David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. "Hiding Routing Information," Workshop on Information Hiding, Cambridge, UK, May, 1996.

Hiding Routing Information

David M. Goldschlag, Michael G. Reed, and Paul F. Syverson

Naval Research Laboratory, Center For High Assurance Computer Systems, Washington, D.C. 20375-5337, USA, phone: +1 202.404.2389, fax: +1 202.404.7942, e-mail: {*last name*}@itd.nrl.navy.mil.

Abstract. This paper describes an architecture, *Onion Routing*, that limits a network's vulnerability to traffic analysis. The architecture provides anonymous socket connections by means of proxy servers. It provides real-time, bi-directional, anonymous communication for any protocol that can be adapted to use a proxy service. Specifically, the architecture provides for bi-directional communication even though no-one but the initiator's proxy server knows anything but previous and next hops in the communication chain. This implies that neither the respondent nor his proxy server nor any external observer need know the identity of the initiator or his proxy server. A prototype of *Onion Routing* has been implemented. This prototype works with HTTP (World Wide Web) proxies. In addition, an analogous proxy for TELNET has been implemented. Proxies for FTP and SMTP are under development.

1 Introduction

This paper presents an architecture that limits a network's vulnerability to traffic analysis. We call this approach *Onion Routing*, because it relies upon a layered object to direct the construction of an anonymous, bi-directional, real-time virtual circuit between two communicating parties, an *initiator* and *responder*. Because individual *routing nodes* in each circuit only know the identities of adjacent nodes (as in [1]), and because the nodes further encrypt multiplexed virtual circuits, studying traffic patterns does not yield much information about the paths of messages. This makes it difficult to use traffic analysis to determine who is communicating with whom.

Onion Routing provides an anonymous socket connection through a proxy server. Since proxies are a well defined interface at the application layer [12, 11], and many protocols have been adapted to work with proxy servers in order to accommodate firewalls, Onion Routing can be easily used by many applications. Our prototype works with HTTP (World Wide Web) proxies. In addition, a proxy for TELNET has been implemented.

Traffic analysis can be used to help deduce who is communicating with whom by analyzing traffic patterns instead of the data that is sent. For example, in most networks, it is relatively easy to determine which pairs of machines are communicating by watching the routing information that is part of each packet. Even if data is encrypted, routing information is still sent in the clear because routers need to know packets' destinations, in order to route them in the right

direction. Traffic analysis can also be done by watching particular data move through a network, by matching amounts of data, or by examining coincidences, such as connections opening and closing at about the same time.

Onion Routing hides routing information by making a data stream follow a path through several nodes en route to its destination. The path is defined by the first node, which is also a proxy for the service being requested (e.g., HTTP requests). Therefore, this Proxy/Routing Node is the most sensitive one, so sites that are concerned about traffic analysis should also manage a Proxy/Routing Node. We will see later that it is important that this Proxy/Routing Node also be used as an intermediate routing node in other virtual circuits. Although the compromise of all routing nodes compromises the hiding, one uncompromised routing node is sufficient to complicate traffic analysis. Figure 1 illustrates the topology of an Onion Routing network with five nodes, one of which (W) is the Proxy/Routing node for the initiator's site.

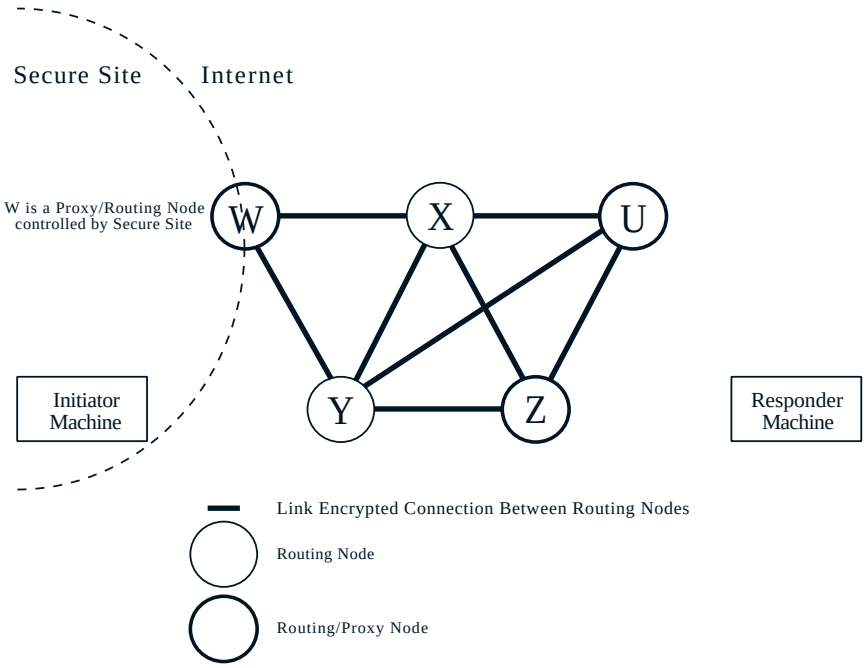

Fig. 1. Routing Topology.

The goal of Onion Routing is not to provide anonymous communication. Parties are free to (and usually should) identify themselves within a message. But the use of a public network should not automatically give away the identities and locations of the communicating parties. For example, imagine a researcher who uses the World Wide Web to collect data from a variety of sources. Although each

piece of information that he retrieves is publicly known, it may be possible for an outside observer to determine his sensitive interests by studying the patterns in his requests. Onion Routing makes it very difficult to match his HTTP requests to his site.

Anonymous re-mailers [5, 6] attempt to limit the feasibility of traffic analysis by providing an anonymous store and forward architecture. To prevent replay attacks, re-mailers keep a log of sent messages. These two characteristics make the anonymous re-mailer approach unsuitable for HTTP applications, as HTTP requests would both generate an enormous log and require bi-directional communication. Anonymous ISDN [8] has even more severe real-time and bi-directional requirements than HTTP, but, the architecture of an ISDN network is considerably different from the architecture of the Internet [4].

Onion Routing provides bi-directional communication, without requiring that the responder know the initiator's identity or location. Individual messages are not logged. In addition, Onion Routing is easily adapted to electronic mail. Messages can include *Reply Onions* that permit a later reply to the sender without knowing his address and without keeping the original virtual circuit open.

The rest of the paper is organized in the following way: Section 2 presents background information. Section 3 describes the *Onion*, the object that directs the construction of the virtual circuit. Section 4 describes the construction and use of these virtual circuits. Section 5 describes the vulnerabilities in the Onion Routing architecture. Section 6 presents some concluding remarks.

2 Background

Chaum [1] defines a layered object that routes data through intermediate nodes, called *mixes*. These intermediate nodes may reorder, delay, and pad traffic to complicate traffic analysis. Some work has been done using mixes in ATM networks [3].

Anonymous Remailers like [5, 6] use mixes to provide anonymous e-mail services and also to invent an address through which mail can be forwarded back to the original sender. Remailers work in a store and forward manner at the mail application layer, by stripping off headers at each mix, and forwarding the mail message to the next mix. These remailers provide confirmation of delivery.

In [8], mixes are used to provide untraceable communication in an ISDN network. In a phone system, each telephone line is assigned to a particular local switch (i.e., local exchange), and switches are interconnected by a (long distance) network. Anonymous calls in ISDN rely upon an anonymous connection within each switch between the caller and the long distance network, which is obtained by routing calls through a predefined series of mixes. The long distance endpoints of the connection are then mated to complete the call. (Notice that observers can tell which local switches are connected.) This approach relies upon two unique features of ISDN switches. Since each phone line has a subset of the switch's total capacity pre-allocated to it, there is no (real) cost associated with keeping

a phone line active all the time, either by making calls to itself, to other phone lines on the same switch, or to the long distance network. Keeping phone lines active complicates traffic analysis because an observer cannot track coincidences.

Also, since each phone line has a control circuit connection to the switch, the switch can broadcast messages to each line using these control circuits. So, within a switch a truly anonymous connection can be established: A phone line makes an anonymous connection to some mix. That mix broadcasts a token identifying itself and the connection. A recipient of that token can make another anonymous connection to the specified mix, which mates the two connections to complete the call.

Our goal of anonymous socket connections over the Internet differs from anonymous remailers and anonymous ISDN. The data is different, with real-time constraints more severe than mail, but somewhat looser than voice. Both HTTP and ISDN connections are bidirectional, but, unlike ISDN, HTTP connections are likely to be small requests followed by short bursts of returned data. In a local switch capacity is pre-allocated to each phone line, and broadcasting is efficient. But broadcasting over the Internet is not free, and defining broadcasts domains is not trivial. Most importantly, the network topology of the Internet is more akin to the network topology of the long distance network between switches, where capacity is a shared resource. In anonymous ISDN, the mixes hide communication within the local switch, but connections between switches are not hidden. This implies that all calls between two businesses, each large enough to use an entire switch, reveal which businesses are communicating. In Onion Routing, mixing is dispersed throughout the Internet, which improves hiding.

3 Onions

To begin a session between an initiator and a responder, the initiator's proxy identifies a series of routing nodes forming a route through the network and constructs an *onion* which encapsulates that route. Figure 2 illustrates an onion constructed by the initiator's Proxy/Routing Node W for an anonymous route to the responder's Proxy/Routing Node Z through intermediate routing nodes X and Y. The initiator's proxy then sends the onion along that route to establish a virtual circuit between himself and the responder's proxy.

The onion data structure is composed of layer upon layer of encryption wrapped around a payload. Leaving aside the shape of the payload at the very center, the basic structure of the onion is based on the route to the responder that is chosen by the initiator's proxy. Based on this route, the initiator's proxy encrypts first for the responder's proxy, then for the preceding node on the route, and so on back to the first routing node to whom he will send the onion. When the onion is received, each node knows who sent him the onion and to whom he should pass the onion. But, he knows nothing about the other nodes, nor about how many there are in the chain or his place in it (unless he is last). What a

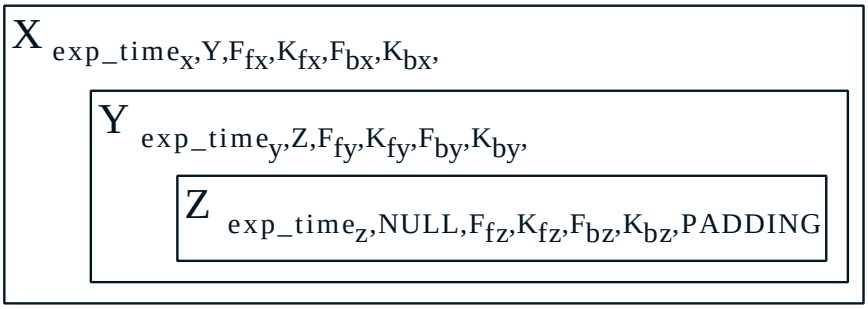

Fig. 2. A Forward Onion.

node P_x receives looks like this

$$\{exp\_time, next\_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$$

Here PK_x is a public encryption key for routing node P_x, who is assumed to have the corresponding decryption key.[1] The decrypted message contains an expiration time for the onion, the next routing node to which the payload is to be sent, the payload, and two function/key pairs specifying the cryptographic operations and keys to be applied to data that will be sent along the virtual circuit. The forward pair (F_f, K_f) is applied to data moving in the forward direction (along the route that the onion is traveling) the backward pair (F_b, K_b) is applied to data moving in the opposite direction (along the onion's reverse route).[2] (If the receiving node is the responder's proxy, then the *next_hop* field is *null*.) For any intermediate routing node the payload will be another onion. The expiration time is used to detect replays, which pairs of compromised nodes could use to try to correlate messages. Each node holds a copy of the onion until *exp_time*. If he receives another copy of the same onion within that time he simply ignores it. And, if he receives an onion that has expired, he ignores that as well.

Notice that at each hop the onion shrinks as a layer is peeled off. To avoid compromised nodes inferring route information from this monotonically diminishing size, a random bit string the size of the peeled off layer is appended to the end of the *payload* before forwarding. No proxy except the last will know how much of the *payload* he receives is such padding because he won't know where

[1] Depending on certain assumptions about the fields in each onion layer, a naive RSA implementation of the simple public key encryption implied by our notation could be vulnerable to an attack as described in [7]. In our implementation, this potential vulnerability is illusory since the public key is only used to encrypt a secret key, and that secret key is used to encrypt the remainder of the message using an efficient symmetric algorithm. This also makes for a more efficient implementation than the simple, straightforward implementation using only public keys.

[2] Specifying two pairs of functions unifies the virtual circuits that are constructed by forward and reply onions. See section 3.3.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.