[3] RFC 1918. `http://tools.ietf.org/html/rfc1918`. [Online; accessed 06-July-2012].

[4] VPN service from riseup.net. `https://help.riseup.net/en/vpn`. [Online; accessed 06-July-2012].

[5] Protecting Children from Internet Pornographers Act H.R., 1981.

[6] Private conversations with Iranian activists, 2012.

[7] AnchorFree. Hotspot Shield. http://anchorfree.com/hotspot-shield-VPN-download-windows.php. [Online; accessed 08-July-2012].

[8] Jacob Appelbaum. Towards a Torbutton for Thunderbird (torbutton-birdy). https://lists.torproject.org/pipermail/tor-talk/2012-May/024138.html, 2012. [Online; accessed 08-July-2012].

[9] Electronic Frontier Foundation. https://panopticlick.eff.org/. [Online; accessed 06-July-2012].

[10] R. Hinden and S. Deering. Rfc 4291: Ip version 6 addressing architecture. *Retrieved from IETF Tools: http://tools.ietf.org/html/rfc4291*, 2006. [Online; accessed 08-July-2012].

[11] Eric Butler Ian Gallagher. Firesheep. http://codebutler.com/firesheep/, 2010. [Online; accessed 06-July-2012].

[12] IOActive. IOActive Labs Research: I can still see your actions on google maps over ssl. http://blog.ioactive.com/2012/02/ssl-traffic-analysis-on-google-maps.html, 2012. [Online; accessed 06-July-2012].

[13] Sukhbir Singh Jacob Appelbaum. TorBirdy. https://trac.torproject.org/projects/tor/wiki/torbirdy, 2012. [Online; accessed 08-July-2012].

[14] Morgan Marquis-Boire. Iranian anti-censorship software Simurgh circulated with malicious backdoor. https://citizenlab.org/2012/05/iranian-anti-censorship-software-simurgh-circulated-with-malicious-backdoor-2/. [Online; accessed 06-July-2012].

[15] Mike Perry. Torbutton design document. 2010. [Online; accessed 08-July-2012].

[16] Inc. Private Tunnel. Private Tunnel. https://www.privatetunnel.com/index.php/why-private-tunnel.html. [Online; accessed 08-July-2012].

[17] Ivan Ristic. State of SSL. *Talk at InfoSec World*, 2011. [Online; accessed 06-July-2012].

[18] Robert Morris Sr. Notes on crypto '95 invited talks by r. morris and a. shamir. http://www.ieee-security.org/Cipher/ConfReports/conf-rep-Crypto95.html. [Online; accessed 08-July-2012].

[19] tagnaq. Towards a Tor-safe Mozilla Thunderbird. 2011. [Online; accessed 08-July-2012].

[20] H. Welte. The netfilter framework in linux 2.4. In *Proceedings of Linux Kongress*, 2000.

[21] H. Welte. What is netfilter/iptables, 2004.

[22] P. Winter and S. Lindskog. How china is blocking tor. *Arxiv preprint arXiv:1204.0447*, 2012.

[23] Shuo Chen, Rui Wang, XiaoFeng Wang, Kehuan Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In Security and Privacy (SP), 2010 IEEE Symposium on (2010), 2010.