Onion Routers: a Dangerous Response to Traffic Analysis?

Dario Forte

Created to safeguard user privacy, the onion routing circuit might allow malicious hackers to cover their tracks.

The need: protection from traffic analysis

Traffic analysis is used, among other things, to identify the addresses that a given IP address seeks to contact. This technique may have various purposes, from simple statistical analysis to illegal interception. In response to this, researchers from the US Naval Research Laboratory conceived a system, dubbed 'onion routing', that eludes the above two operations.

Onion routing: what it is

The objective of onion routing is to make it completely impossible for third parties to perform traffic analysis. This goal is achieved by applying cryptographic techniques to networking. The packets transporting the chain of onion routers thus appear anonymous. Yes, we are talking about a chain. Practically speaking, there is a group of onion routers distributed around the public network, each of which has the task of encrypting the socket connections and to act in turn as a proxy. Experiments with onion routing have already been carried out on Sun Solaris 2.4 using proxies for http (WWW) and RLOGIN. At the moment, proxy operations are planned for email (SMTP), FTP and a selection of other protocols.

Let's imagine we have to make a http transaction. This is how it works:

- The application doesn't connect directly to the destination Web server, but rather to a socket connection with an onion routing proxy.
- The Onion Routing proxy establishes a direct anonymous connection with its nearest sister. To guarantee the impossibility of interceptions, the first onion routing proxy makes another connection with others of its kind to complete the chain. To avoid hijacking and man-in-the-middle phenomena, the communication between onion routers is forced. Practically speaking, each onion router is only able to identify and dialog with its adjacent kin included in the route. Each packet can currently make a maximum of 11 hops, then it has to reach its destination.
- Each time an onion router handles a transaction, it strips away a layer of encryption with respect to the preceding hop. This means that at the end of the route the packet arrives in clear-text.
- In addition, the encryption and transmission of data through the links of the chain is carried out randomly in such a way as to render impossible any sort of 'sequence prediction'. Furthermore, whenever the connection is interrupted, for any reason, all information relating to a given transaction is deleted from the rest of the chain. It's basically a sort of 'no cache' system.

It is also possible to use onion routing together with the Windows 95/NT NRaD redirector, acting at the TCP/IP network protocol stack level and forcing the connection routing through the onion routing network. The only practical limitation is that the NRaD redirector cannot be freely distributed because of licensing restrictions.

The differences with the other 'anonymizers'.

According to the official project documents (www.onion-router.net), onion routing differs from other anonymity services in three ways:

- Communication is real-time and bi-directional.
- The anonymous connections are application-independent (as opposed to services like anonymizer.com and its like)
- There is no centralized trusted component.

Applications may choose whether to identify their users over an anonymous connection. However, the use of a switched public network should not automatically reveal who is talking to whom. This is the traffic analysis that onion routing complicates.

The onion routing roadmap

The onion routing concept was introduced in early 1996. The basic idea achieved 'proof-of-concept' with the implementation of the 'Onion Router I' project comprising of five OR devices, wholly managed by the US Naval Research Laboratory. The project has recently undergone further developments and now includes 50 'core onion routers' comprising the second generation of the chain and having the characteristics of the hop randomization described above. The interesting aspect with respect to the first generation is that onion routing the Next Generation (OrtNG) has a series of added features, many of which constitute improvements of the cryptosystem with particular reference to transaction speed. This thus resolves the potential overheads characteristic of the earlier project, which were obviously performance limiting, even with the use of accelerators.

ORtNG can be currently divided into seven basic modules. Here are the details:

- Application Proxy (AP) — This is the application-specific proxy that handles interfacing into the onion routing network. It also contains a Database Engine (DB) since the AP now does route planning and onion creation (formerly done by the first Core (C); the trust for generating the onions has been moved closer to the user). The team is currently planning APs for HTTP/1.1-HTML/4.0, SMTP, FTP, RLOGIN, TELNET, NNTP, talk, finger, whois, gopher, WAIS, dns, nfs, RAW sockets, Virtual LANs, and SOCKS5.

- Input Funnel (IF) — This is an optional unit used to multiplex more APs into one Core, or to span a firewall without having to reveal the network topology on the secure side of the firewall. IFs can be stacked as deep as necessary (no limit) between the AP and the Core. Ultimately, IFs will be able to load-balance between multiple Cores.
- Database Engine (DB) — The DB is responsible for distributing and maintaining information about the entire network. It learns the public certificates for all nodes, the link state of the entire network graph, the exit access control policies for each node, and the current operational state of each node. This information is critical to creating an effective route through the network by the AP.
- Core (C) — The Core is the heart of onion routing. It moves cells along anonymous connections throughout the onion routing network. Currently it is the only element that contains a Chaum MIX, but other elements, e.g., IF, AP, or Output Funnel (OF), could also have them added.
- Crypto Processor (CP) — The CP is responsible for processing onions at each C. The CP performs the necessary public-key decryption and prepares the onion for the next hop, returning the result back to the C. This unit is critical to prevent processing 'burps' at Cs during costly public key operations.
- Output Funnel (OF) — The OF is responsible for de-multiplexing the circuits from the C to the Responder Proxies (RP). Since there are multiple types of RPs, the OF must peek initially into the stream to determine which RP is most appropriate for a new circuit.
- Responder Proxy (RP) — There are a number of different types of RPs which deal with different types of circuits:
  - Short Lived (RPSL) — Short lived connections are things like HTTP.
  - Long Lived (RPLL) — Long lived connection are things like RLOGIN or TELNET.
  - Reply Onion (RPRO) — Any connection utilizing a reply onion must route through here or else all crypto will fail for that circuit.
  - Virtual LAN (RPVL) — Specialized RP to handle VLANs.

The potential dangers of onion routers

While on the one hand onion routers mean that user privacy can be definitively protected, the adoption of these chaining systems represents a potential means of limiting backtracing. Here are the main reasons:

- Within the encryption done by the onion router another cryptographic operation may be encapsulated which is completely transparent to the former. This means a doubling of packet payload masking operations.
- At the moment, the system is able to generate Access Control Policies (ACP) regarding who can access the service and from what ingress, what types of protocols can be used, who manages the pertinent public key infrastructures, and so on. On this point let us remind you that there is no centralized body for administrating architectural design credentials.
- The following protocols and services are currently supported: HTTP, SMTP, FTP, RLOGIN, Telnet, NNTP, Talk, Finger, Whois, gopher, WAIS, DNS, NFS, VLANs, RAW connections (NRaD redirector), and SOCKS5. The designers do not exclude a rapid updating of the list, which is potentially limitless.

If the features so far described may seem marginal, there are significant problems in the realm of logging. Practically speaking, it is not clear who, in this period of cyber-terrorism threat, has to keep the transaction logs necessary for backtracing alleged attacks. The problem is that, as opposed to the project called 'onion router I', managed as we said by the US Navy, the second generation of onion routers can be independently managed by different groups and distributed anywhere in the world. Who handles the cryptography? How? Is it possible in all cases to get back to cleartext?

Personally I believe a two-pronged study should be carried out:

- Architecture: a possibility for monitoring and assessment should be integrated into the Database Engine via opportune policies including granular logging, at least for the connection. The objective is to be able to reconstruct, at any time and on a justified basis, the transaction of a pedophile or a cyber-terrorist. Regarding the Proxies, on the other hand, screening must be possible in order to inspect packets and eliminate those with a potentially damaging payload. I am personally thinking of a control at the proxy level aimed at isolating potential covert channels, notorious as tools for esoteric attacks. To achieve this the assessment could be delegated to a firewall positioned upstream of each node, or at least the entrance node.
- Legislation: The legal framework for this system is still in the works in spite of the fact that onion routers were implemented by the US military. At the moment the only thing outlawed is the use of the service by countries subject to embargo. This is the same line followed for cryptosystem exportation, a development of the ITAR and EAR legislation that has evolved over the last five years. We realize that privacy has to be safeguarded, but so do security and stability. We may still be in time.

[1] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Proxies for anonymous routing. In *12th Annual Computer Security Applications Conference*, 1996