Hamachi Security – an Overview

LogMeIn White Papers

Abstract

The purpose of this white paper is to provide an overview of LogMeIn's VPN service, Hamachi. We at LogMeIn do not believe in security through obscurity, nor do we expect our customers to blindly accept our claims as to important security features, such as end-to-end encryption. By disclosing our security architecture, we are also inviting the public to scrutinize our efforts.

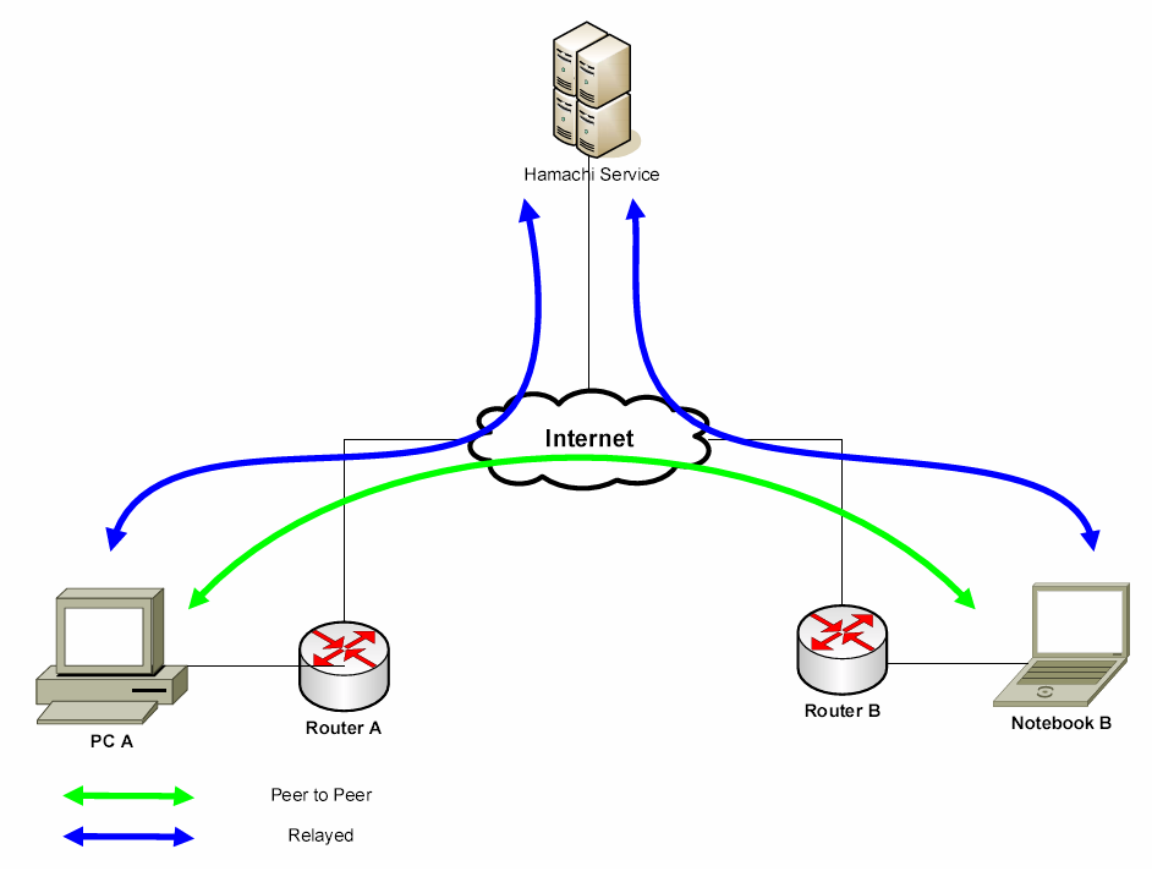
Audience

This paper is an overview of LogMeIn Hamachi's security architecture, and is written for those who have a general understanding of VPN technology. The audience, after reading this white paper and combining the information contained herein with existing knowledge and resources, can perform the necessary analysis before deploying our product.

Terminology

In the LogMeIn Hamachi architecture, there are two entities that take part in every VPN session. The "End-Node," "Client" or "Peer" is the machine on which Hamachi is installed. The "Mediation Server" is the Hamachi Server that mediates the connection between various peers.

LogMeIn Hamachi Architecture



LogMeIn Hamachi is a virtual private network system, comprised of mediation servers managed by LogMeIn, and end-node Hamachi clients. The Hamachi servers provide the mediation services required for establishing direct peer-to-peer tunnels between clients.

Every client creates and maintains a control connection to the mediation server. The connection is used for learning the client's location, tracking its online presence and assisting a pair of clients in establishing a VPN tunnel. If the connection becomes idle, it is re-activated with a small "keep alive" packet to ensure that any intermediate firewalls do not shutdown the connection due to inactivity.

When the control connection is established, the client goes through a three step initialization process: Login, Probing, and Synchronization:

- The Login step authenticates both the client to the server, and the server to the client.
- The Probing step determines the topology of client's internet connection, specifically it detects the presence of NAT and firewall devices that may be present on the route between the peer and the Internet.
- The Synchronization step brings the client's view of its networks in sync with the server and marks it as online. It also initiates the setup of VPN tunnels as dictated by the client's network memberships and the online status of its peers.

When a peer of any given network goes online or offline, the server directs other network peers to either create or tear down the tunnel to the former. When establishing tunnels between the peers, Hamachi always attempts to setup a direct connection first. It does this by Hamachi's proprietary NAT-traversal technique, which is similar in concept to the ideas of UDP hole punching. If a direct UDP connection cannot be established, Hamachi attempts to connect peers using a direct TCP connection. If this also fails, the peers will be instructed to communicate via one of the Hamachi relay servers, using UDP if available or TCP if either client is not capable of UDP communication.

If a control connection with the server is unexpectedly lost, the client retains all of its tunnels and begins to actively check their status. When the server loses a client's connection, it informs the client's peers and they also start tunnel status checks. This enables Hamachi tunnels to withstand chance issues on the route between the client and the server, as well as short periods of complete server unavailability.

In approximately 95% of the tunnels established by LogMeIn Hamachi, a direct peer to peer connection is established. This includes peers who both reside behind different firewalls or broadband routers (aka, NAT devices).

Design Fundamentals

LogMeIn Hamachi was designed with security and ease of use as the foremost priorities. Hamachi Security breaks down into three distinct categories: authentication, encryption, and administrative options.

LogMeIn Hamachi Security Features

VPNs ensure message security between clients and servers by verifying peers' identities and encrypting and authenticating data. The LogMeIn Hamachi authentication process is based on the principles of public key cryptography.

The provisions below ensure both the privacy and authenticity of all client-server and client-client communications.

Authentication

The authentication process ensures that the identities of everyone in your network, from the LogMeIn Hamachi servers to your peers, are verified.

Client

A Hamachi client is identified by its Hamachi network address. The address is assigned the first time the client connects to the mediation servers, and it stays the same for as long as the client's account exists in the system.

The client also generates an RSA key pair, which is used for authentication purposes during the login sequence. The public key is passed to the server once - during the first connection when creating a new account. The private key stays on the client machine and it is never disclosed to anyone else.

To login, the client submits its Hamachi IP and uses its private key to sign the server's challenge. The server verifies the signature and this authenticates the client.

Server

Each Hamachi server owns an RSA key pair. The public key is distributed with the client's installation package, and thus, it is known to the client prior to the first contact with the server.

When the client connects to the server, it announces which key pair it expects the server to have. If the server has the requested key pair, the login sequence commences. In the last message of this sequence, the server sends a signature of the client's data and this confirms the server's identity to the client.

For further information on Authentication, please refer to the Appendix.

Encryption

Encryption is a method that scrambles and unscrambles various pieces of information so that they can be sent securely from one location to another. When a Hamachi peer sends data to a member of their network, they encrypt it before sending it through the Internet. At the other location, the authenticated peer verifies that the data has not been altered or duplicated and decrypts it.

In the case of LogMeIn Hamachi, a key exchange takes place between peers once each is authenticated by the Hamachi mediation server, and a tunnel has been established. Clients use each other's public keys to authenticate this exchange.

Example: User A and User B wish to exchange information for the first time. First, both of them are authenticated by the Hamachi server using an RSA key pair. Once the users are deemed authentic, they are then associated with their respective networks (if any). Since User A and User B are new users, they have to join the same network and establish tunnels. Once the tunnels are created, each peer requests the others public key from the Hamachi server and uses this key to verify each others authenticity. In parallel with this process, Users A and B also generate and exchange a random set of encryption keys. These keys are not known to the Hamachi servers, or anyone else outside of these two peers.

For added security, the client software also includes a feature that enables users to verify the authenticity of all previously unseen peers' public keys. If enabled, when a key is first retrieved from the server, the client will use it to authenticate the key exchange as usual, but all VPN traffic over the respective tunnel will be blocked. The user will then be required to manually verify a fingerprint of each peer's key via an out of band channel (for example, over the phone) and explicitly remove the tunnel block. This will mark the key as trusted and it will be cached in the client's local storage of peers' keys.

In its default configuration, LogMeIn Hamachi uses industry standard AES encryption algorithm with 256 bit key to protect all bulk data traffic in server to client and peer to peer traffic. Hamachi uses the following crypto suite:

- DH group – 2048-bit MODP group from RFC 3526
- Message encryption – AES-256-CBC using ESP-style padding
- Message authentication – 96-bit version of HMAC-SHA1

For more detail on the encryption process, please refer to the Appendix.

Administrative Security

LogMeIn Hamachi includes administrative features to help maintain the security of your Hamachi Network.

Network Access

- Password Protection
  - Anyone trying to join the network is required to present a valid network password.
  - The password may be turned off and it can be changed by an owner or an administrator.

Network Lock

- The network can be locked to prevent any new members from joining it. This preserves all other access control settings. This is a recommended setting for networks that do not change frequently.

Membership Approval

- All new members do not have full access to the network until manually approved by an owner or an administrator. This is a recommended setting for all private networks as this feature completely eliminates the chances of somebody infiltrating the network by guessing or brute-forcing the password.
- Note however that Hamachi includes effective mechanism for preventing network password brute-forcing, so any reasonably random and long password provides an adequate level of protection in a majority of cases.

Network Administration

- Member Eviction
  - An owner or an administrator can evict existing network members. This revokes their membership, but does not prevent them from re-joining the network.
- Member Ban
  - A network can prepare a list of Hamachi clients that cannot join it under any condition.
  - The ban list is managed by an owner or a network administrator.
- Network Administrators
  - An owner of the network can assign one or more network members to be network administrators.
  - An owner can also control the list of rights assigned to administrators:
    - Set network password
    - Lock network and modify other network access options
    - Approve and reject membership requests
    - Evict members
    - Ban and unban members

Windows Services

- For added security, access to Windows services that are typically considered vulnerable can be blocked from within the LogMeIn Hamachi client. This block applies only to the requests sent/received via Hamachi networks and it includes NetBIOS and SMB protocols, which are responsible for Windows File and Printer Sharing, Network Neighborhood browsing and other services.

Conclusion

A secure VPN makes the best use out of the right technology. LogMeIn Hamachi is a verifiably secure peer-to-peer VPN: the security and privacy of your data is ensured from the moment it leaves your computer until it arrives at your peer's.

Appendix

Encryption Protocol Details

HELO

The client connects to the server and sends a HELO message:

HELO CryptoSuite ServerKfp Ni Gi

CryptoSuite is 1 for the default crypto suite, ServerKfp is the OpenSSH-style fingerprint of the server's public key, Ni and Gi are the client's 1024-bit nonce and public DH exponent.

If the server has a public key that matches the ServerKfp, it replies with:

HELO OK Nr Gr

where Nr and Gr are the server's nonce and public DH exponent.

KEYMAT

At this point both the server and client can compute shared DH secret and generate keying material as follows:

KEYMAT = T1 | T2 | T3 | ..

T1 = prf (K, Ni | Nr | 0x01)
T2 = prf (K, T1 | Ni | Nr | 0x02)
T3 = prf (K, T2 | Ni | Nr | 0x03)
..

where K is a shared DH secret, and prf is HMAC-SHA1.

All subsequent protocol messages are encrypted with the Ke key and authenticated using the Ka key, where both keys are instantiated from KEYMAT. In case of the default crypto suite, Ke uses the first 256 bits of KEYMAT, and Ka the next 160 bits.

Message Protection

Prior to encrypting a protocol message, the sender pads it to the size of the cipher block (16 bytes with the default crypto suite) using ESP padding. Next the random IV is generated and the message is encrypted. The resulting ciphertext is appended with an IV and prepended with a message ID (a monotonically increasing 32-bit number). If the message is sent over TCP, it is also appended with a 32-bit size field. For UDP messages it is prepended with a 32-bit SPI (see below). And finally, HMAC is computed over the entire message, appended to it and then the message is sent out.

The above message protection scheme is consistent with those employed by TLS and IKE/IPsec.

AUTH

The client logs into the system by sending an AUTH message:

AUTH Identity Signature(CryptoSuite | Ni | Nr | Gi | Gr, Kpri_cli)

where Identity is the client's 32-bit Hamachi address and Signature is the SHA1 hash of the concatenated nonces and public DH exponents, encrypted with client's private key.

The server uses the ID to locate the client's account, obtains its public key and verifies the signature. If the signature is correct, the server replies with:

AUTH OK Signature(Nr | Ni | Gr | Gi, Kpri_srv)

where Signature is created using the server's private key that matches the ServerKfp from the HELO message.

Note: CryptoSuite is included into an authentication hash, starting with version 1.0.0.52 of the Windows client. This is not strictly required, but it does protect against crypto suite downgrade attacks should these become feasible as the protocol develops. This change is in response to comments received on the *cryptography* mailing list.

Peer-to-peer traffic

Once the mediation server brings the tunnel up between the two peers, it generates a pair of random 32-bits numbers a.k.a. SPIs. The SPIs are used to tag the traffic flowing between these two peers - one SPI identifies traffic in one direction, the other in the opposite direction.

When the client receives an encrypted UDP packet, it looks the peer up by the packet's SPI, obtains the crypto keys and proceeds with packet authentication and decryption.

P2P crypto keys are derived differently depending on the client version. Earlier versions (0.9.9.5 and older) were received KEYMAT from the server, which generated it randomly on a per-tunnel basis.

All recent clients (0.9.9.6 and newer) perform their own key exchange to generate KEYMAT. The key exchange protocol is as follows:

First, the peers select who will initiate the key exchange by comparing the tunnels' SPI values with each other. The peer with the smaller inbound SPI is the initiator.

The initiator sends:

KE1 Ni Gi

where Ni is a 1024 nonce and Gi a public DH exponent. The initiator keeps re-sending this message in N msec intervals until it receives a response:

KE2 Nr Gr Signature(Nr | Ni | Gr | Gi, Kpri_r)

where Nr and Gr are the responder's nonce and exponent and Signature is created with the responder's private key. Note that the responder merely generates Nr and Gr, and does not derive p2p keys yet, as Ni and Gi may not be authentic.

The initiator verifies the signature, derives p2p keys (using the same KEYMAT generation and extension mechanism as with in client-server case) and replies with:

KE3 Ni Gi Signature(Ni | Nr | Gi | Gr, Kpri_i)

The responder verifies the signature, derives p2p keys and sends a dummy encrypted message to the initiator to indicate successful completion of the key exchange.

Similar to the KE1 message, the initiator re-sends the KE3 message periodically until it hears back from the responder.

In order to verify the peers' signatures during the key exchange, the client must have a peer's public key. It can obtain this key in two ways – either by querying it from the mediation server or by receiving it from the peer through another channel and installing it manually.

Note that for uber-secure deployments it is a good idea to subject all newly obtained public keys to a manual verification to ensure their authenticity.

References

The described security architecture is based on design ideas found in the following protocols and frameworks:

- IPsec/IKE – RFC 2401, 2407, 2408, 2409, 2412
- SSL/TLS – RFC 2246
- JFK – IETF draft ipsec-jfk-04

For more general information on key systems, please visit the following links:

- Block ciphers and chaining modes – http://www.rsasecurity.com/rsalabs/node.asp?id=2168
- Diffie-Hellman key exchange – http://www.rsasecurity.com/rsalabs/node.asp?id=2248
- Public key encryption – http://www.rsasecurity.com/rsalabs/node.asp?id=2165
- Message authentication codes – http://www.rsasecurity.com/rsalabs/node.asp?id=2177