I2P's Threat Model - I2P

30-38 minutes

This page was last updated in November 2010 and is accurate for router version 0.8.1.

What do we mean by "anonymous"?

Your level of anonymity can be described as "how hard it is for someone to find out information you don't want them to know?" - who you are, where you are located, who you communicate with, or even when you communicate. "Perfect" anonymity is not a useful concept here - software will not make you indistinguishable from people that don't use computers or who are not on the Internet. Instead, we are working to provide sufficient anonymity to meet the real needs of whomever we can - from those simply browsing websites, to those exchanging data, to those fearful of discovery by powerful organizations or states.

The question of whether I2P provides sufficient anonymity for your particular needs is a hard one, but this page will hopefully assist in answering that question by exploring how I2P operates under various attacks so that you may decide whether it meets your needs.

We welcome further research and analysis on I2P's resistance to the threats described below. More review of existing literature (much of it focused on Tor) and original work focused on I2P is needed.

Network Topology Summary

I2P builds off the ideas of many other systems, but a few key points should be kept in mind when reviewing related literature:

- **I2P is a free route mixnet** - the message creator explicitly defines the path that messages will be sent out (the outbound tunnel), and the message recipient explicitly defines the path that messages will be received on (the inbound tunnel).
- **I2P has no official entry and exit points** - all peers fully participate in the mix, and there are no network layer in- or out-proxies (however, at the application layer, a few proxies do exist)
- **I2P is fully distributed** - there are no central controls or authorities. One could modify some routers to operate mix cascades (building tunnels and giving out the keys necessary to control the forwarding at the tunnel endpoint) or directory based profiling and selection, all without breaking compatibility with the rest of the network, but doing so is of course not necessary (and may even harm one's anonymity).

We have documented plans to implement nontrivial delays and batching strategies whose existence is only known to the particular hop or tunnel gateway that receives the message, allowing a mostly low latency mixnet to provide cover traffic for higher latency communication (e.g. email). However we are aware that significant delays are required to provide meaningful protection, and that implementation of such delays will be a significant challenge. It is not clear at this time whether we will actually implement these delay features.

In theory, routers along the message path may inject an arbitrary number of hops before forwarding the message to the next peer, though the current implementation does not.

The Threat Model

I2P design started in 2003, not long after the advent of [Onion Routing], [Freenet], and [Tor]. Our design benefits substantially from the research published around that time. I2P uses several onion routing techniques, so we continue to benefit from the significant academic interest in Tor.

Taking from the attacks and analysis put forth in the anonymity literature (largely Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems), the following briefly describes a wide variety of attacks as well as many of I2Ps defenses. We update this list to include new attacks as they are identified.

Included are some attacks that may be unique to I2P. We do not have good answers for all these attacks, however we continue to do research and improve our defenses.

In addition, many of these attacks are significantly easier than they should be, due to the modest size of the current network. While we are aware of some limitations that need to be addressed, I2P is designed to support hundreds of thousands, or millions, of participants. As we continue to spread the word and grow the network, these attacks will become much harder.

The network comparisons and "garlic" terminology pages may also be helpful to review.

Brute force attacks

A brute force attack can be mounted by a global passive or active adversary, watching all the messages pass between all of the nodes and attempting to correlate which message follows which path. Mounting this attack against I2P should be nontrivial, as all peers in the network are frequently sending messages (both end to end and network maintenance messages), plus an end to end message changes size and data along its path. In addition, the external adversary does not have access to the messages either, as inter-router communication is both encrypted and streamed (making two 1024 byte messages indistinguishable from one 2048 byte message).

However, a powerful attacker can use brute force to detect trends - if they can send 5GB to an I2P destination and monitor everyone's network connection, they can eliminate all peers who did not receive 5GB of data. Techniques to defeat this attack exist, but may be prohibitively expensive (see: Tarzan's mimics or constant rate traffic). Most users are not concerned with this attack, as the cost of mounting it are extreme (and often require illegal activity). However, the attack is still possible, for example by an observer at a large ISP or an Internet exchange point. Those who want to defend against it would want to take appropriate countermeasures, such as setting low bandwidth limits, and using unpublished or encrypted leasesets for I2P Sites. Other countermeasures, such as nontrivial delays and restricted routes, are not currently implemented.

As a partial defense against a single router or group of routers trying to route all the network's traffic, routers contain limits as to how many tunnels can be routed through a single peer. As the network grows, these limits are subject to further adjustment. Other mechanisms for peer rating, selection and avoidance are discussed on the peer selection page.

Timing attacks

I2P's messages are unidirectional and do not necessarily imply that a reply will be sent. However, applications on top of I2P will most likely have recognizable patterns within the frequency of their messages - for instance, an HTTP request will be a small message with a large sequence of reply messages containing the HTTP response. Using this data as well as a broad view of the network topology, an attacker may be able to disqualify some links as being too slow to have passed the message along.

This sort of attack is powerful, but its applicability to I2P is non obvious, as the variation on message delays due to queuing, message processing, and throttling will often meet or exceed the time of passing a message along a single link - even when the attacker knows that a reply will be sent as soon as the message is received. There are some scenarios which will expose fairly automatic replies though - the

streaming library does (with the SYN+ACK) as does the message mode of guaranteed delivery (with the DataMessage+DeliveryStatusMessage).

Without protocol scrubbing or higher latency, global active adversaries can gain substantial information. As such, people concerned with these attacks could increase the latency (using nontrivial delays or batching strategies), include protocol scrubbing, or other advanced tunnel routing techniques, but these are unimplemented in I2P.

References: Low-Resource Routing Attacks Against Anonymous Systems

Intersection attacks

Intersection attacks against low latency systems are extremely powerful - periodically make contact with the target and keep track of what peers are on the network. Over time, as node churn occurs the attacker will gain significant information about the target by simply intersecting the sets of peers that are online when a message successfully goes through. The cost of this attack is significant as the network grows, but may be feasible in some scenarios.

In summary, if an attacker is at both ends of your tunnel at the same time, he may be successful. I2P does not have a full defense to this for low latency communication. This is an inherent weakness of low-latency onion routing. Tor provides a similar disclaimer.

Partial defenses implemented in I2P:

- strict ordering of peers
- peer profiling and selection from a small group that changes slowly
- Limits on the number of tunnels routed through a single peer
- Prevention of peers from the same /16 IP range from being members of a single tunnel
- For I2P Sites or other hosted services, we support simultaneous hosting on multiple routers, or multihoming

Even in total, these defenses are not a complete solution. Also, we have made some design choices that may significantly increase our vulnerability:

- We do not use low-bandwidth "guard nodes"
- We use tunnel pools comprised of several tunnels, and traffic can shift from tunnel to tunnel.
- Tunnels are not long-lived; new tunnels are built every 10 minutes.
- Tunnel lengths are configurable. While 3-hop tunnels are recommended for full protection, several applications and services use 2-hop tunnels by default.

In the future, it could for peers who can afford significant delays (per nontrivial delays and batching strategies). In addition, this is only relevant for destinations that other people know about - a private group whose destination is only known to trusted peers does not have to worry, as an adversary can't "ping" them to mount the attack.

Reference: One Cell Enough

Denial of service attacks

There are a whole slew of denial of service attacks available against I2P, each with different costs and consequences:

Greedy user attack: This is simply people trying to consume significantly more resources than they are willing to contribute. The defense against this is:

- Set defaults so that most users provide resources to the network. In I2P, users route traffic by default. In sharp distinction to other networks, over 95% of I2P users relay traffic for others.

- Provide easy configuration options so that users may increase their contribution (share percentage) to the network. Display easy-to-understand metrics such as "share ratio" so that users may see what they are contributing.
- Maintain a strong community with blogs, forums, IRC, and other means of communication.

Starvation attack: A hostile user may attempt to harm the network by creating a significant number of peers in the network who are not identified as being under control of the same entity (as with Sybil). These nodes then decide not to provide any resources to the network, causing existing peers to search through a larger network database or request more tunnels than should be necessary. Alternatively, the nodes may provide intermittent service by periodically dropping selected traffic, or refusing connections to certain peers. This behavior may be indistinguishable from that of a heavily-loaded or failing node. I2P addresses these issues by maintaining profiles on the peers, attempting to identify underperforming ones and simply ignoring them, or using them rarely. We have significantly enhanced the ability to recognize and avoid troublesome peers; however there are still significant efforts required in this area.

Flooding attack: A hostile user may attempt to flood the network, a peer, a destination, or a tunnel. Network and peer flooding is possible, and I2P does nothing to prevent standard IP layer flooding. The flooding of a destination with messages by sending a large number to the target's various inbound tunnel gateways is possible, but the destination will know this both by the contents of the message and because the tunnel's tests will fail. The same goes for flooding just a single tunnel. I2P has no defenses for a network flooding attack. For a destination and tunnel flooding attack, the target identifies which tunnels are unresponsive and builds new ones. New code could also be written to add even more tunnels if the client wishes to handle the larger load. If, on the other hand, the load is more than the client can deal with, they can instruct the tunnels to throttle the number of messages or bytes they should pass on (once the advanced tunnel operation is implemented).

CPU load attack: There are currently some methods for people to remotely request that a peer perform some cryptographically expensive operation, and a hostile attacker could use these to flood that peer with a large number of them in an attempt to overload the CPU. Both using good engineering practices and potentially requiring nontrivial certificates (e.g. HashCash) to be attached to these expensive requests should mitigate the issue, though there may be room for an attacker to exploit various bugs in the implementation.

Floodfill DOS attack: A hostile user may attempt to harm the network by becoming a floodfill router. The current defenses against unreliable, intermittent, or malicious floodfill routers are poor. A floodfill router may provide bad or no response to lookups, and it may also interfere with inter-floodfill communication. Some defenses and peer profiling are implemented, however there is much more to do. For more information see the network database page.

Tagging attacks

Tagging attacks - modifying a message so that it can later be identified further along the path - are by themselves impossible in I2P, as messages passed through tunnels are signed. However, if an attacker is the inbound tunnel gateway as well as a participant further along in that tunnel, with collusion they can identify the fact that they are in the same tunnel (and prior to adding unique hop ids and other updates, colluding peers within the same tunnel can recognize that fact without any effort). An attacker in an outbound tunnel and any part of an inbound tunnel cannot collude however, as the tunnel encryption pads and modifies the data separately for the inbound and outbound tunnels. External attackers cannot do anything, as the links are encrypted and messages signed.

Partitioning attacks

Partitioning attacks - finding ways to segregate (technically or analytically) the peers in a network - are important to keep in mind when dealing with a powerful adversary, since the size of the network plays a key role in determining your anonymity. Technical partitioning by cutting links between peers to create fragmented networks is addressed by I2P's built in network database, which maintains statistics about

various peers so as to allow any existing connections to other fragmented sections to be exploited so as to heal the network. However, if the attacker does disconnect all links to uncontrolled peers, essentially isolating the target, no amount of network database healing will fix it. At that point, the only thing the router can hope to do is notice that a significant number of previously reliable peers have become unavailable and alert the client that it is temporarily disconnected (this detection code is not implemented at the moment).

Partitioning the network analytically by looking for differences in how routers and destinations behave and grouping them accordingly is also a very powerful attack. For instance, an attacker harvesting the network database will know when a particular destination has 5 inbound tunnels in their LeaseSet while others have only 2 or 3, allowing the adversary to potentially partition clients by the number of tunnels selected. Another partition is possible when dealing with the nontrivial delays and batching strategies, as the tunnel gateways and the particular hops with non-zero delays will likely stand out. However, this data is only exposed to those specific hops, so to partition effectively on that matter, the attacker would need to control a significant portion of the network (and still that would only be a probabilistic partition, as they wouldn't know which other tunnels or messages have those delays).

Also discussed on the network database page (bootstrap attack).

Predecessor attacks

The predecessor attack is passively gathering statistics in an attempt to see what peers are 'close' to the destination by participating in their tunnels and keeping track of the previous or next hop (for outbound or inbound tunnels, respectively). Over time, using a perfectly random sample of peers and random ordering, an attacker would be able to see which peer shows up as 'closer' statistically more than the rest, and that peer would in turn be where the target is located.

I2P avoids this in four ways: first, the peers selected to participate in tunnels are not randomly sampled throughout the network - they are derived from the peer selection algorithm which breaks them into tiers. Second, with strict ordering of peers in a tunnel, the fact that a peer shows up more frequently does not mean they're the source. Third, with permuted tunnel length (not enabled by default) even 0 hop tunnels can provide plausible deniability as the occasional variation of the gateway will look like normal tunnels. Fourth, with restricted routes (unimplemented), only the peer with a restricted connection to the target will ever contact the target, while attackers will merely run into that gateway.

The current tunnel build method was specifically designed to combat the predecessor attack. See also the intersection attack.

References: http://forensics.umass.edu/pubs/wright.tissec.2008.pdf which is an update to the 2004 predecessor attack paper http://forensics.umass.edu/pubs/wright-tissec.pdf.

Harvesting attacks

"Harvesting" means compiling a list of users running I2P. It can be used for legal attacks and to help other attacks by simply running a peer, seeing who it connects to, and harvesting whatever references to other peers it can find.

I2P itself is not designed with effective defenses against this attack, since there is the distributed network database containing just this information. The following factors make the attack somewhat harder in practice:

- Network growth will make it more difficult to obtain a given proportion of the network
- Floodfill routers implement query limits as DOS protection
- "Hidden mode", which prevents a router from publishing its information to the netDb, (but also prevents it from relaying data) is not widely used now but could be.

In future implementations, basic and comprehensive restricted routes, this attack loses much of its

power, as the "hidden" peers do not publish their contact addresses in the network database - only the tunnels through which they can be reached (as well as their public keys, etc).

In the future, routers could use GeoIP to identify if they are in a particular country where identification as an I2P node would be risky. In that case, the router could automatically enable hidden mode, or enact other restricted route methods.

Identification Through Traffic Analysis

By inspecting the traffic into and out of a router, a malicious ISP or state-level firewall could identify that a computer is running I2P. As discussed above, I2P is not specifically designed to hide that a computer is running I2P. However, several design decisions made in the design of the transport layer and protocols make it somewhat difficult to identify I2P traffic:

- Random port selection
- Point-to-Point Encryption of all traffic
- DH key exchange with no protocol bytes or other unencrypted constant fields
- Simultaneous use of both TCP and UDP transports. UDP may be much harder for some Deep Packet Inspection (DPI) equipment to track.

In the near future, we plan to directly address traffic analysis issues by further obfuscation of I2P transport protocols, possibly including:

- Padding at the transport layer to random lengths, especially during the connection handshake
- Study of packet size distribution signatures, and additional padding as necessary
- Development of additional transport methods that mimic SSL or other common protocols
- Review of padding strategies at higher layers to see how they affect packet sizes at the transport layer
- Review of methods implemented by various state-level firewalls to block Tor
- Working directly with DPI and obfuscation experts

Reference: Breaking and Improving Protocol Obfuscation

Sybil attacks

Sybil describes a category of attacks where the adversary creates arbitrarily large numbers of colluding nodes and uses the increased numbers to help mounting other attacks. For instance, if an attacker is in a network where peers are selected randomly and they want an 80% chance to be one of those peers, they simply create five times the number of nodes that are in the network and roll the dice. When identity is free, Sybil can be a very potent technique for a powerful adversary. The primary technique to address this is simply to make identity 'non free' - Tarzan (among others) uses the fact that IP addresses are limited, while IIP used HashCash to 'charge' for creating a new identity. We currently have not implemented any particular technique to address Sybil, but do include placeholder certificates in the router's and destination's data structures which can contain a HashCash certificate of appropriate value when necessary (or some other certificate proving scarcity).

Requiring HashCash Certificates in various places has two major problems:

- Maintaining backward compatibility
- The classic HashCash problem - selecting HashCash values that are meaningful proofs of work on high-end machines, while still being feasible on low-end machines such as mobile devices.

Various limitations on the number of routers in a given IP range restrict the vulnerability to attackers that don't have the ability to put machines in several IP blocks. However, this is not a meaningful defense against a powerful adversary.

See the network database page for more Sybil discussion.

Buddy Exhaustion attacks

(Reference: In Search of an Anonymous and Secure Lookup Section 5.2)

By refusing to accept or forward tunnel build requests, except to a colluding peer, a router could ensure that a tunnel is formed wholly from its set of colluding routers. The chances of success are enhanced if there is a large number of colluding routers, i.e. a Sybil attack. This is somewhat mitigated by our peer profiling methods used to monitor the performance of peers. However, this is a powerful attack as the number of routers approaches $f = 0.2$, or 20% malicious nodes, as specifed in the paper. The malicous routers could also maintain connections to the target router and provide excellent forwarding bandwidth for traffic over those connections, in an attempt to manipulate the profiles managed by the target and appear attractive. Further research and defenses may be necessary.

Cryptographic attacks

We use strong cryptography with long keys, and we assume the security of the industry-standard cryptographic primitives used in I2P, as documented on the low-level cryptography page. Security features include the immediate detection of altered messages along the path, the inability to decrypt messages not addressed to you, and defense against man-in-the-middle attacks. The key sizes chosen in 2003 were quite conservative at the time, and are still longer than those used in other anonymity networks. We don't think the current key lengths are our biggest weakness, especially for traditional, non-state-level adversaries; bugs and the small size of the network are much more worrisome. Of course, all cryptographic algorithms eventually become obsolete due to the advent of faster processors, cryptographic research, and advancements in methods such as rainbow tables, clusters of video game hardware, etc. Unfortunately, I2P was not designed with easy mechanisms to lengthen keys or change shared secret values while maintaining backward compatibility.

Upgrading the various data structures and protocols to support longer keys will have to be tackled eventually, and this will be a major undertaking, just as it will be for others. Hopefully, through careful planning, we can minimize the disruption, and implement mechanisms to make it easier for future transitions.

In the future, several I2P protocols and data structures support securely padding messages to arbitrary sizes, so messages could be made constant size or garlic messages could be modified randomly so that some cloves appear to contain more subcloves than they actually do. At the moment, however, garlic, tunnel, and end to end messages include simple random padding.

Floodfill Anonymity attacks

In addition to the floodfill DOS attacks described above, floodfill routers are uniquely positioned to learn about network participants, due to their role in the netDb, and the high frequency of communication with those participants. This is somewhat mitigated because floodfill routers only manage a portion of the total keyspace, and the keyspace rotates daily, as explained on the network database page. The specific mechanisms by which routers communicate with floodfills have been carefully designed. However, these threats should be studied further. The specific potential threats and corresponding defenses are a topic for future research.

Other Network Database attacks

A hostile user may attempt to harm the network by creating one or more floodfill routers and crafting them to offer bad, slow, or no responses. Several scenarios are discussed on the network database page.

Central Resource Attacks

There are a few centralized or limited resources (some inside I2P, some not) that could be attacked or

used as a vector for attacks. The absence of jrandom starting November 2007, followed by the loss of the i2p.net hosting service in January 2008, highlighted numerous centralized resources in the development and operation of the I2P network, most of which are now distributed. Attacks on externally-reachable resources mainly affect the ability of new users to find us, not the operation of the network itself.

- The website is mirrored and uses DNS round-robin for external public access.
- Routers now support multiple external reseed locations, however more reseed hosts may be needed, and the handling of unreliable or malicious reseed hosts may need improvement.
- Routers now support multiple update file locations. A malicious update host could feed a huge file, need to limit the size.
- Routers now support multiple default trusted update signers.
- Routers now better handle multiple unreliable floodfill peers. Malicious floodfills needs more study.
- The code is now stored in a distributed source control system.
- Routers rely on a single news host, but there is a hardcoded backup URL pointing to a different host. A malicious news host could feed a huge file, need to limit the size.
- Naming system services, including address book subscription providers, add-host services, and jump services, could be malicious. Substantial protections for subscriptions were implemented in release 0.6.1.31, with additional enhancements in subsequent releases. However, all naming services require some measure of trust, see the naming page for details.
- We remain reliant on the DNS service for i2p2.de, losing this would cause substantial disruption in our ability to attract new users, and would shrink the network (in the short-to-medium term), just as the loss of i2p.net did.

Development attacks

These attacks aren't directly on the network, but instead go after its development team by either introducing legal hurdles on anyone contributing to the development of the software, or by using whatever means are available to get the developers to subvert the software. Traditional technical measures cannot defeat these attacks, and if someone threatened the life or livelihood of a developer (or even just issuing a court order along with a gag order, under threat of prison), we would have a big problem.

However, two techniques help defend against these attacks:

- All components of the network must be open source to enable inspection, verification, modification, and improvement. If a developer is compromised, once it is noticed the community should demand explanation and cease to accept that developer's work. All checkins to our distributed source control system are cryptographically signed, and the release packagers use a trust-list system to restrict modifications to those previously approved.
- Development over the network itself, allowing developers to stay anonymous but still secure the development process. All I2P development can occur through I2P - using a distributed source control system, a distributed source control system, IRC chat, public web servers, discussion forums (forum.i2p), and the software distribution sites, all available within I2P.

We also maintain relationships with various organizations that offer legal advice, should any defense be necessary.

Implementation attacks (bugs)

Try as we might, most nontrivial applications include errors in the design or implementation, and I2P is no exception. There may be bugs that could be exploited to attack the anonymity or security of the communication running over I2P in unexpected ways. To help withstand attacks against the design or protocols in use, we publish all designs and documentation and solicit review and criticism with the hope that many eyes will improve the system. We do not believe in security through obscurity.

In addition, the code is being treated the same way, with little aversion towards reworking or throwing out

something that isn't meeting the needs of the software system (including ease of modification). Documentation for the design and implementation of the network and the software components are an essential part of security, as without them it is unlikely that developers would be willing to spend the time to learn the software enough to identify shortcomings and bugs.

Our software is likely, in particular, to contain bugs related to denial of service through out-of-memory errors (OOMs), cross-site-scripting (XSS) issues in the router console, and other vulnerabilities to non-standard inputs via the various protocols.

I2P is still a small network with a small development community and almost no interest from academic or research groups. Therefore we lack the analysis that other anonymity networks may have received. We continue to recruit people to get involved and help.

Other Defenses

Blocklists

To some extent, I2P could be enhanced to avoid peers operating at IP addresses listed in a blocklist. Several blocklists are commonly available in standard formats, listing anti-P2P organizations, potential state-level adversaries, and others.

To the extent that active peers actually do show up in the actual blocklist, blocking by only a subset of peers would tend to segment the network, exacerbate reachability problems, and decrease overall reliability. Therefore we would want to agree on a particular blocklist and enable it by default.

Blocklists are only a part (perhaps a small part) of an array of defenses against maliciousness. In large part the profiling system does a good job of measuring router behavior so that we don't need to trust anything in netDb. However there is more that can be done. For each of the areas in the list above there are improvements we can make in detecting badness.

If a blocklist is hosted at a central location with automatic updates the network is vulnerable to a central resource attack. Automatic subscription to a list gives the list provider the power to shut the i2p network down. Completely.

Currently, a default blocklist is distributed with our software, listing only the IPs of past DOS sources. There is no automatic update mechanism. Should a particular IP range implement serious attacks on the I2P network, we would have to ask people to update their blocklist manually through out-of-band mechanisms such as forums, blogs, etc.