Computer Communications 35 (2012) 829–841

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Ferris wheel: A ring based onion circuit for hidden services

Hakem Beitollahi *, Geert Deconinck

Katholieke Universiteit Leuven, Electrical Engineering Department, Kasteelpark Arenberg 10, Leuven, Belgium

ARTICLE INFO

Article history:
Received 28 July 2011
Received in revised form 10 January 2012
Accepted 11 January 2012
Available online 20 January 2012

Keywords:
Hidden services
Location hiding
Anonymity
Traffic analysis attack
Privacy

ABSTRACT

The capability that a server can hide its location while offering various kinds of services to its clients is called hidden services or location-hiding. Almost previous low-latency anonymous communication systems such as Tor, MorphMix, etc., that can be used to implement hidden services are vulnerable against end-to-end traffic analysis attack. This paper introduces Ferris wheel, a novel architecture for implementing hidden services which is robust against end-to-end traffic analysis attack. Moreover, our scheme is more robust against various traffic analysis attacks than previous low-latency anonymous communication architectures.

1. Introduction

The capability that a server node can hide its location (IP address) while offering various kinds of services (e.g., web pages) to clients is called hidden services or location-hiding. Hidden services were introduced to resist distributed DoS attacks since these attacks depend on the knowledge of their victim's IP addresses [1]. *The idea* is that a victim server distributes its services (e.g., web pages) among multiple overlay nodes such that any overlay node takes a fraction of the services. Then any overlay node that hosts the services secretly gives the services to the clients through the Ferris wheel architecture which we discuss it in this paper. An overlay node that hosts a fraction of the services of the victim server is called a proxy server. We call it, the hidden server throughout this paper. The second advantages of a hidden server is that a server that is accessible but hidden can resist a variety of threats (both physical and logical) simply because it cannot be found. Location-hiding has also been recommended for preserving the anonymity of the services which need to resist censorship such as for dissidents or journalists publishing information accessible from anywhere [2]. Again, the idea is that the pages that resist censorship are scattered among several overlay nodes and then overlay nodes anonymously provides services to clients through the Ferris wheel architecture.

Most activities in anonymous communications offer sender and relationship anonymity (see Section 2). In fact, the complementary problem, hiding user identity, has been well studied since the early 1980s. However, recent years have provided little literature on hidden services such as Tor [3]. In fact, although almost low-latency anonymous communication systems such as MorphMix [4], Tarzan [5], Freedom [6], Web MIXes [7] can be used to enable hidden services through the rendezvous point protocol, only Tor [3] has deployed hidden services. Tor deployed hidden services in early 2004. When Tor deployed these services, it claimed that it is strongly robust against non-global adversaries. After that time, several attacks [8–11] have been demonstrated against Tor and have challenged the claim of Tor. In Tor, the security of the hidden service is only as strong as the position of the last node (exit node) is compromised in the circuit. In fact, if an adversary could monitor traffic of the exit node, then it can find the location of the hidden service immediately. This is not only the problem of Tor, but the problem of all other anonymous communication systems [12,7,5,4] that are based on linear onion circuits.

A linear (serial) onion circuit is a circuit that starts from an entry node (entry onion router), continues through some cascade middle onion routers and closures in an exit node (exit onion router). The entry and exit nodes are called endpoints in these circuits (Fig. 1). The clients' traffic enters the circuit at the entry node, routes through middle onion routers and finally reaches the exit node. The exit node delivers traffic to the destination (e.g., the hidden server). Tor [3], MorphMix [4], Tarzan [5], Web MIXes [7], Freedom [6] are examples of linear onion circuits.

Previous works [10,13] show that, upon compromising (e.g., hacking or eavesdropping) the entry and exit points of a linear circuit, it is possible to compromise the anonymity of a connection via traffic analysis. This is well-known as an end-to-end traffic analysis

* Corresponding author. Tel.: +32 16321020.
E-mail addresses: Hakem.Beitollahi@esat.kuleuven.be (H. Beitollahi), Geert.Deconinck@esat.kuleuven.be (G. Deconinck).

0140-3664/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2012.01.008

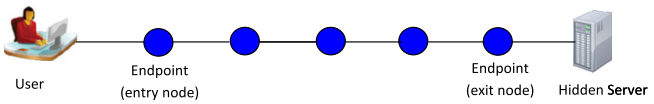

Fig. 1. A linear onion circuit.

attack; thereby linear onion circuits are vulnerable against end-to-end traffic analysis attack. Normally, in all attack models we assume that attackers have control over a fraction of nodes; thereby it is possible that some malicious nodes participate in the circuit. The goal of attackers is to locate their malicious nodes in the endpoint positions, the entry and exit nodes, to compromise the anonymity of the system. When attackers realize that their malicious nodes that have participated in the circuit are unable to compromise the anonymity of a connection, meaning that malicious nodes are not in entry and exit positions, they break the circuit simply by dropping all traffic and force the initiator (hidden server or user) to rebuild a new circuit. They use this technique repeatedly with the hope that the new circuit will contain the malicious nodes as both entry and exit points. Now the question is: can anything be done to make low-latency anonymous communication systems resist end-to-end traffic analysis attack?

In this paper, we introduce a novel architecture to enable hidden services that withstands end-to-end traffic analysis attack. The Ferris wheel architecture is a ring-based onion circuit that lacks any exit node; thereby spontaneously is robust against end-to-end traffic analysis attacks. The hidden server constructs a ring of onion routers, including itself; i.e., the hidden server also acts as an onion router and it is a part of the ring. The entry node of the ring periodically generates a constant number of dummy Query and Response packets and circulates them in the ring in the anticlockwise and clockwise directions, respectively. Any node of the ring encrypts and decrypts the anticlockwise and clockwise traffic respectively. Clients' query packets enter the ring through the entry node of the ring. The entry node of the ring replaces sufficient number of dummy Query packets by clients' query packets and rotates them in the anticlockwise direction. All peers of the ring see traffic, but only the hidden server (HS) can understand the traffic because only HS can decrypts all layers of encryption. When HS wishes to reply to a client, it replaces sufficient number of dummy Response packets by its response packets to the client. When clockwise packets reach the entry node of the ring, then the entry node of the ring delivers traffic to the client. HS inappreciably controls flows of the query and response packets through fingerprint test; hence, it can easily detect malicious behavior in the ring.

The Ferris wheel architecture has three major properties: (1) lack of exit node that indicates the system withstands end-to-end traffic analysis attack; (2) node homogeneity that indicates none of nodes of a ring are distinguishable from each other; (3) traffic homogeneity that indicates clients' query packets are not distinguishable from dummy query packets and HS's response packets to clients are not distinguishable from dummy Response packets.

The next preference that the Ferris wheel architecture has over linear onion circuits is that in linear based circuits, position of ORs is important. The ORs closer to endpoints are more valuable for attackers than those ORs that are farther from endpoints, because through them and via iterated attacks [14,3], the attackers can discover the IP address of endpoints and thereby try to compromise entry and exit points. However, In the Ferris wheel architecture unlike the linear circuits, the position of ORs is not important because in a ring all nodes have the same position. The Ferris wheel architecture has some more preferences over linear onion circuits which we discuss them in Section 7.

Our architecture looks like a Ferris wheel, because suppose there is a Ferris wheel that all people (regardless of their sex) sat on the Ferris wheel's seats have the same clothes, same color and same appearance (node homogeneity). One of these people is the target (e.g., HS). The Ferris wheel is rotating and adversaries want to find the target person, but whom of them?

The rest of the paper is structured as follows: Section 2 reviews the related work. Section 3 describes the attack model. Section 4 presents the design of the Ferris wheel architecture. Section 5 discusses security analysis of Ferris wheel. Section 6 shows robustness of our architecture against traffic analysis attack. Section 7 discusses performance evaluation and finally. Section 8 concludes the paper.

2. Related work

Anonymous communication networks first time were introduced by David Chaum in 1981 [15]. He described a network that distributes trust across multiple nodes (aka mixes) that carry the communication. The design is of a public key based, high-latency anonymous communication network such as might be appropriate for emails, but it cannot be used for bidirectional, low-latency communications such as web-browsing, chat or remote login.

The first published, as well as the first deployed, distributed and circuit-based system for low-latency anonymous communications was onion routing [16] and then followed by MorphMix [4], Tarzan [5], Web MIXes [7], and Tor [3]. All of these architectures work by passing traffic through multiple onion routers that have composed a linear circuit. At each onion router (OR) the traffic changes its appearance by adding or removing a layer of encryption to/from the traffic, depending on whether it is traveling from the circuit initiator to responder or vice versa. Onion Routing roject directly uses public key cryptography for nested encryption while other architectures use public key cryptography only to distribute symmetric session keys to the nodes along a route, thus establishing a circuit. ORs use these session keys for nested encryption.

All the above architectures are fundamentally based on linear onion circuits but differ in some details such as implementation procedure, types of onion routers (overlay node vs. trusted mixes), utilizing the dummy traffic and guardian nodes. All the above architectures except Web MIXes [7] use overlay nodes as onion routers; while in the architecture of Web MIXes, few trusted well-known mixes are used as onion routers. The architectures that use overlay nodes as onion routers are more vulnerable to traffic analysis attack, while due to large number of overlay nodes are more resist to DDoS and host compromising attacks. The architectures that use trusted well-known mixes are more resist against traffic analysis attack, while more vulnerable against DDoS and host compromising attacks.

Although all the above architectures can be used to implement hidden services via rendezvous protocol, only Tor [3] supports hidden services. In Tor, any user to connect the hidden server must select its rendezvous point and informs the contact information of this node to the hidden server through the introduction points (introduction points are the nodes that listen to users' connections on the behalf of hidden server). Next, the hidden server constructs a linear onion circuit toward the rendezvous point; thereby the user can communicate with the hidden server while does not know the location of the server.

There is much literature on attacking anonymous communication architectures [17]. In fact, there is a little variety in the architectures that support anonymity especially for hidden services; while much literature in this field on attacking these architectures. Moreover, the only architecture we can find that is not linear in the fields of both anonymous communication and hidden services is Tree Based Circuits (TBCs) [11]. TBCs was proposed for thwarting