Decoy Routing: Toward Unblockable Internet Communication

Josh Karlin *jkarlin@bbn.com*
Daniel Ellard *dellard@bbn.com*
Alden W. Jackson *awjacks@bbn.com*
Christine E. Jones *cej@bbn.com*
Greg Lauer *glauer@bbn.com*
David P. Mankins *dm@bbn.com*
W. Timothy Strayer *strayer@bbn.com*
Raytheon BBN Technologies

Abstract

We present decoy routing, a mechanism capable of circumventing common network filtering strategies. Unlike other circumvention techniques, decoy routing does not require a client to connect to a specific IP address (which is easily blocked) in order to provide circumvention. We show that if it is possible for a client to connect to any unblocked host/service, then decoy routing could be used to connect them to a blocked destination without cooperation from the host. This is accomplished by placing the circumvention service in the network itself – where a single device could proxy traffic between a significant fraction of hosts – instead of at the edge.

1 Introduction

It is increasingly common for network service providers to block, filter, redirect, intercept, or even modify traffic between clients on their networks and popular and controversial websites or other Internet-based services [6, 19, 21]. In response, resourceful users have devised a number of proxy services to circumvent these filters, but these services typically suffer from the common flaw of having an easily-identifiable traffic signature, such as having the proxy bound to a well-known IP address.

We use the term *adversary* to describe a network service provider that wishes to deny access from clients on its network to a given set of hosts or services in the greater Internet, and we say that a host or service is *blacklisted* if it is a member of this set.

In the absence of an adversary, a client can reach any destination server unimpeded. However, if an adversary blocks traffic between the client and the destination then the client must reach the destination using an indirect method, such as a proxy server placed outside of the filtered network. At a high level, this is the mechanism used by popular services such as Tor [13], Global Pass [4], and anonymizer.com [2]. Clients connect to a server located outside of the filtered network, and the server connects to the blacklisted destination on the client's behalf. Although these services use different techniques (VPNs, SOCKS proxies, HTTP proxies, content forwarders) we categorize them collectively as a *circumvention proxy* or proxy.

The problem with the proxy approach to circumvention is that the adversary can also blacklist the circumvention proxy, preventing communication between the client and the proxy. Circumvention tools such as darknets can try to hide the addresses of their proxies but this makes the tool less generally accessible (if the proxies are hard for an adversary to find, then they are also hard for ordinary users to find) and the darknet might still be infiltrated and enumerated by an active adversary. A cat and mouse game is played between circumventors and their adversaries, and this has pushed the circumventors to hide their services on more diverse and dynamic IP addresses, such as home computers that have low bandwidth, availability, and reliability.

Taken to its logical extreme, an adversary can be stymied if proxy services are so ubiquitous across the IP address space that the adversary would be forced to blacklist the majority of IP addresses, effectively disconnecting its clients entirely from the Internet. Ubiquitous proxy deployment seems infeasible because it would require every host to transit traffic for strangers, putting them at risk to exploitation. It is possible, however, to make any (and every) host on the Internet look and act like a latent proxy server — from the adversary's perspective — without that host running any proxy software. This is the notion behind *decoy routing*, a network system designed to make every destination IP address on the Internet act like a proxy server.

Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

1.1 Threat Model and Goal

Our goal is to provide a client within a filtered network access to blacklisted sites located outside of the adversary's network at near-normal latency and throughput while preserving the secrecy of the contents and true destination of the client communication.

The specific threat addressed by this paper is a simple IP firewall, capable of blocking all traffic to and from any address in a blacklist. It is assumed that our method of circumvention is publicly known and that the circumvention tool implementation, specification, and algorithms are freely available and open source. We also assume that the adversary can normalize IP traffic, search for regular expressions in traffic, and alter TCP/IP packet headers (such as for a NAT). Finally, we assume that a client can obtain the client software and a symmetric key of the type described in this paper.

1.2 Overview of Decoy Routing

The fundamental insight of decoy routing is that, while hosts can easily be filtered by IP address, it is difficult to filter a well-placed router or other transit device in the network. The current architecture of the Internet has three key properties that we leverage: first, routers are difficult to filter at the IP level because IP packets do not contain router addresses; second, IP routing is federated and therefore a network has little control over the upstream paths that their packets take; and finally, a well-placed router might lie on the path between a client and many millions of destination addresses. Filtering all paths that include a particular router is intensive (requires traceroutes to many prefixes) and error-prone (some routers filter traceroutes). Further, the router's paths might substantially affect the adversary's reachability to large fractions of the Internet.

Since it is difficult to filter routers and a single router can stand between a client and many destinations, a router is an ideal place for a circumvention proxy service. We propose that a router could double as (or provide access to) a proxy server. Such a modified router is called a *decoy router*. The proxy to which it connects is called a *decoy proxy*. In order for a client to connect to the decoy proxy, it cannot simply address packets to the router or the decoy proxy, since these these would easily be filtered. Instead, the client connects to *any destination* (called a *decoy destination*) that includes a decoy router on the path. Once connected, the client covertly signals over the TCP/IP flow that the decoy router should reroute the flow to a decoy proxy. The decoy proxy then hijacks the TCP connection and acts as a standard proxy server for the client.

An overview is provided in Figure 1. In it, a client

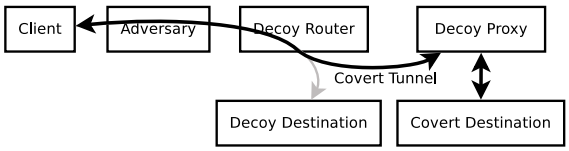

Figure 1: Decoy Routing

that wishes to reach a blacklisted website (covert destination) addresses its packets to an innocuous decoy destination, such as `http://www.unm.edu`. After signaling to the decoy router that its packets should be proxied, the client's TCP flow is forwarded to (and hijacked by) the decoy proxy. A tunnel is formed between the client and decoy proxy, and the proxy talks to the covert destination (such as `http://www.youtube.com`) on the client's behalf.

Decoy routing nullifies IP address filtering's ability to block communication to a destination because the IP address in a decoy routed packet is essentially meaningless. The address could be any destination with a decoy router on the path. Decoy routers can provide a large number of decoy addresses (depending on placement), essentially making circumvention proxies look ubiquitously deployed from the adversary's perspective.

The primary components of a decoy routing system are the client software, decoy router, decoy proxy, and covert signaling mechanism. The decoy router must be able to maintain line-rate communication while still searching for signals from clients. The decoy proxy must be able to hijack a client's TCP session in order to communicate with it. Finally, a covert signal must be developed to allow the client to communicate with the decoy router and decoy proxy to initiate the proxy session.

2 Related Work

There has been sustained interest in this decade in both the research and development of circumvention tools. The simplest tools provide proxy services or tunnels in unfiltered locations. Examples include Proxify [7], Psiphon [12], Global Pass [4], SSH tunnels, Corporate VPNs, Freegate [3], Ultrasurf [8], and Guardster [5]. Since the proxy and tunnel hosts are easily discovered and blocked, some services cycle the hosts through a range of IP addresses to avoid blacklists. The effectiveness of this evasion technique depends upon the number of available IP addresses and their distribution across IP address space.

Researchers have also developed means of covertly burying traffic inside of communication channels meant for other types of communication (such as web servers, images, and video files) via stegonagraphy or similar

methods. These are particularly useful when encrypted traffic might be considered suspect or is banned. Some examples of covert channels include TCP-over-DNS [1], Collage [10], Infranet [14], and CovertFS [9]. Such techniques are complementary to decoy routing, as they could be used for covert signaling between the client and the decoy router as is discussed in Section 5.

Both Coral [15] and Freenet [11] store data in peer-to-peer networks and make it available to end users. Users can find their data at multiple locations, some of which might not be blocked (although Coral is accessed through a domain that is easily blocked). Content distribution networks are often optimized for specific applications (such as web browsers), whereas decoy routing is intended for any IP application.

Freenet was specifically designed for address blocking circumvention and offers a darknet feature in which members of the network distribute their IP addresses to one another privately. Darknets are harder to block since the IP addresses are secret, but they are also harder to join unless you know somebody willing to admit that they use the network and willing to share data with you. The success of a darknet depends on the addresses being hard for the adversary to find but easy for the friends to share, and this is a difficult combination.

One of the most popular anonymization tools, Tor [13], is sometimes also used in an attempt to avoid IP address blocking. Tor works by forwarding encrypted packets between a randomly selected set of routers, making it difficult for the destination to discover the source of received packets. Tor nodes function like proxies—and suffer from the same drawbacks. In an attempt to thwart destination IP address blocking, Tor has created a network of private routers (called bridges) that they try to hide from the adversary by requiring users to send an email to get a partial list of private routers. However, this private network can be enumerated (and then blocked) by the adversary. Our work could enhance Tor and other anonymity services by providing access to their otherwise blocked servers.

The most similar work to ours, Telex [22], which was developed in parallel to this paper, also modifies the network infrastructure to support circumvention in a method similar to decoy routing. Telex provides a web-proxy service to the clients and the sentinel is generated using asymmetric cryptography.

3 Architecture

Figure 2 illustrates the basic operations of decoy routing: the client software connects to a decoy destination with a decoy router on the path, the client covertly signals to the decoy router that the flow should be hijacked, the decoy proxy hijacks the flow, and the decoy proxy acts as a standard SOCKS proxy.

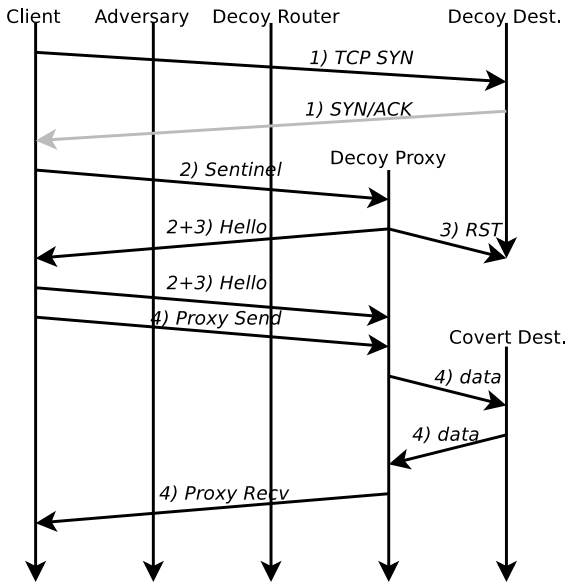

Figure 2: Decoy Routing Timeline
1) The client connects to the decoy destination. 2) The client sends a sentinel (covert signal) to suggest that the flow should be hijacked. 3) The hijack occurs. 4) The decoy proxy functions as a standard SOCKSv5 proxy.

3.1 Connecting to a Decoy Destination

The first step of decoy routing is that the client software must connect to a destination via a path that contains a decoy router. Assuming that some fraction of paths include a decoy router, the client software simply needs to probe various destinations until it finds one.

In order to probe for a decoy router, the client software first creates a normal TCP/IP connection. Once connected, the client inserts a sentinel value (perhaps a well-known string) into a covert channel. For instance, if the client connects to the destination on an HTTP port, then the sentinel should be placed covertly within an HTTP message, such as in a cookie or query string. If the client connects to a destination on an HTTPS port, then the sentinel could be placed within the random number field in the TLS client hello message. We use the latter method in our prototype.

The decoy router will look for sentinels and forward flows that contain them to a decoy proxy. Once the decoy proxy determines that the flow should be hijacked (as described in Section 3.2) it will send a Hello message to the client software. It is from this message that the client software discovers that a decoy router is on the path. If no such message is discovered, the connection is closed

and the client software continues its hunt.

3.2 Covertly Requesting Decoy Routing

There are many ways that the client software could covertly signal to the decoy router that it would like its flow to be hijacked. It could happen out-of-band, the signal could be a string generated by a symmetric key and placed in a packet payload, it could be the ports used for consecutive flows (port knocking [16]), or the signal could be hidden within a series of payload lengths. The key characteristics of the signal are that it must be possible for the decoy router to detect the request at line rate, and that the request must be difficult for an adversary to detect.

In this paper we describe a sentinel embedded within the TLS handshake. We assume that each client shares a secret key with the decoy proxy. This key is used to generate a set of time-varying nonces (or *sentinels*). To generate time-varying sentinels we apply an HMAC to the concatenation of the key, the current hour, and a per-hour sequence number. The sentinels may be used as part of the client random number field in the TLS Client Hello Messages sent by the client [20]. This field is in a fixed position and TLS client hello messages can be quickly identified by a router.

Once a decoy router detects a packet that contains a sentinel at the correct offset it forwards it and the remaining packets in the flow to a decoy proxy. It is the decoy proxy's job to finish the covert handshake with the client software and to hijack the TCP flow. First, the decoy proxy will wait for the client software to finish its TLS handshake with the decoy destination so that all plaintext communication (such as certificate exchange) occurs with the decoy and the adversary will see nothing out the ordinary. When the first encrypted TLS data record is detected, the decoy proxy hijacks the connection (as described in Section 3.3) and sends the client a Hello message to continue the covert handshake.

The purpose of the Hello message is to inform the client software that a decoy proxy has hijacked the flow. Since the decoy proxy does not know the TLS session key between the client and decoy destination, the decoy proxy encrypts the Hello message with the same key that the client used to create its sentinel. The resulting encrypted data should be shaped to look like the underlying TLS channel. The message itself includes the sentinel used by the client, and is signed by the decoy proxy's private key. The client software includes the decoy proxy's public key for verification. The client software includes a modified TLS library that detects the change in keys, converts the session to the new key, and responds with a similar Hello message.

At this point the client and decoy proxy can communicate using TLS over a TCP connection that looks to the adversary to be a TLS session to the decoy destination.

3.3 Hijacking a TCP Flow

Once the decoy proxy has decided to hijack a flow, it first forges a TCP RST packet to the decoy destination using the sequence number of the packet that triggered the hijack. This will cause the decoy destination to silently close its connection.

In order to hijack a TCP/IP flow the decoy proxy needs to know the proper TCP sequence number for the session. We assume asymmetric routing and therefore do not assume that return traffic is visible to the decoy router. Therefore all flow information must be gleaned from packets originated by the client. So long as there are not any packets in flight from the decoy destination to the client, the expected sequence number can be extracted from ACK values from packets from the client to decoy destination. In protocols where the server awaits a message from the client, such as the first HTTP GET request in a flow, we can be reasonably sure that there are no missed packets in flight.

A TCP hijack is harder for an adversary to detect if the hijacker uses the same TCP options as the decoy destination. Some options can be ascertained by looking at the current packet (such as TCP Timestamp) while other options (such as window scale and SACK) are determined during the TCP handshake. The options can be conveyed to the decoy proxy by the client during the covert signaling by inserting them, in an encrypted form, alongside the sentinel in the TLS client 28-byte random field. The client can discover the options used by its own network stack by watching the initial TCP handshake.

3.4 Providing Proxy Services

Once the flow is hijacked and the client and decoy proxy can communicate over TLS with an established key, the session can be used for any purpose. In our architecture, the session is used to communicate with a standard SOCKSv5 proxy. The client's applications (such as Firefox, Jabber, etc.) use SOCKS to connect to the client software. The client software tunnels the SOCKS requests to the decoy proxy, which in turn passes the stream to an unmodified SOCKS server. In order to tunnel multiple SOCKS streams over a single TCP flow, we use a simple protocol to multiplex multiple SOCKS streams between the client and decoy proxy.

The client to decoy proxy tunnel could be used for protocols other than SOCKS. For instance, IP packets from the client could be captured and sent to the decoy proxy to mimic a VPN. In this architecture we have chosen

SOCKS because of its simplicity and common adoption by applications.

4 Vulnerabilities

In comparison with simple firewall rules and IP blacklisting, decoy routing significantly raises the cost and effort required to block traffic, but decoy routing can be defeated. In this section we address methods by which clients using decoy routing may be detected by an adversary and aspects of the architecture that offer an opportunity for attack. We also describe how the architecture could be extended to mitigate the risk of such attacks.

4.1 Detection of Decoy Routing

If the adversary can detect that a client is using decoy routing, then the adversary can deny future network access to that client. The adversary may be able to detect decoy routing via passive analysis or by careful manipulation of the network to defeat the establishment of the covert channel.

For instance, if the hijacked path has different latencies, path lengths, or path MTU than the client to decoy destination path, then analysis may reveal that the flow is suspicious. The decoy router can delay packets and adjust the TTL and MTU of the packets it creates in order to attenuate the risk of detection.

A more active adversary can replay or preplay sentinels. If the adversary observes a TLS client hello, it can replay the hello message to the same destination to see whether the response is the expected response or appears to be from a decoy proxy. If the session fails in an unexpected manner but the TCP connection is not closed or reset, then the adversary may suspect that the client is using decoy routing.

An approach to defeating a replay probe is to allow each sentinel to be used only once, and poison sentinels immediately after they are detected the first time. Poisoned sentinels can be defeated with a preplay attack in which the adversary intercepts and delays each client hello message for a moment while it sends a copy of the hello message to the decoy destination. The client will never be able to establish a connection to a decoy proxy, although ordinary communication with decoy destinations via TLS will succeed as if the decoy routers had vanished. To make matters worse, the adversary could detect the response from the decoy proxy and use this to identify clients using decoy routing.

Note that this problem is specific to the TLS sentinel. Future sentinels (for instance those that use port knocking) may not be susceptible to preplay attacks.

4.2 Denial of Access to Decoy Routing

If the adversary discovers the location of a decoy router, it can start a DoS attack on the router. The router must be able to handle packets at line rate and not be the first network component to fail in a DoS attack.

A slightly more sophisticated adversarial attack would be to fragment all packets to a small size (particularly if the first fragment of a packet does not contain the complete TCP header). This would prevent the decoy router from assigning packets to TCP flows for redirection. If the decoy router had to perform packet reassembly in order to reconstruct flows, the amount of state it would need to maintain would grow enormously. (This problem is not unique to decoy routing and is also an issue with NAT.)

An adversary could also thwart decoy routing by routing a flow's packets over different paths, such that the decoy router only observes some of the packets. However, round-robin forwarding is difficult for an adversary to accomplish since BGP selects a single best path and round-robin forwarding causes significant TCP performance issues.

Finally, if the adversary compromises a decoy routing user's machine, then it could discover the user's key. The compromised key could be used to create several decoy destinations and overwhelm the decoy proxy. This is similar to a DoS attack on the decoy router, but focuses on the decoy proxy instead, which might not be able to handle as high a load as the decoy router. This attack could be mitigated by rate limiting connections from a single key, or even revoking keys from abusers.

5 Discussion

Implementation A proof-of-concept system is in active development. The decoy router is developed with Click [18] and the client's TLS changes are implemented in OpenSSL. The remainder of the system is written in Python. All major components are functional and have been shown to work on the DETER [17] testbed. In the future we intend to add more covert channels, improve the efficiency and scalability of the system, and address any existing vulnerabilities.

Decoy Router Placement In order to intercept flows intended for a decoy router, the router needs to be on the path of a decoy destination. Identifying good locations for decoy routers is challenging for several reasons. The set of decoy routers must lie between the client and a target fraction of destinations. This fraction might be small (although large is better), but without complete topology information it is difficult to obtain accurate estimates of coverage. Second, decoy routers identify and redirect

packets at line speed, which may place restrictions on the link speeds of the redirection locations. Finally, location of the decoy routers can impact end-to-end latency.

Passive Decoy Routers This paper described active redirection, where packets of interest are extracted from the network, edited, and re-inserted back into the network so they head to the covert destination, not to the decoy destination. We are also investigating methods for passive redirection, where the traffic to the decoy destination is not removed from the network.

Covert Channel Bonding We are investigating algorithms and techniques to mix multiple covert channels together to support higher bandwidth client applications that are prohibitive in a single covert channel. We anticipate our solution will permit the covert channels to not only have distinct decoy destinations and span multiple decoy routers but also respect the constraints imposed by the individual covert channels to ensure that the channel characteristics remain realistic.

6 Conclusion

As adversaries develop more sophisticated techniques for filtering, tampering with, and monitoring network traffic, clients will require more sophisticated tools to access the Internet in a secure and unhindered manner. Existing circumvention tools rely on proxies with fixed IP addresses that are trivial to filter with a simple IP firewall and blacklist. We have introduced an architecture for decoy routing, a technique that nullifies IP firewalls by breaking the relationship between the apparent IP and true destination IP addresses of packets. We have shown that blocking a decoy router could have significant unintended consequences, making it difficult to block decoy routers without disrupting ordinary network traffic. We have developed a proof-of-concept prototype for this mechanism that demonstrates that TCP hijacking by a decoy proxy is practical and significantly raises the bar for the sophistication needed by an adversary when compared to contemporary techniques.

7 Acknowledgments

The authors would like to thank Professor Jennifer Rexford of Princeton University and Professor Nick Feamster of the Georgia Institute of Technology for their comments, insights, and critiques of decoy routing.

This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Navy SPAWAR under Contract N66001-11-C-4017. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

[1] Analogbit: Tcp-over-dns tunnel software howto. `http://analogbit.com/tcp-over-dns_howto`.

[2] Anonymizer.com.

[3] Freegate. `http://www.dit-inc.us/freegate`.

[4] Global pass. `http://gpass1.com/gpass/`.

[5] Guardster. `http://www.guardster.com`.

[6] Opennet initiative (ONI). `http://opennet.net/research/profiles`.

[7] Proxify web proxy. `https://proxify.com`.

[8] Ultrasurf. `http://www.ultrareach.com`.

[9] BALIGA, A., KILIAN, J., AND IFTODE, L. A web based covert file system. *Proceedings of the 11th USENIX workshop on Hot topics in operating systems HOTOS* (2007).

[10] BURNETT, S., FEAMSTER, N., AND VEMPALA, S. Chipping away at censorship with user-generated content. *USENIX Security Symposium* (2010).

[11] CLARKE, I. A distributed decentralised information storage and retrieval system. Master's thesis, University of Edingburgh, 1999.

[12] DEIBERT, R. Psiphon. `http://psiphon.civisec.org/`.

[13] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. *13th USENIX Security Symposium* (2004).

[14] FEAMSTER, N., BALAZINSKA, M., AND HARFST, G. Infranet: Circumventing web censorship and surveillance. *Proceedings of the 11th USENIX Security Symposium* (2002).

[15] FREEDMAN, M. J., FREUDENTHAL, E., AND MAZIRES, D. Democratizing content publication with coral. *Network Systems Design and Implementation* (2004).

[16] KRZYWINSKI, M. Port knocking: Network authentication across closed ports. *SysAdmin Magazine 12* (2003), 12–17.

[17] MIRKOVIC, J., BENZEL, T., FABER, T., BRADEN, R., WROCLAWSKI, J., AND SCHWAB, S. The DETER project: Advancing the science of cyber security experimentation and test. In *2010 IEEE HST* (2010), pp. 1 –7.

[18] MORRIS, R., KOHLER, E., JANNOTTI, J., AND KAASHOEK, M. F. The click modular router. *SIGOPS Oper. Syst. Rev. 33* (December 1999), 217–231.

[19] NEWS, B. Tehran blocks access to facebook. BBC News`http://news.bbc.co.uk/2/hi/8065578.stm`, May 2009.

[20] SALOWEY, J., ZHOU, H., ERONEN, P., AND TSCHOFENIG, H. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 4507 (Proposed Standard), May 2006. Obsoleted by RFC 5077.

[21] Tor partially blocked in China. Blog post at `https://blog.torproject.org/blog/tor-partially-blocked-china`.

[22] WUSTROW, E., WOLCHOK, S., GOLDBERG, I., AND HALDERMAN, J. A. Telex: Anticensorship in the network infrastructure. *To appear in the proceedings of the 20th USENIX Security Symposium* (2011).