LIVRE BLANC

v1.0.20

Warren

Un VPN no-log construit sur QUIC : identité non-custodiale, résistance à la censure, et le port forwarding qu'on a cru condamné

DATE	14 juillet 2026
PÉRIMÈTRE	Motivations, architecture de la stack, moteur WarrenGuard, réseau, identité, paiement, applications, modèle de menace, limites et trajectoire
STATUT	Public
VERSION	v1.0.20
AUTEUR	Équipe Warren

Résumé

Warren est un VPN no-log grand public dont l'architecture repose sur une décision radicale : faire du protocole QUIC, et non d'une brique de tunnel dédiée, la fondation de toute la pile. De ce choix découle le reste. Le trafic se confond avec du HTTP/3 ordinaire, ce qui donne une résistance à la censure native plutôt que rapportée. L'identité de l'utilisateur est une clé publique dérivée localement d'une phrase de récupération, sans compte ni adresse e-mail : le serveur ne détient aucun secret et un vol de sa base de données ne compromet aucun accès. Les serveurs de sortie fonctionnent sur une image immuable, leur état d'exécution ne vit qu'en mémoire vive, et il n'y a aucune partition de données : une saisie physique ne rend rien.

La confidentialité a changé de nature. La collecte de données est devenue une infrastructure permanente et peu coûteuse : ce qui est capté est conservé, et reste exploitable des années plus tard, sous d'autres lois et par d'autres acteurs que ceux du moment. Face à une mémoire d'une telle portée, la prudence individuelle ne protège plus grand-chose tant que la donnée continue d'exister quelque part. La seule protection qui tienne dans le temps est l'absence de donnée, et c'est le principe qui gouverne Warren : rendre la donnée sensible absente plutôt que d'en promettre le bon usage.

Trois convictions guident le projet. La première : le no-log n'a de valeur que s'il est vérifiable, pas seulement promis. La deuxième : le port forwarding, que les principaux acteurs du marché ont abandonné en 2023 sous la pression de l'abus, reste une fonctionnalité légitime dont il faut durcir le mécanisme sans en restreindre l'usage. La troisième : chaque garantie de sécurité doit tenir par construction, sans case à cocher que l'utilisateur aurait à trouver. Ces protections forment une offre unique, sans paliers payants : ce qui varie d'une défense à l'autre est son état par défaut, réglé selon son coût en performance.

Ce document décrit l'ensemble de la stack, du moteur cryptographique jusqu'au plan de contrôle, en restant au niveau architectural. Il montre par quels mécanismes concrets chaque garantie tient, et où se situe la frontière de ce que Warren protège.

Sommaire

Partie I. Pourquoi Warren

Avant l'architecture, les raisons. Ce que vaut réellement une promesse no-log, ce que le sacrifice du port forwarding en 2023 dit du métier, et pourquoi la furtivité est une condition d'existence.

1. Le no-log, et pourquoi la parole ne suffit pas

Un VPN grand public vend une promesse simple : votre fournisseur d'accès, les réseaux publics que vous traversez et les sites que vous visitez ne peuvent plus vous suivre. Cette promesse repose entièrement sur une hypothèse rarement examinée : que le fournisseur de VPN, lui, ne garde aucune trace. Toute l'industrie s'appuie sur ce mot, « no-log », et la quasi-totalité des acteurs le décline en une politique de confidentialité que personne ne peut vérifier.

La faiblesse est structurelle. Une politique est un texte ; un serveur est un système. Rien n'oblige le second à respecter le premier, et l'utilisateur n'a aucun moyen d'observer ce qui se passe réellement à l'intérieur d'un serveur de sortie. Le marché du VPN est un marché de la confiance déclarative, et un marché de la confiance déclarative tend vers le moins-disant : classements sponsorisés, audits de complaisance, promesses invérifiables.

La seule sortie de ce piège consiste à remplacer la confiance par la vérification, et surtout à faire en sorte que la donnée sensible **n'existe pas**. Une donnée qui n'a jamais été écrite ne peut être ni saisie, ni réquisitionnée, ni fuitée, ni vendue. L'histoire récente du secteur en donne l'illustration la plus nette : le 18 avril 2023, au moins six policiers de la police nationale suédoise se sont présentés dans les bureaux de Mullvad à Göteborg, munis d'un mandat pour saisir des machines contenant des données clients. Ces données n'existaient pas. Après que Mullvad leur a démontré comment le service fonctionne, et après consultation du procureur, la police est repartie sans rien emporter. La minimisation des données avait fonctionné comme défense technique **et** juridique en même temps.

Mullvad est l'acteur qui a le plus sérieusement défriché ce terrain : ses décisions publiques constituent la meilleure documentation disponible sur les pièges du métier, et une part de ce que Warren construit consiste à en tirer les leçons.

Warren pousse cette idée jusqu'au bout de ses conséquences. Aucune identité réutilisable côté serveur. Aucun journal d'activité, aucune adresse IP source, aucune requête DNS, aucune association durable entre un paiement et un usage. Là où une trace résiduelle est techniquement inévitable, elle est bornée dans le temps, chiffrée au repos sous une clé absente de la base, et sa durée de vie est fixée précisément. Un no-log crédible se mesure à ce qu'il rend impossible, et ce document décrit les mécanismes qui le garantissent.

2. Mai 2023 : quand le port forwarding est devenu indéfendable

Le port forwarding permet à une machine derrière le VPN d'accepter des connexions entrantes : partager en pair-à-pair au lieu de seulement télécharger, héberger un service accessible depuis l'extérieur, obtenir un NAT ouvert pour du jeu en ligne. C'est une fonctionnalité que les utilisateurs techniques réclament et qui, longtemps, a distingué les bons fournisseurs des autres.

Elle a été abandonnée. Le 29 mai 2023, Mullvad annonçait le retrait total du port forwarding, les ports existants cessant de fonctionner le 1er juillet 2023. La justification, dans ses propres termes : *« des individus ont fréquemment utilisé cette fonctionnalité pour héberger du contenu indésirable et des services malveillants depuis les ports ouverts sur nos serveurs. Cela a conduit à des contacts des forces de l'ordre, à la mise en liste noire de nos IP, et à la résiliation de nos contrats par des hébergeurs. »* Un mois plus tard, le 29 juin 2023, IVPN emboîtait le pas, retrait complet au 30 septembre, en observant que l'afflux de clients venus d'un autre fournisseur, à la suite d'un changement de politique similaire, avait démultiplié le risque chez eux. L'abus se concentre toujours sur le dernier fournisseur qui tient la porte ouverte.

Le dilemme est réel : le coût d'abus d'une fonctionnalité peut dépasser sa valeur, jusqu'à devenir une question de survie pour l'infrastructure. Mais la réponse du marché, la suppression pure et simple, punit l'écrasante majorité des usages légitimes pour se prémunir contre une minorité nuisible. Warren fait le pari inverse, en le documentant : nous restaurons le port forwarding, nous **durcissons le mécanisme** contre les abus techniques et les fuites de désanonymisation, et nous **ne restreignons pas l'usage**. La section 15 décrit le mécanisme ; la section 32, les limites.

Un VPN doit servir ses utilisateurs plutôt que se protéger d'eux, et la bonne réponse à l'abus combine des mesures techniques et juridiques au lieu d'amputer une fonctionnalité.

3. L’obfuscation comme condition d’existence

On associe l’obfuscation à la censure d’État : l’Iran, la Chine, la Russie inspectent le trafic en profondeur (DPI) pour détecter et bloquer les protocoles de VPN. C’est vrai, mais c’est réducteur. Le blocage de l’UDP, ou celui d’une signature de protocole trop reconnaissable, arrive bien avant l’appareil d’État : dans le Wi-Fi d’un café, d’un train, d’un hôtel, d’une entreprise. Un VPN dont le protocole se laisse identifier au premier paquet est un VPN qui tombe sur une part importante des réseaux du quotidien.

La plupart des protocoles de tunnel exposent une signature triviale. Le handshake de WireGuard, par exemple, tient dans un paquet de taille fixe avec un octet de type caractéristique : un équipement de filtrage la reconnaît sans effort. Rendre un tel protocole furtif oblige à greffer une couche d’obfuscation par-dessus, à maintenir séparément, activée après coup, souvent réservée aux utilisateurs qui pensent à la chercher.

Nous avons voulu que la furtivité soit une propriété du transport lui-même. C’est la raison profonde du choix de QUIC (sections 5 à 8) : un tunnel QUIC ressemble, sur le fil, à un navigateur qui charge un site en HTTP/3. Un censeur qui voudrait le bloquer devrait bloquer HTTP/3, c’est-à-dire une fraction majeure du web moderne, avec les dégâts collatéraux que cela suppose. L’obfuscation de premier niveau, celle qui gomme les signaux visibles du protocole, est chez Warren **active par défaut**, sans réglage, pour tout le monde. Les défenses plus lourdes, celles qui déforment la forme même du trafic, se paient en débit et en latence : elles sont là, elles sont à la disposition de chacun, mais elles ne s’allument que si l’utilisateur les demande. Cette ligne de partage, entre ce qui est permanent parce que sans contrepartie et ce qui est optionnel parce qu’il en a une, structure toute notre approche de la censure (sections 12 et 14).

NOTE · UNE LIGNE DE PARTAGE TECHNIQUE, JAMAIS COMMERCIALE

Warren n’a ni palier, ni option payante, ni fonctionnalité réservée à qui paierait davantage. Il existe une offre unique, et tout ce que décrit ce document en fait partie, du multi-hop au port forwarding en passant par la défense contre l’analyse de trafic. La seule chose qui change d’une fonctionnalité à l’autre, c’est son état par défaut, et le seul critère est l’arbitrage entre protection et performance. Ce qui ne coûte rien en débit tourne en permanence pour tout le monde ; ce qui coûte en performance reste un choix de l’utilisateur et ne pèse jamais sur la facture.

4. Trois propriétés, une conviction

De ces motivations découlent trois propriétés structurantes, qui résument Warren avant tout détail technique.

Aucune identité réutilisable côté serveur. L'utilisateur est une clé publique qu'il génère lui-même, jamais un compte que nous détenons. Le serveur ne connaît que des clés publiques, inutiles seules, et des signatures à usage unique. Un vol de notre base de données ne donne accès à aucun compte, car il n'y a rien à voler qui soit un secret.

Aucun journal sur les serveurs de sortie. Les sessions actives, les allocations de ports, les tables de traduction d'adresses n'existent qu'en mémoire vive. Une machine éteinte ou saisie ne contient rien sur ses utilisateurs. Le no-log tient au système lui-même, conçu pour ne pas pouvoir journaliser durablement.

Le port forwarding restauré. Fonctionnalité de premier plan, avec un mécanisme durci et un usage non restreint.

La conviction sous-jacente est que la sécurité et la confidentialité doivent être des **propriétés de construction**. Un kill-switch protège parce qu'un socket est mort, et non parce qu'une règle de pare-feu a eu le temps de s'installer. Un no-log est garanti parce que la donnée n'est jamais écrite, et non parce qu'un script la purge plus tard. Une identité est anonyme parce que le protocole ne transporte pas le nom, et non parce qu'on a promis de ne pas le regarder. Tout le reste du document décline cette exigence.

Partie II. La genèse de la stack

Comment on arrive à un VPN bâti sur QUIC : les protocoles écartés, le détour qui a coûté cher, et ce que ce choix engage.

5. Pourquoi pas WireGuard

WireGuard est un excellent protocole : petit, audité, rapide, intégré au noyau Linux. Il fut le point de départ de notre réflexion.

Trois raisons nous en ont écartés. D'abord, un tunnel WireGuard nu se fait bloquer ou brider sur les réseaux qui filtrent l'UDP ou reconnaissent sa signature, et le rendre furtif impose une couche d'obfuscation supplémentaire à maintenir. Ensuite, WireGuard n'offre aucun chemin natif vers les fonctionnalités qui font le cœur de Warren : le multi-hop chiffré de bout en bout, le port forwarding en bande, la défense contre l'analyse de trafic. Chacune devrait être greffée par-dessus, avec son propre canal de contrôle et sa propre maintenance, car WireGuard n'offre ni négociation de capacités, ni canal de contrôle applicatif, ni extensibilité du handshake sur lesquels s'appuyer. Enfin, l'identité de WireGuard repose sur une clé propre au tunnel, distincte de l'identité cryptographique de l'utilisateur : deux identités à synchroniser, deux surfaces de corrélation, là où nous voulions une seule clé.

Ces trois raisons convergent vers la même conclusion : les propriétés requises ne se greffent pas sur WireGuard, elles doivent être portées par le transport lui-même.

6. Le détour par Iroh, le pivot vers QUIC

Le raisonnement nous a conduits vers QUIC, le protocole de transport moderne sur lequel repose HTTP/3. La première tentative s'est appuyée sur Iroh, une pile pair-à-pair bâtie sur QUIC, séduisante parce qu'elle alignait l'identité du nœud sur une clé publique et promettait le franchissement de NAT et le multipath. À l'usage, elle apportait surtout du poids : des milliers de lignes inutilisées pour notre cas, une interface qui bougeait chaque semaine, une classe de bugs de traversée de NAT non résolue, et une longue traîne de dépendances transitives.

En mai 2026, nous avons opéré un pivot net : abandonner Iroh pour Quinn, l'implémentation Rust de référence de QUIC, sans la surcouche pair-à-pair. Ses fonctions phares, le franchissement de NAT et le multipath, ne servaient à rien dans la topologie de Warren, où un client s'adresse à un serveur de sortie connu : c'était du poids mort pour ce produit. La migration a pris quelques jours et a produit des résultats mesurables : variabilité du débit fortement réduite, binaires réduits de plus de moitié, arbre de dépendances ramené de quatre-vingts crates à environ vingt-cinq.

La règle que nous en avons tirée s'applique partout : préférer une fondation modeste et comprise à une fondation ambitieuse et opaque. Une dépendance qu'on ne maîtrise pas est une dette de sécurité autant qu'une dette technique.

7. Pourquoi forker Mullvad

Construire un client VPN de qualité production sur trois systèmes de bureau et deux systèmes mobiles représente des années de travail : gestion du kill-switch au niveau du pare-feu, prévention des fuites DNS, routage par politique, interface graphique multiplateforme, intégration aux extensions réseau d'iOS et au service VPN d'Android. Tout cela existe déjà, mûri par des années de production, dans l'application open source de Mullvad.

Notre application de bureau et mobile en est un fork. Nous héritons de la partie difficile et éprouvée (le pilotage du système d'exploitation, la discipline anti-fuite) et nous y remplaçons la brique de tunnel : là où le client d'origine monte un tunnel WireGuard, le nôtre monte un tunnel QUIC WarrenGuard. Nous ne réécrivons pas ce qui fonctionne ; l'effort porte sur notre valeur propre, le transport et le plan de contrôle.

8. Le pari QUIC, et son prix

QUIC n'est pas qu'un transport plus rapide. C'est la primitive dont dérive toute la pile Warren, et il apporte plusieurs propriétés qu'un VPN peut exploiter directement.

Son handshake combine l'établissement de connexion et la négociation TLS 1.3 en un seul aller-retour. Il intègre TLS 1.3 nativement, ce qui donne accès à l'authentification mutuelle par clés publiques brutes (Raw Public Keys, RFC 7250) sans avoir à opérer une infrastructure à clés publiques. Il transporte des datagrammes non fiables (RFC 9221), parfaits pour véhiculer des paquets IP qui n'ont pas besoin d'être réordonnés par le transport. Il sait migrer une connexion d'un réseau à un autre, ce qui permet de passer du Wi-Fi au cellulaire sans rompre la session. Et, surtout, il se confond sur le fil avec HTTP/3.

QUIC seul n'est pas un VPN. Warren ajoute par-dessus une couche applicative propre : un format de handshake, l'identité par wallet, le chiffrement multi-hop, la défense contre l'analyse de trafic, l'obfuscation. QUIC est la fondation sur laquelle tout le reste se construit.

Ce choix a un prix. Une pile QUIC en espace utilisateur chiffre là où WireGuard s'appuie sur le noyau, et QUIC ajoute le chiffrement des numéros de paquet, la protection d'en-tête et la machinerie d'acquittement. C'est le coût du mimétisme HTTP/3, c'est-à-dire de la propriété qui porte tout le reste de la pile. L'ingénierie du datapath le rattrape : le tunnel soutient plusieurs gigabits par seconde sur du matériel adapté (section 10).

À RETENIR · CINQ PROPRIÉTÉS QUE WARREN TIENT DU TRANSPORT

Le choix de QUIC tient à cinq propriétés dont Warren dépend directement, qu'un tunnel WireGuard ne peut pas porter sans qu'on les lui greffe.

- **Furtivité native.** Le trafic se confond avec HTTP/3 sur le port 443, et cette furtivité est une propriété permanente du transport. Un tunnel WireGuard expose au contraire une signature reconnaissable dès le premier paquet, et n'atteint la même discrétion qu'en recevant une couche d'obfuscation séparée à maintenir.
- **Canal de contrôle en bande.** QUIC apporte des flux fiables, une négociation de capacités et un handshake extensible. Le multi-hop scellé, le port forwarding, la défense contre l'analyse de trafic et les jetons de session voyagent dans la même connexion. WireGuard n'offre aucun de ces points d'appui, et chacune de ces fonctions y demanderait un canal séparé.
- **Une seule identité.** TLS 1.3 est intégré au transport : l'authentification mutuelle par clés publiques brutes, la liaison de canal signée, le certificat de couverture et le scellement post-quantique en découlent, tous adossés à la clé de l'utilisateur. Aucune clé propre au tunnel n'est à synchroniser en parallèle.
- **Paquets IP en datagrammes.** Les datagrammes non fiables (RFC 9221) transportent les paquets IP un pour un, sans réordonnancement inutile, tandis que les flux fiables ne portent que le handshake applicatif.
- **Migration de connexion.** Un changement de réseau, du Wi-Fi vers l'Ethernet ou vers le cellulaire, est détecté et la connexion QUIC bascule sur le nouveau chemin sans nouvelle poignée de main, le temps d'un aller-retour de revalidation. L'identifiant de connexion, qui remplace le quadruplet d'adresses, est renouvelé sur le nouveau chemin, donc un observateur ne relie pas les deux chemins par cet identifiant.

Partie III. WarrenGuard, le moteur

Le cœur technique : un seul datapath, en Rust, partagé par toutes les plateformes, et les mécanismes qui le rendent furtif, étanche et difficile à piéger.

9. WarrenGuard : un seul cœur, toutes les plateformes

WarrenGuard est le moteur de Warren : la pile logicielle qui implémente le tunnel VPN sur QUIC, indépendamment de tout ce qui est spécifique au produit Warren. C'est un composant open source (licence AGPL) conçu pour être générique : il ne porte ni politique d'abonnement, ni clé de signature, ni annuaire Warren, et laisse ces décisions au déployeur. Un tiers pourrait l'utiliser pour opérer son propre VPN sur QUIC.

Cette séparation est une loi d'architecture, appliquée strictement. La pile se lit en couches, et les dépendances ne remontent jamais.

DÉTAILS TECHNIQUES · LES QUATRE COUCHES ET LA LOI DE DÉPENDANCE

Le moteur générique (WarrenGuard) est au socle. Au-dessus, deux couches indépendantes l'une de l'autre : le **client Warren** (identité, sélection des serveurs, façade applicative) et le **serveur Warren** (le plan de contrôle, l'application de l'abonnement, les clés privées). Au sommet, l'**application produit**. La règle : le moteur ne connaît rien de Warren ; le client et le serveur dépendent du moteur, jamais l'inverse ; le client et le serveur ne se connaissent jamais directement ; leur seul pont est le réseau. Cette direction de dépendance est structurelle : le moteur vit dans son propre dépôt, se compile seul, et ne référence aucun code Warren, si bien qu'une dépendance inversée ne compilerait pas. Elle garantit que le code sensible, celui qui détient les clés et applique la politique, ne peut pas fuiter dans le code client.

Il existe **une seule implémentation** du datapath, en Rust, réutilisée partout. L'application phare (le fork Mullvad) et la famille de SDK (Rust, Dart, TypeScript) consomment le même moteur natif et le même fork QUIC. Les SDK des autres langages enveloppent ce cœur natif de datapath plutôt que de le réécrire. Une conséquence directe pour la résistance à la censure : toutes les applications Warren, quelle que soit leur origine, émettent exactement la même signature de handshake QUIC. Il n'y a pas une application officielle reconnaissable et des applications tierces distinguables ; il y a un seul comportement sur le fil.

DÉTAILS TECHNIQUES · QUIC COMME TRANSPORT DE VPN : L'USAGE RÉPANDU, ET LE CHOIX DE WARREN

Faire passer un VPN par QUIC est devenu courant, de deux façons distinctes de celle de Warren. La première est l'enrobage : le tunnel reste du WireGuard, enveloppé dans QUIC pour se cacher quand le réseau bloque. Le transport demeure WireGuard, et l'enveloppe se paie en fonctions. Mullvad, qui a ajouté cette obfuscation QUIC en 2025, impose de renoncer au multi-hop et à la défense contre l'analyse de trafic pour l'activer.

La seconde est le relais adossé à un compte. Apple iCloud Private Relay depuis 2021, Cloudflare WARP passé à MASQUE par défaut fin 2024, et Google IP Protection dans Chrome tunnelent nativement sur QUIC, mais restent des relais liés à un compte de plateforme, mono-hop pour WARP, limités au navigateur pour les deux autres, sans identité par clé, sans port forwarding, sans défense contre l'analyse de trafic. Leur handshake MASQUE laisse le nom de domaine lisible dans le premier paquet, que la fragmentation de Warren déjoue (section 12).

Warren fait de QUIC la fondation d'un VPN complet sur un seul datapath : identité non-custodiale par clé, multi-hop scellé à la clé de sortie, port forwarding durci, et défense contre l'analyse de trafic qui s'exécute sur le tunnel QUIC lui-même.

La frontière entre l'ouvert et le fermé suit une ligne simple. Tout le code que l'utilisateur exécute, le moteur et le kit client, est public et auditable par quiconque. Restent en propre le plan de contrôle, les clés de signature et la flotte de serveurs de sortie, c'est-à-dire l'opération du réseau.

10. Le datapath QUIC

Le transport de WarrenGuard repose sur Quinn, l'implémentation Rust de référence de QUIC, dont Warren maintient un fork ciblé, détaillé à la section 11. La présente section décrit le datapath qu'il porte : établissement de session, structure du trafic, contrôle de congestion et gestion du MTU.

Handshake et authentification. Le tunnel utilise TLS 1.3 exclusivement ; TLS 1.2 est refusé. Le 0-RTT est désactivé des deux côtés, délibérément : réutiliser un secret de session permettrait de rejouer des paquets IP tunnelés, une faille inacceptable pour un VPN. Le moteur générique s'authentifie par clés publiques brutes Ed25519, sans certificat X.509 ni chaîne PKI. L'identité de l'utilisateur n'est pas présentée pendant le handshake TLS lui-même : depuis la version 5 du protocole, le client signe le lien cryptographique de canal (une valeur dérivée de la session TLS, RFC 5705) pour prouver la possession de sa clé, et depuis la version 6, le serveur de sortie fait de même en retour. Il n'y a donc pas de demande de certificat client sur le fil, un signal qui trahirait le protocole.

En production, les serveurs de sortie présentent un certificat X.509 valide pour un domaine de couverture anodin, ce qui rend le handshake indistinguable d'une connexion HTTP/3 ordinaire pour un observateur. Un serveur configuré pour un marché censuré refuse de démarrer sans ce certificat plutôt que de retomber silencieusement sur les clés brutes : la posture de furtivité est un invariant de démarrage.

Structure du trafic. Les flux fiables de QUIC ne transportent que le handshake applicatif (la négociation de l'adresse tunnel, du MTU, des fonctionnalités). Les paquets IP de l'utilisateur voyagent dans les datagrammes non fiables de QUIC, un pour un, sans réordonnancement inutile. L'encodage des messages de contrôle est strict : un champ inconnu provoque un rejet, il n'y a pas de compatibilité ascendante approximative. Les versions du protocole évoluent par pas nets, jamais par mutation d'un format existant.

Contrôle de congestion et MTU. Le contrôleur de congestion par défaut est BBR, choisi parce qu'il tient bien mieux que les algorithmes classiques en présence de pertes. Avec deux pour cent de perte injectée sur un flux, un contrôleur classique s'effondre à quelques mégabits par seconde là où BBR maintient plusieurs centaines. Une gestion active de la file d'attente, appliquée par flux sur les datagrammes du tunnel, borne la latence quand la charge monte : plutôt que de laisser un tampon gonfler et le délai grimper, les paquets en excès sont écartés tôt, et la fenêtre d'émission se dimensionne sur le débit réellement disponible. Le MTU du tunnel est initialisé prudemment (1280 octets) pour éviter les trous noirs de fragmentation sur les chemins intercontinentaux, puis sondé à la hausse dynamiquement.

DÉTAILS TECHNIQUES · L'ADAPTATION DYNAMIQUE DU MTU

Un tunnel traverse des chemins de tailles variées, et certains réseaux (Wi-Fi encapsulé, lien satellite, réseau ferroviaire) imposent un MTU réduit sous la valeur habituelle. Warren s'y adapte sur deux couches, sans jamais redimensionner l'interface tunnel, qui reste à 1280 octets.

La première couche est la découverte du MTU du chemin extérieur. Partant du seuil prudent de 1280 octets, elle sonde à la hausse quand le chemin natif le permet et ne descend jamais sous 1200 octets, le minimum que QUIC garantit de bout en bout. Sur le repli TCP, où la pile peut fragmenter d'elle-même, cette découverte est désactivée et le budget interne est plafonné à 1100 octets, la taille qui traverse ce transport sans tomber dans un trou noir.

La seconde couche adapte en direct la place utile offerte aux paquets de l'utilisateur, à partir de ce budget vivant. Pour les flux TCP, elle réécrit l'option MSS des paquets d'ouverture de connexion, de sorte que les deux extrémités se dimensionnent d'elles-mêmes pour tenir dans le tunnel, de façon transparente et sans dépendre d'ICMP. Ce même ajustement est appliqué côté serveur de sortie, ce qui répare les chemins réduits même pour un client qui n'a pas encore été mis à jour. Pour les autres flux, un paquet trop grand est écarté et son émetteur reçoit un vrai message ICMP (« fragmentation nécessaire » en IPv4, « Packet Too Big » en IPv6) portant la bonne taille, pour que sa propre découverte de MTU s'ajuste. L'application signale enfin, à titre purement informatif, quand le MTU effectif descend sous sa valeur nominale ; l'ajustement MSS et le message ICMP synthétique font le travail, que cette indication s'affiche ou non.

Performance mesurée. Sur un serveur bare-metal 10 GbE (processeur AMD EPYC Zen4), entre deux machines d'un même centre de données, un tunnel unique soutient de manière stable de l'ordre de 5,5 à 7 Gbit/s, soit 55 à 70 pour cent du débit de la ligne, sans saturer le processeur d'aucun des deux côtés. C'est la capacité du datapath, et elle dépasse de très loin ce qu'un usage réel demande. Sur le plan de la charge, un serveur tient plus de cinq mille sessions simultanées à moins de dix pour cent de charge processeur : le facteur limitant d'un serveur de sortie est la bande passante de sa liaison, un choix de dimensionnement réseau plutôt qu'une limite de la pile.

11. Le fork de Quinn

QUIC est un protocole vaste, et en réécrire une pile complète serait autant un gouffre d'ingénierie qu'une régression de sécurité. Warren part donc de Quinn, l'implémentation Rust de référence de QUIC, et la fait diverger par un petit nombre de modifications ciblées. La base suit fidèlement le dépôt officiel : le fork est aligné sur la version publiée courante de Quinn et resynchronisé quand l'amont avance, chaque divergence étant isolée en un correctif documenté qu'on peut lire, appliquer ou retirer séparément.

DÉTAILS TECHNIQUES · LE GARDE-FOU ANTI-RÉGRESSION DU FORK

Repasser silencieusement du fork à la version amont ferait perdre le gain de performance et l'obfuscation. Ce risque est neutralisé par un garde-fou de compilation : le code appelle des réglages qui n'existent que dans le fork, si bien qu'un retour à la version amont provoque une erreur de compilation immédiate au lieu d'une régression discrète. Le fork ne peut pas être abandonné par accident, seulement par décision explicite. Chaque nouvelle version du fork est par ailleurs validée par un banc d'essai comparatif A/B avant d'être adoptée.

Les divergences se rangent en trois familles.

Obfuscation du handshake. Le fork ajoute deux réglages qui gouvernent la mise en forme du premier échange : un plancher de remplissage sur le paquet initial, et un plafond sur la taille du premier fragment de la négociation TLS, de sorte que le handshake s'étale sur au moins deux datagrammes UDP. C'est ce qui déjoue l'extracteur de nom de domaine du grand pare-feu chinois (section 12), selon la même idée que la protection anti-ossification du navigateur Chrome. Ces réglages sont inertes par défaut dans l'amont ; le moteur les active.

Performance du datapath. Trois modifications, sans effet sur le protocole visible. L'émission par lots est agrandie, pour pousser davantage de datagrammes par appel système. Les tampons des sockets sont dimensionnés automatiquement à la création, ce qui évite les pertes au-dessus du gigabit que la petite taille par défaut de Linux provoque, et comble au passage une lacune de l'amont sur Windows. Un chemin d'émission par lots propre aux plateformes Apple exploite leurs appels système de traitement groupé.

Contrôle de congestion et latence. C'est la famille la plus substantielle, et elle corrige de vrais défauts. Deux d'entre eux étaient hérités de l'amont dans le contrôleur de congestion BBR : sur une connexion peu sollicitée, ce qui est le cas d'un tunnel au repos, la fenêtre de congestion pouvait croître sans borne, observée à un demi-gigaoctet sur des serveurs de sortie de production. Le premier correctif tient en un seul terme de comparaison, aligné sur l'implémentation de Chromium ; le second rétablit la règle d'admission des mesures de bande passante. Par-dessus, une gestion active de la file d'attente de type FQ-CoDel remplace la file profonde par défaut sur les datagrammes du tunnel : elle borne la latence sous charge et répartit équitablement le débit entre les flux in-

ternes multiplexés dans le tunnel, au prix d'une fraction du débit en conditions idéales. Un dernier réglage ramène la taille du tampon d'émission au produit bande passante-délai réellement mesuré du chemin, au lieu d'une constante calculée pour le pire cas, ce qui évite d'accumuler des secondes de file sur une liaison lente.

Tout n'est pas forké. Plusieurs propriétés qu'on pourrait croire spécifiques à Warren relèvent de l'amont pur, que le moteur se contente de configurer : la neutralisation du spin bit, la désactivation du 0-RTT, la récupération après trou noir de MTU. Le moteur en choisit les réglages sans toucher au code.

Deux de ces correctifs ont vocation à remonter dans l'amont : la fragmentation du paquet initial et la réparation du BBR, tous deux conformes à la spécification et utiles à n'importe quel utilisateur de Quinn. Ils sont préparés pour une proposition en amont. Le reste relève du réglage propre au cas d'un tunnel VPN, que l'amont n'activerait pas par défaut, et reste donc dans le fork.

12. L’obfuscation native

L’obfuscation de premier niveau de Warren gomme les signaux par lesquels un équipement d’inspection reconnaîtrait un protocole de VPN. Elle est active par défaut, pour tout le monde, sans réglage, et son coût en bande passante est faible.

Plusieurs signaux sont neutralisés simultanément. Le protocole applicatif annoncé dans le handshake est uniquement `h3`, celui de HTTP/3. Le nom de serveur (SNI) et la première partie du message d’ouverture TLS sont fragmentés sur au moins deux datagrammes UDP : c’est une défense précise contre un extracteur de SNI documenté du grand pare-feu chinois (travaux présentés à USENIX Security 2025), qui ne réassemble pas un message étalé sur plusieurs datagrammes. Le spin bit de QUIC, qui pourrait servir de fingerprint s’il était constant, est neutralisé. Et un sondeur actif qui tenterait de provoquer le serveur pour l’identifier ne reçoit qu’une fermeture générique du transport, jamais un code d’erreur spécifique à Warren : les octets propres au protocole ne sont émis qu’après un premier message d’établissement valide, qu’un sondeur est incapable de produire.

Ces propriétés sont des invariants testés automatiquement : une modification qui les casserait fait échouer l’intégration continue.

DÉTAILS TECHNIQUES · LE RAISONNEMENT D'INGÉNIERIE, TRACÉ DANS LES DÉCISIONS D'ARCHITECTURE

Chaque choix d’obfuscation est adossé à un document de décision (ADR) qui expose la menace et l’arbitrage. Le modèle d’adversaire retenu inclut explicitement un sondeur actif de classe étatique, pas seulement une inspection passive. Une découverte a été structurante : l’authentification mutuelle TLS classique laissait passer une demande de certificat client sur le fil, invisible à l’inspection passive mais détectable par un sondeur ; la version 5 du protocole l’a supprimée au profit de la signature en bande. Une autre : l’identité par clé publique brute du serveur était elle-même un signal détectable, en amont de toute autre défense, ce qui a motivé le passage à un certificat de couverture X.509 en version 6. La barre visée est précise : indistinguable de HTTP/3 face à un adversaire réaliste combinant inspection profonde, sondage actif et fingerprinting. La parité exacte face à un adversaire global doté d’apprentissage automatique, qui compare octet par octet du trafic à volume de torrent, est l’objet du chantier de fingerprint décrit en section 33.

L’obfuscation ne s’arrête pas à cette première couche. Quand le tunnel est au repos, plutôt qu’un battement régulier de messages de maintien (une cadence qu’un observateur relie facilement), Warren émet un trafic de couverture aux intervalles et aux tailles variés, qui noie ce rythme. Et le certificat de couverture X.509 s’accompagne, sur les serveurs de sortie, d’un service HTTP de leurre : un sondeur qui interroge le serveur comme un site web reçoit une réponse de serveur web ordinaire, au lieu d’une erreur qui trahirait un VPN. Ces défenses tournent sans que l’utilisateur ait à les chercher.

Reste le cas des réseaux qui ne filtrent pas seulement une signature, mais bloquent l’UDP dans son entier : Wi-Fi d’entreprise, portail captif, réseau verrouillé. Un tunnel purement QUIC sur UDP y serait injoignable. Warren bascule alors le **même** tunnel QUIC sur une connexion TCP protégée par TLS 1.3, vers le **même** domaine de couverture et le même port 443. Ce repli est armé par défaut : le client tente le chemin UDP et, si celui-ci ne répond pas très vite, ouvre en parallèle la voie de secours TCP, sans réglage ni longue attente. Sur le fil, la connexion de secours ressemble à une session HTTPS ordinaire, exactement comme le chemin nominal. Là où l’UDP passe, il reste préféré, plus rapide ; là où il est coupé, le tunnel tient quand même.

13. Le multi-hop et le relais aveugle

Le multi-hop fait transiter le trafic par deux serveurs successifs plutôt qu'un, de sorte qu'aucun point unique ne connaisse à la fois qui vous êtes et ce que vous faites. Le premier nœud (le relais) voit votre adresse IP mais pas votre destination ; le second (la sortie) voit votre destination mais pas votre adresse IP.

Nous avons écarté la solution naïve, un tunnel QUIC dans un autre tunnel QUIC, parce qu'elle empile deux contrôles de congestion, deux gestions de fenêtre et deux ajustements de MTU, avec une perte de performance de l'ordre de la moitié. Warren utilise à la place deux connexions QUIC indépendantes, le relais recopiant les datagrammes d'une vers l'autre sans les déchiffrer.

Le relais est **aveugle par construction**. Le client scelle le contenu de son trafic à destination de la clé publique du serveur de sortie, selon le standard de chiffrement hybride HPKE (RFC 9180). Le relais ne détient jamais la clé de déchiffrement du serveur de sortie ; il ne manipule que des octets opaques. Une tentative de détourner un datagramme vers un autre serveur de sortie échoue proprement au déchiffrement, car l'identifiant du serveur de destination est lié cryptographiquement au message : il n'y a pas de « député confus » exploitable.

DÉTAILS TECHNIQUES · DES CLÉS PAR PAQUET SUR UN CANAL NON FIABLE

Le chiffrement HPKE suppose habituellement un flux ordonné et fiable. Or les datagrammes QUIC peuvent se perdre et arriver dans le désordre. Plutôt que de subir la désynchronisation du compteur de séquence, Warren dérive une clé unique par paquet à partir du contexte HPKE, indexée par l'époque et le numéro de séquence, et chiffre avec un nonce fixe : c'est sûr précisément parce que la clé est unique pour chaque paquet, la même technique que QUIC emploie en interne. Les données associées lient chaque datagramme à l'identifiant du serveur de sortie visé, ce qui empêche tout détournement.

Une infrastructure à clés à deux niveaux (une racine hors ligne, une clé opérationnelle) signe l'identité des serveurs, avec une protection contre le retour en arrière de version. Le contexte de scellement d'une session tourne fréquemment, toutes les trente minutes, si bien que la fenêtre pendant laquelle une même clé protège du trafic reste courte par construction.

Le scellement lui-même est **hybride post-quantique**. À l'échange de clés classique X25519 s'ajoute un mécanisme post-quantique (ML-KEM), sans le remplacer : la confidentialité ne peut jamais être moins bonne qu'avec le seul X25519, même si l'un des deux mécanismes se révélait un jour faible. Cette combinaison répond à l'adversaire qui archive aujourd'hui du trafic chiffré pour le déchiffrer le jour où un ordinateur quantique le permettra. Le choix post-quantique est verrouillé par une signature, jamais par un bit non authentifié qu'un équipement intermédiaire pourrait effacer pour forcer un repli, et c'est le comportement par défaut.

14. La défense contre l'analyse de trafic

Chiffrer le contenu ne cache pas la **forme** du trafic. Les tailles de paquets, les intervalles entre eux, les rafales dessinent un fingerprint que des techniques d'apprentissage automatique savent relier à un site visité, même à travers le chiffrement. C'est un angle d'attaque contre lequel un simple tunnel chiffré ne fait rien.

Warren s'en défend avec Maybenot, le framework issu des travaux de l'université de Karlstad et parrainé par Mullvad, qui l'utilise sous le nom de DAITA. Le principe : des machines à états probabilistes qui ajoutent du padding (des paquets factices) et du délai pour brouiller la forme. Cette défense est branchée sur le flux de datagrammes QUIC et négociée à chaque session. Warren embarque un jeu de défenses issues de la littérature (padding à débit constant, bruit aléatoire, et variantes). Quand un client la demande, le serveur de sortie en tire une au sort pour cette connexion, l'annonce au client, puis l'applique sur le trafic qu'il envoie tandis que le client applique la défense annoncée sur le trafic qu'il émet. Comme le tirage est indépendant à chaque connexion, deux clients d'un même serveur de sortie présentent rarement la même forme de trafic, ce qui élargit l'éventail des comportements qu'un observateur voit passer.

Cette défense se paie en bande passante : brouiller la forme du trafic et maximiser le débit sont deux objectifs opposés. Warren en fait donc un choix de l'utilisateur, sur un unique datapath et dans une offre unique : un interrupteur, dans l'application, active la défense contre l'analyse de trafic au prix du débit. L'éteindre conserve toute l'obfuscation de protocole de la section 12, qui, elle, ne coûte presque rien et tourne en permanence. Le seul critère de partage est l'arbitrage entre protection et performance.

À RETENIR · LE SIGNAL DE NÉGOCIATION NE MENT PAS

Une défense contre l'analyse de trafic qui croit tourner sans tourner est pire qu'aucune défense : l'utilisateur adapte son comportement à une protection absente. C'est pourquoi l'activation n'est pas un simple réglage local. Le client demande la défense, le serveur de sortie répond en annonçant la machine qu'il applique, et l'application n'affiche « active » que sur cette confirmation. Si le serveur ne l'accorde pas, l'interface le montre clairement, sur la page dédiée comme sur l'écran de connexion, au lieu de laisser croire à une protection qui ne s'exécute pas.

15. Le port forwarding, durci

Warren restaure le port forwarding via un serveur NAT-PMP complet (RFC 6886) tournant sur le serveur de sortie, accessible uniquement depuis l'intérieur du tunnel. Le contrat est celui promis par la section 2 : usage non restreint, mécanisme durci.

Le durcissement contre l'abus technique est appliqué dans le code. Chaque adresse tunnel est limitée à cinq redirections simultanées. Le rythme d'allocation est plafonné à douze nouveaux ports par fenêtre glissante de soixante secondes. Un port libéré observe une quarantaine de cinq minutes avant qu'un autre client puisse le réutiliser. La durée de vie d'une redirection va de soixante secondes à une heure, renouvelable. Toutes ces limites échouent de manière fermée : une requête au-delà du quota est refusée explicitement. Aucune table persistante ne relie une clé d'utilisateur à un port : les correspondances sont éphémères et n'existent qu'en mémoire.

Le durcissement contre la désanonymisation vise une attaque classique, connue sous le nom de « Port Fail ». Si l'adresse d'entrée et l'adresse de sortie d'un serveur sont identiques, un attaquant abonné au même service peut, via une redirection de port, provoquer une fuite de l'adresse IP réelle d'une victime dont le trafic vers cette adresse contourne le tunnel. La cause racine est que l'exclusion de route vers le serveur VPN est fondée sur la destination.

DÉTAILS TECHNIQUES · CORRIGER AU SOCKET, RESTER EN MONO-IP

Mullvad était immunisé contre Port Fail dès 2010, en séparant l'adresse d'entrée de l'adresse de sortie de ses serveurs. Warren attaque la cause racine plutôt que le symptôme : au lieu d'exclure du tunnel *tout* ce qui va vers l'adresse du serveur, ce qui fait fuir n'importe quel paquet vers cette adresse, l'exception est restreinte **au seul socket du tunnel lui-même**, par des primitives système. C'est l'approche de WireGuard et la recommandation des travaux TunnelCrack (USENIX 2023) ; elle ferme Port Fail et la variante TunnelCrack-ServerIP d'un même geste, sans le coût d'une seconde adresse IP, et Warren reste mono-IP. Comme nous maîtrisons l'intégralité de nos applications, un correctif côté client suffit là où un VPN générique devrait corriger côté serveur, et le serveur de sortie ajoute par ailleurs sa propre barrière en défense en profondeur.

macOS demande un soin particulier, parce que plusieurs interfaces réseau y coexistent souvent (Wi-Fi et Ethernet, par exemple) et que lier un socket à l'une d'elles peut, dans certaines configurations, lui faire perdre toute route de sortie. La liaison du socket y est donc doublée d'une surveillance d'egress : si le socket lié cesse d'émettre réellement, l'application bascule d'elle-même sur un repli par route dédiée, plutôt que de laisser le trafic dans un trou noir. La protection se corrige toute seule au lieu de dépendre d'une configuration réseau idéale.

16. Kill-switch et fail-closed

Un kill-switch empêche le trafic de fuir hors du tunnel quand celui-ci tombe. La leçon architecturale majeure des incidents documentés du secteur (TunnelVision, TunnelCrack) est qu'il ne faut jamais se fier à la table de routage comme mécanisme de sécurité : le fail-closed doit reposer sur un pare-feu qui bloque tout ce qui ne passe pas par le tunnel.

Warren en fait une propriété de construction plutôt qu'une réaction. Tant que le tunnel porte du trafic, le pare-feu de l'OS est en refus par défaut : la règle est posée avant le premier paquet, et rien ne sort du tunnel parce que rien d'autre n'est autorisé à sortir.

La défaillance du logiciel lui-même ne rouvre rien. Les règles de blocage vivent dans le noyau et survivent au processus : un plantage laisse l'hôte muet au lieu de laisser fuir, puis le service se relance de lui-même, redémarre en position bloquée et rétablit le tunnel sans intervention. Et parce qu'une protection qui séquestrerait sa propre machine serait une faute d'ingénierie, l'application garde en toute circonstance une sortie explicite : un clic, sous l'élévation native du système, rétablit l'accès à Internet sans le VPN, pare-feu et DNS restaurés.

Pour qui en a besoin, le **kill-switch persistant** franchit le pas suivant. Une fois armé, le blocage survit à l'arrêt du démon, à son plantage et au redémarrage de la machine : la défaillance a beau être totale, l'hôte reste muet jusqu'à ce que l'utilisateur désarme lui-même la protection. Warren ne l'impose pas par défaut : un blocage persistant imposé transforme le moindre incident logiciel en machine sans réseau, sans recours, y compris pour désinstaller. Côté serveurs de sortie, où aucun humain n'est là pour désarmer quoi que ce soit, l'arbitrage s'inverse : leurs binaires sont compilés de sorte qu'une panne fatale **interrompt** l'exécution sans dérouler le moindre nettoyage.

Sur les systèmes de bureau en mode tunnel intégral, le pare-feu de l'OS (nftables sur Linux, pf sur macOS, l'équivalent sur Windows) est mis en refus par défaut, n'autorisant que le loopback, le périphérique tunnel, et le trafic du démon vers le serveur de sortie. La correction Port Fail tient un cran plus bas, dans le routage : c'est le socket du tunnel lui-même qui est lié, au lieu d'excepter une destination, ce qui couvre le chemin UDP nominal comme le repli TCP de la section 12. Le DNS est poussé dans le tunnel, sans chemin vers le résolveur de l'hôte.

Le niveau de garantie diffère selon le mode, et Warren distingue les deux. Le fail-closed appliqué par le pare-feu de l'OS relève du mode tunnel privilégié. Le mode proxy non privilégié offre un fail-closed d'une autre nature, structurel mais borné à l'application configurée (section 30). Une promesse du type « zéro paquet hors tunnel » n'a de sens que sous un niveau de garantie précis : certains systèmes d'exploitation s'exemptent eux-mêmes des pare-feux applicatifs.

17. Sécurité mémoire et hygiène des secrets

Le moteur est écrit en Rust avec l'interdiction du code `unsafe` posée au niveau de l'atelier de compilation. Trois crates seulement relâchent cette interdiction, chacune pour une raison précise et documentée : l'appel système de liaison de socket, l'interface avec l'API réseau de Windows, l'ouverture du périphérique tunnel privilégié. Partout ailleurs, le code est sans `unsafe`. Cette discipline élimine par construction des classes entières de vulnérabilités mémoire.

La cryptographie par défaut est entièrement en Rust (bibliothèque `ring`), sans chaîne de compilation C dans le profil courant. Les matériaux secrets (graines, clés de signature, phrases de récupération) sont effacés de la mémoire à leur libération, et aucun type portant un secret n'expose de représentation de débogage qui le révélerait : au mieux l'adresse publique. La discipline no-log est appliquée jusqu'aux messages d'erreur et aux journaux : jamais une clé publique complète, une adresse IP source, un nonce ou une graine en clair, de part et d'autre de la frontière entre langages. Un identifiant, si un fragment est vraiment nécessaire au diagnostic, est tronqué.

18. Le wire format comme source de vérité

Le protocole client de Warren est implémenté une fois en Rust, puis réexposé aux autres langages via des SDK. Pour qu'un kit TypeScript ou Dart parle exactement le même langage que les serveurs de production, la compatibilité binaire doit être stricte, octet pour octet.

Nous garantissons cela par des **golden vectors** : un ensemble de fichiers de référence partagés qui figent chaque format (dérivation d'identité, adresse, signature de requête, trames de handshake, trame multi-hop, annuaire signé) au bit près. Chaque SDK rejoue ces mêmes fichiers. Un format est ancré par des octets exacts plutôt que décrit par une prose sujette à interprétation. Modifier un vecteur signifie changer le wire format, ce qui impose de monter une version de schéma, jamais de muter l'existant. On ne modifie jamais un vecteur pour faire passer un test : une divergence de vecteur est une vraie régression de protocole. Cette rigueur est ce qui permet à plusieurs implémentations de coexister sans jamais dériver, et donc à toutes les applications de présenter la même signature sur le fil.

Le versionnement du protocole et la négociation de capacités suivent la même logique. Les fonctionnalités sont annoncées par un masque de bits, les versions cohabitent proprement le temps d'une migration, et un nœud qui ne comprend pas une nouvelle version la refuse nettement plutôt que de tenter une interprétation approximative.

Partie IV. Le réseau

Ce qui tourne en face : des serveurs sans rien à saisir, un relais aveugle, et un plan de contrôle qui en sait le moins possible.

19. Les serveurs de sortie : rien à saisir

Un serveur de sortie est l'endroit le plus sensible de tout le système : c'est là que le trafic redevient clair et que coexistent, le temps d'une session, l'adresse IP réelle de l'utilisateur et sa destination. Toute la conception vise à ce qu'une saisie de cette machine ne rende rien.

Chaque serveur de sortie de production tourne sur une image Alpine Linux immuable, installée par un outil qui prend le contrôle d'un serveur nu et le reconstruit en un système en lecture seule. Le système de fichiers racine est une image compressée en lecture seule (squashfs), superposée à une couche d'écriture (un overlay) qui n'existe qu'en mémoire vive. Le système en cours d'exécution est jetable : un redémarrage l'efface intégralement.

À RETENIR · CE QUI PERSISTE, ET CE QUI NE PERSISTE PAS

Il n'y a **aucune partition de données**. La seule chose qui doit survivre à un redémarrage est l'identité du nœud (sa clé de signature), et elle survit non par un disque inscriptible mais parce qu'elle est gravée en lecture seule dans l'image et regravée à l'identique à chaque mise à jour. Tout le reste (sessions actives, allocations de ports, cache DNS, journaux) vit dans la couche mémoire et disparaît au redémarrage. Une machine éteinte puis saisie rend le système d'exploitation en lecture seule et sa propre clé d'identité, et rien sur les utilisateurs. Une machine allumée et saisie rend, au pire, les sessions vivantes présentes en mémoire à l'instant de la saisie, jamais un historique. Le noyau n'écrit aucun core dump : à l'arrêt d'un processus sur erreur, le système sauvegarde normalement toute sa mémoire dans un fichier, qui contiendrait ici les adresses des utilisateurs et leurs sessions ; cette écriture est désactivée. Les journaux des services, eux, vivent en mémoire vive et ne touchent jamais le disque.

Le déploiement des mises à jour suit la même logique. Le système utilise deux slots racine (A et B). Une mise à jour se fait sans interruption ni redémarrage, en deux temps : d'abord le remplacement à chaud du binaire pour le processus en cours, ensuite la reconstruction du slot inactif pour qu'il serve la même version au prochain démarrage. Un mécanisme de secours dans l'initramfs bascule automatiquement sur l'ancien slot si un nouveau échoue à démarrer. Cette immuabilité a un corollaire précieux pour la vérifiabilité : ce qui tourne est exactement ce qui a été construit, et toute altération suppose de reconstruire une image.

20. Anatomie d'un serveur de sortie

Le plan VPN d'un serveur de production tient sur le seul port **443** : le socket UDP qui porte QUIC, et le socket TCP qui termine le repli de la section 12. Derrière eux tourne un répartiteur unifié qui est à la fois le relais multi-hop et le terminateur de sortie. Chaque connexion QUIC entrante porte, en clair, une étiquette de routage indiquant le serveur de sortie visé : si c'est le nœud lui-même, il termine la connexion localement ; sinon, il la retransmet en aveugle vers un autre nœud. Cette conception replie le simple saut et le double saut sur un même datapath, dans tous les modes. Sur ce port, un serveur de sortie ressemble à un serveur HTTP/3 : le 443, le protocole `h3`, un nom de serveur plausible.

Plusieurs mécanismes de sécurité tiennent dans des détails qui font ou défont un VPN no-log.

Traduction d'adresses anti-corrélation. Le serveur masque les adresses des utilisateurs derrière la sienne, et il le fait avec une allocation de port source **entièrement aléatoire**. Le noyau ne préserve pas le port source éphémère choisi par le client dans le quintuplet de sortie : un client qui réutiliserait un port source fixe ne pourrait pas faire émerger un port de sortie stable qu'un observateur relierait à ses sessions.

Résolveur DNS sans journal. Le serveur de sortie ne journalise aucune requête DNS, par construction. Un résolveur récursif tourne comme service séparé, résolvant lui-même les noms avec minimisation de requête (aucun tiers ne voit la requête complète), son cache épinglé en mémoire vive de sorte que les noms résolus ne touchent jamais le disque, sans module de journalisation, les core dumps désactivés.

Modèle de privilèges. Le serveur de sortie tourne en utilisateur non privilégié. Les capacités système dont il a besoin (administrer le réseau, se lier au port 443) lui sont accordées comme un jeu de capacités ambiantes (*ambient capabilities*) après abandon des privilèges root, jamais comme des capacités attachées au fichier binaire.

DÉTAILS TECHNIQUES · POURQUOI LES CAPACITÉS AMBIANTES, PAS LES CAPACITÉS DE FICHIER

Le noyau efface le jeu de capacités ambiantes lors de l'exécution d'un binaire qui porte des capacités de fichier ; et sur le système de fichiers superposé en lecture seule, les capacités de fichier s'appliquent de manière peu fiable. Attacher des capacités au binaire cassait donc la liaison au port 443 de façon intermittente. Le modèle de capacités ambiantes, transmis à travers l'abandon de privilèges, est à la fois plus sûr (le binaire ne porte aucune capacité au repos) et plus robuste sur ce type d'image.

21. Le relais aveugle et l’annuaire signé

Le rôle de relais (le premier saut du multi-hop) consiste à transmettre du texte chiffré opaque du client vers le serveur de sortie choisi, sans jamais détenir la clé de déchiffrement de ce dernier. Ce que le relais peut voir : l’adresse IP source du client (il en est le pair réseau), l’étiquette de routage en clair, et des octets opaques. Ce qu’il ne peut pas voir : le texte clair entre le client et le serveur de sortie, la destination, ou le trafic déchiffré. Le sceau HPKE est la vraie frontière de sécurité.

Un nœud apprend l’existence des autres nœuds de la flotte par un **annuaire signé**, récupéré sur un point d’accès authentifié réservé aux serveurs de sortie enregistrés. Chaque entrée d’annuaire est vérifiée contre la clé opérationnelle avant qu’un relais accepte de composer vers elle : un plan de contrôle compromis ne peut ni injecter un serveur de sortie malveillant, ni laisser une simple clé d’abonné énumérer les adresses des serveurs de sortie.

La liste des serveurs publiée aux applications suit le même principe de minimisation. Elle est signée, et elle ne révèle que ce qu’une application a besoin pour choisir un serveur et le composer : l’identifiant du nœud, sa localisation (pays et ville, ce que l’utilisateur voit dans la liste), un poids de sélection, ses points d’entrée et les capacités qu’il annonce. Elle a été délibérément amaigrie pour ne plus contenir les rôles des nœuds ni **l’adresse de sortie**, autant de leviers qu’un censeur utiliserait pour énumérer et bloquer la flotte. Une application apprend par où entrer, jamais par où l’on sort.

22. Le plan de contrôle

Le plan de contrôle est le service HTTP unique auquel parlent les serveurs de sortie et les applications. Il gère les abonnements, les vouchers anonymes, les paiements, le registre des serveurs de sortie et leurs heartbeats, l'annuaire multi-hop, l'émission des jetons de session anonymes. Il n'est jamais exposé directement : il est protégé par un reverse proxy.

L'asymétrie de connaissance entre le plan de contrôle et les serveurs de sortie est le cœur du modèle de confidentialité.

Le plan de contrôle sait : quelle clé publique est abonnée, et jusqu'à quand. C'est un fait d'authentification inévitable, mais sans aucun historique d'activité. Il connaît les références comptables des paiements, le registre des serveurs de sortie, des statistiques agrégées. Il ne sait pas quel serveur de sortie un abonné a utilisé, ni ses destinations.

Un serveur de sortie sait : les sessions vivantes en mémoire, les allocations d'adresses tunnel, le trafic qu'il fait sortir. Rien de tout cela n'est écrit sur disque.

Le relais, enfin, ne voit que l'adresse IP du client et des octets opaques.

Aucun composant, seul, ne détient la chaîne complète reliant une identité à une activité. C'est la propriété que la section 29 pousse à sa conclusion avec les crédentiels de session anonymes.

23. Le no-log, vérifié par la machine

Côté serveur, le no-log est un ensemble de contrôles automatiques.

Chaque service porte un test qui parcourt son propre code source et **fait échouer l'intégration continue** si une ligne de journalisation venait à interpoler une clé publique complète, une adresse IP client, un nonce ou un secret. Le no-log est ainsi gardé par l'outillage, pas par la vigilance humaine.

Le reverse proxy filtre ses journaux d'accès pour en retirer l'adresse IP du client et l'ensemble des en-têtes, ne gardant que la méthode, l'URL, le statut et la durée ; il désactive entièrement la journalisation pour les URL susceptibles de contenir un secret. Là où un service n'a pas besoin de l'adresse du client, l'en-tête d'adresse transmise est épinglé à une valeur nulle, de sorte qu'aucun composant en aval ne l'apprenne ; là où l'API en a besoin pour plafonner le débit, elle la garde en mémoire, jamais journalisée.

Ce qui est stocké dans la base de données est inventorié dans une matrice de rétention, elle-même gardée par un test d'intégration continue : une nouvelle table durable ne peut pas être livrée sans une ligne documentant ce qu'elle contient, pourquoi, et pour combien de temps. Les données délibérément jamais stockées sont explicites : trafic, DNS, connexions, adresses IP source, bande passante par utilisateur.

NOTE · DEUX TABLES, ZÉRO LIGNE

Le schéma contient deux tables capables de journaliser de l'activité : les migrations les créent, elles sont donc présentes dans la base de production. Leur écriture est derrière un interrupteur désactivé, et cet interrupteur n'est pas armé. Un auditeur qui ouvrirait la base verrait deux tables vides. C'est une propriété vérifiable, et c'est la forme la plus forte que puisse prendre un no-log : non pas une donnée qu'on promet de ne pas exploiter, mais une donnée qui n'est jamais écrite.

24. Opérer la flotte sans casser le no-log

Mettre à jour une flotte de serveurs de sortie sans coupure ni fenêtre de vulnérabilité est un problème d'ingénierie à part entière. Warren le résout par un plan de contrôle de déploiement dont la propriété de sécurité tient dans la chaîne de signature : le catalogue des versions est signé hors ligne, avec une protection contre le retour en arrière (un compteur monotone) et contre le rejeu (une date d'expiration).

Le déploiement suit une machine à états par nœud (drainer, basculer, vérifier, remettre en service) et une règle stricte : un seul serveur canary, le moins chargé, passe en premier ; son état de santé est comparé au reste de la flotte laissé intact, et l'analyseur qui rend le verdict est séparé de l'actionneur qui applique. Ce n'est qu'une fois le canary validé que le reste suit. Le drainage est un vrai « make-before-break » : le serveur en cours de retrait signale son état dans la bande, les applications l'excluent de leur sélection, et le superviseur de l'application établit une nouvelle session avant de fermer l'ancienne. Une mise à jour n'interrompt pas les utilisateurs qui y sont connectés.

25. Le blocage DNS : opt-in, uniforme, public, sans détournement

Warren propose un blocage de contenu au niveau du DNS (publicités, traqueurs, logiciels malveillants, et catégories optionnelles). Sa conception le tient à distance de tout ce qui ferait d'un filtre un outil de surveillance ou de censure.

Il est **opt-in** et désactivé par défaut. L'utilisateur choisit les catégories, et son choix voyage encodé dans l'adresse du résolveur qu'il utilise dans le tunnel, sans que le serveur ait à tenir un profil par utilisateur. Il est **uniforme et public** : les listes de blocage sont identiques pour tous et publiées dans un dépôt ouvert, et aucun ciblage par utilisateur n'est même exprimable dans le système, faute d'une quelconque dimension utilisateur dans le résolveur. Un changement de liste imposé sous contrainte apparaîtrait dans l'historique public du dépôt. Il est **sans détournement** : un domaine bloqué reçoit une réponse « ce domaine n'existe pas » (NXDOMAIN), et non une redirection silencieuse vers un serveur qui observerait la tentative. Et il est **sans journal**, comme tout le reste.

Partie V. Identité, paiement et applications

Comment prouver un abonnement sans compte, payer sans se lier à son usage, et ce que les applications garantissent réellement sur chaque système.

26. L’identité : un wallet, pas un compte

L’identité d’un utilisateur Warren est une paire de clés Ed25519 qu’il génère lui-même sur son appareil, à partir d’une phrase de récupération de douze mots au standard BIP39 (le *mnemonic*, 128 bits d’entropie). La clé publique, encodée en une adresse lisible qui commence par `wb`, est son identifiant. La clé privée ne quitte jamais l’appareil, où elle est conservée chiffrée au repos par le magasin de secrets du système d’exploitation : trousseau sur macOS et iOS, sans synchronisation vers le cloud ; DPAPI sur Windows ; magasin de clés matériel sur Android ; clé scellée à la machine sur Linux.

Il n’y a ni compte, ni adresse e-mail, ni mot de passe. Ce modèle est **non-custodial** au sens cryptographique du terme, une propriété que ni Mullvad, ni IVPN, ni Proton n’offrent à ce jour.

EN BREF · CE QUE « NON-CUSTODIAL » VEUT VRAIMENT DIRE

Prenez le numéro de compte à seize chiffres de Mullvad, qui est pourtant ce que le marché fait de mieux en matière de compte sans e-mail : il est généré par le serveur, stocké par lui, et comparé à chaque connexion. C’est, cryptographiquement, un bearer token que le fournisseur détient, et un vol de sa base exposerait tous les comptes en clair. Chez Warren, la paire de clés est générée localement, la clé privée ne quitte jamais l’appareil, et le serveur ne voit que des clés publiques (inutiles seules) et des signatures à usage unique. Un vol de notre infrastructure ne donne accès à aucun compte, une signature ne peut être rejouée, et l’utilisateur prouve son identité sans jamais révéler son secret. La contrepartie est qu’il n’y a aucune récupération côté serveur : perdre les douze mots, c’est perdre l’abonnement. Personne d’autre ne les détient, pas même nous.

L’adresse `wb` emprunte un format d’encodage au monde des chaînes de blocs (le format SS58), mais il n’y a **aucune chaîne de blocs** dans Warren, aucun contrat intelligent, aucun abonnement inscrit sur un registre distribué. Le composant que nous appelons « contract » est un contrat au sens logiciel, celui du format d’échange entre le client et le serveur ; c’est une bibliothèque, pas un jeton. Les abonnements vivent dans une base de données classique, hors chaîne.

27. Prouver son abonnement sans compte

Puisqu'il n'y a pas de session ni de bearer token, comment un client prouve-t-il au serveur qu'il a le droit d'agir ? Par une signature. Chaque requête au plan de contrôle porte quatre en-têtes : la clé publique, une signature Ed25519, un horodatage et un nonce. Le message signé couvre la méthode, le chemin, l'horodatage, le nonce et une empreinte du corps ; il ne contient pas la clé publique elle-même. Le serveur vérifie la signature, refuse un horodatage hors d'une fenêtre de soixante secondes, et rejette un nonce déjà vu grâce à une table anti-rejeu.

Un vol de la base de données donne, au mieux, l'ensemble des clés publiques abonnées et leur date d'expiration : aucune adresse e-mail, aucun nom, aucune adresse IP, aucun journal, aucune série temporelle d'activité par compte. Rien ne permet de reconstruire une clé privée, puisqu'aucune ne transite jamais par le serveur.

Le contrôle du nombre d'appareils n'est pas une liaison durable mais une limite de simultanéité : un plafond d'appareils **connectés en même temps** par abonnement, appliqué via un registre de baux à durée de vie courte qui se régénèrent tout seuls. Rien n'est durablement enregistré ; réinstaller ne coûte pas de place. Un identifiant d'appareil aléatoire, en mémoire seulement, est tiré à chaque exécution.

28. Le paiement, décorrélé de l'usage

Le paiement est le point où l'anonymat d'un VPN se gagne ou se perd, parce que payer suppose souvent de révéler une identité réelle (une carte bancaire, un e-mail). Warren découple le paiement de l'usage par un sas.

Toute la vérification des paiements est **auto-hébergée** : nous vérifions nous-mêmes la signature de chaque événement de paiement, sans passer par un intermédiaire tiers qui verrait à la fois le paiement et l'identité. Chaque fournisseur (carte, chaînes de blocs, magasins mobiles) est réduit à un événement neutre ne portant qu'un montant, une devise et une référence opaque : c'est un type de données dont l'absence de donnée personnelle est vérifiée à la compilation.

Le mécanisme du sas repose sur un voucher anonyme. Le paiement produit un voucher dont seul le hash est stocké ; ce voucher est ensuite échangé contre une clé publique, l'échange lui-même ne demandant aucune authentification puisque le voucher est le seul justificatif. Au moment de payer, l'application tire un identifiant d'achat aléatoire ; le fournisseur de paiement ne voit que cet identifiant, jamais la clé publique de l'utilisateur ; le site de paiement ne voit jamais la clé publique ; et le plan de contrôle n'apprend la clé publique qu'à l'instant de l'échange du voucher.

DÉTAILS TECHNIQUES · LA JOINTURE AU REPOS, BORNÉE ET DURCIE

Une jointure au repos relie une référence de paiement à une clé publique, pour qu'un remboursement ou une contestation puisse effectivement couper l'accès (sans ce lien, un voucher remboursé resterait utilisable). Elle est durcie de trois façons. Elle est chiffrée au repos sous une clé qui n'est **pas dans la base de données**, si bien qu'un vol brut de la base ne permet pas de la lire. La ligne est supprimée après vingt jours (quatre-vingt-dix pour les paiements mobiles). Les horodatages sont tronqués au jour et l'expiration arrondie au minuit suivant, de sorte qu'aucune durée ne révèle la seconde d'un échange. Combinée à l'absence totale de journal d'adresse IP, cette jointure ne donne prise à aucune corrélation exploitable dans les conditions normales d'exploitation.

La carte bancaire est en service, et les canaux à plus forte décorrélation, Bitcoin, Lightning et Monero, s'appuient sur ce même voucher anonyme : c'est le sas, et non le moyen de paiement, qui porte la propriété d'anonymat. Le renouvellement automatique est piloté par le client, la carte restant chez le fournisseur de paiement et jamais chez nous.

29. Les crédentiels de session anonymes

Le sas de paiement empêche de relier une identité réelle à une clé publique. Reste une dernière jointure à briser : celle entre la clé publique abonnée et l'usage du tunnel. Tant que le serveur de sortie voit la clé publique de l'utilisateur au moment de la connexion, un lien existe, même s'il n'est pas journalisé.

Warren le brise avec des crédentiels de session anonymes fondés sur Privacy Pass (jetons à signature aveugle RSA, standards RFC 9578 et RFC 9474). Le principe : le plan de contrôle émet des jetons à une clé publique abonnée, mais ces jetons sont aveuglés, de sorte que le jeton finalisé que l'utilisateur dépense est cryptographiquement impossible à relier à l'émission. Les clés d'émission changent à chaque heure, dérivées d'une graine, ce qui garantit qu'un jeton n'est dépensable que dans l'heure pour laquelle il a été signé.

Aucun composant de Warren ne peut produire un enregistrement reliant une clé publique abonnée à un serveur de sortie, une session ou une destination. Le plan de contrôle ne voit que l'émission, authentifiée par le wallet. Le serveur de sortie ne voit qu'un jeton anonyme et un numéro de série à usage unique. Le relais ne voit que l'adresse IP et du texte chiffré.

Ce mécanisme est en production sur toutes les plateformes : l'émetteur de jetons tourne, les serveurs de sortie acceptent le protocole, et les applications de bureau comme mobiles ouvrent leurs sessions avec des jetons anonymes. Les jetons sont frappés à l'avance, en tâche de fond et à cadence fixe, jamais au moment de la connexion. L'application entretient ainsi une réserve locale de jetons valides pour les heures à venir ; les clés d'émission changeant chaque heure, elle en garde quelques-uns par tranche horaire. Si le calendrier d'émission suivait celui des connexions, il révélerait un lien entre l'abonnement et l'usage ; c'est pourquoi les deux sont découplés. Et si la réserve se trouve épuisée, par exemple après une longue période hors ligne, la session s'ouvre par la signature du wallet (section 27) au lieu d'échouer : la connexion aboutit toujours, la disponibilité ne dépend jamais du stock de jetons.

30. Les applications : deux chemins, une même exigence de vérité

Warren s'exécute sur macOS, Windows et Linux (application de bureau, fork Mullvad), sur Android et iOS (applications natives), en ligne de commande pour les serveurs sans interface, et en extension de navigateur. Sous toutes ces surfaces, un seul cœur natif.

Deux chemins de données coexistent, avec des propriétés de sécurité différentes.

Le premier est un **datapath proxy non privilégié**. Une pile TCP/IP en espace utilisateur, exposée localement comme un proxy SOCKS5 ou HTTP CONNECT, termine les flux de l'application locale et les pousse comme datagrammes QUIC vers le serveur de sortie. Il ne demande aucun privilège sur aucun système. Sa propriété de fail-closed est structurelle, mais d'une nature particulière.

À RETENIR · LE FAIL-CLOSED COMME PROPRIÉTÉ DE CYCLE DE VIE

Le port du proxy local n'existe **que tant que le tunnel est monté**. L'application est pointée vers ce port. Si le tunnel tombe, le port meurt, et les connexions de l'application échouent au lieu de retomber sur la route directe. Aucune règle de pare-feu à installer à temps : c'est une propriété du cycle de vie, pas de tunnel, pas de port, pas de fuite. Cette garantie vaut par application, pour l'application configurée : une application non configurée, ou un composant qui contournerait le proxy, peut fuir. C'est un niveau de garantie distinct du fail-closed intégral du pare-feu.

Le second est le **chemin tunnel système**, privilégié, qui capture tout le trafic de l'appareil via un vrai périphérique tunnel avec routage par défaut scindé, poussée du DNS et kill-switch au niveau du pare-feu de l'OS. C'est le chemin de l'application de bureau et des applications mobiles. Sa garantie de fail-closed est celle du pare-feu (section 16), appliquée par le système.

Le DNS ne fuit dans aucun des deux modes : en mode proxy, les résolutions se font au serveur de sortie dans le tunnel, sans passer par le résolveur de l'hôte ; en mode tunnel, la politique du pare-feu n'autorise le port 53 que vers le résolveur du tunnel, dont l'adresse n'est routable que par le tunnel.

« Connecté » n'est pas « protégé ». La distinction vient d'un cas concret : un serveur de sortie en cours de retrait peut continuer à répondre aux paquets de maintien du tunnel, si bien que la connexion QUIC paraît parfaitement vivante alors qu'il ne fait plus rien passer. L'interface afficherait « connecté » alors que l'utilisateur n'a plus aucun accès à Internet.

Surveiller le transport ne suffit donc pas ; il faut exercer le datapath de bout en bout. L'application ouvre périodiquement une vraie connexion **à travers** le tunnel, vers une cible extérieure : elle ne peut aboutir que si le datapath transporte réellement des octets d'un bout à l'autre. Quand elle échoue plusieurs fois de suite, la session est déclarée morte et l'application se reconnecte ailleurs.

C’est un principe qui gouverne toute la conception des applications Warren : l’état affiché découle du trafic qui passe réellement, mesuré, et non d’un drapeau que le logiciel se contente de positionner.

31. Le forum, sans e-mail ni adresse IP

Le forum communautaire de Warren pousse la même logique jusqu'à l'authentification. On y prouve son identité par une signature du wallet, exactement la même requête canonique que partout ailleurs, le navigateur ne touchant jamais la clé. Le forum reçoit un identifiant apparié, un e-mail synthétique, et jamais de mot de passe, d'e-mail réel ou d'adresse IP client. Le nom d'utilisateur public est une chaîne prononçable dérivée d'une empreinte de la clé, déterministe pour que le support puisse recalculer le lien mais non inversible et non corrélable à l'adresse publique. La colonne qui contenait la clé publique en clair a d'ailleurs été retirée du schéma.

Partie VI. Sécurité et limites

Ce que Warren protège, ce qu'il ne protège pas, et qui contrôle quoi.

32. Modèle de menace synthétique

Un système de sécurité se juge à ce qu'il protège, à ce qu'il protège partiellement, et à ce qu'il ne protège pas. Voici la synthèse pour Warren.

Garanties acquises. Votre fournisseur d'accès et les réseaux que vous traversez ne voient qu'un trafic confondu avec du HTTP/3 vers un serveur de sortie, jamais vos destinations. Les sites que vous visitez voient l'adresse du serveur de sortie, pas la vôtre. Un vol de la base de données du plan de contrôle ne donne aucun accès et aucun historique. Une saisie physique d'un serveur de sortie ne rend rien sur les utilisateurs. Un relais multi-hop est cryptographiquement aveugle au contenu. Les sessions s'ouvrent avec des jetons anonymes, sur toutes les plateformes : aucun composant ne peut relier votre abonnement à votre usage. Le no-log côté serveur est gardé par des tests qui bloquent l'intégration continue.

Garanties partielles ou conditionnelles. L'obfuscation défait l'inspection profonde et le sondage actif réalistes, mais pas encore la parité de fingerprint exacte face à un adversaire qui compare octet par octet le trafic à celui d'un vrai navigateur (section 33). La défense contre l'analyse de trafic protège la forme du trafic quand l'utilisateur l'active ; éteinte, la protection porte sur le contenu et la reconnaissabilité du protocole, pas sur la forme. Le no-log est garanti par l'architecture, le code ouvert et des tests qui gardent chaque service.

Hors de portée, par nature. Warren ne protège pas contre un adversaire passif global capable d'observer les deux extrémités et de corréler par le temps. Il ne protège pas contre un logiciel malveillant sur votre appareil. Il ne protège pas contre le fingerprinting de votre navigateur (le VPN masque l'adresse IP, pas le fingerprint du navigateur). Il ne protège pas contre une coalition qui contrôlerait à la fois le relais et le serveur de sortie de votre circuit.

Reste la question de qui contrôle quoi. Sont **contrôlés centralement** : le plan de contrôle, la clé de signature des versions (conservée hors ligne) et le certificat de couverture. Sont **réellement décentralisés** : l'identité de l'utilisateur (non-custodiale, sur son appareil), l'identité de chaque serveur de sortie, et le datapath en mémoire vive des serveurs de sortie.

À RETENIR · LE MODÈLE DE MENACE EN TROIS PHRASES

Warren garantit que la donnée sensible n'existe pas là où on pourrait la saisir, et que le trafic ne se laisse pas trivialement identifier ni bloquer. Il ne garantit pas l'anonymat absolu face à un adversaire qui observe à la fois votre accès à Internet et la sortie du réseau et recoupe les deux, ni face à un logiciel qui contrôle votre appareil. Aucun VPN ne le peut, et un VPN qui le prétend ment.

33. Trajectoire

Warren est en production, et son architecture a été taillée pour absorber de nouvelles briques sans être réécrite. Une pile de confidentialité qui cesse d'avancer est une pile qui se fait rattraper : voici où porte l'effort.

La parité de fingerprint. Les serveurs de sortie présentent un certificat de couverture ordinaire, le handshake est fragmenté pour déjouer les extracteurs de nom de domaine, et l'empreinte du tunnel est surveillée en intégration continue, à chaque commit, pour interdire toute dérive. L'objectif suivant est le plus exigeant du domaine : rendre l'empreinte QUIC de Warren **identique octet pour octet** à celle d'un navigateur grand public. Aucun fournisseur ne l'a atteinte, et l'outillage nécessaire n'existe pas encore en Rust : nous le construisons.

L'attestation de l'image des serveurs. L'image des serveurs de sortie est immuable (section 19). Nous travaillons à en faire une preuve opposable : une signature de son empreinte, contrôlée au démarrage, ferait refuser toute image non officielle. C'est elle qui conditionnera l'ouverture du réseau à des opérateurs tiers : sans attestation, « décentralisé » voudrait dire « des serveurs inconnus qu'on ne peut pas vérifier », un recul déguisé en progrès. Nous préférons décentraliser quand ce sera vérifiable.

Conclusion

Warren est le résultat d'une suite de convictions tenues jusqu'à leurs conséquences techniques. Le no-log est une propriété de construction : de là viennent l'identité non-custodiale, les serveurs de sortie en mémoire vive, les crédentiels de session anonymes, le no-log gardé par l'outillage. La résistance à la censure est native, portée par le transport lui-même : de là viennent le choix de QUIC comme fondation et l'obfuscation permanente. Servir l'utilisateur prime sur se protéger de lui : de là vient le port forwarding restauré et durci, là où Mullvad et IVPN l'ont supprimé en 2023.

Ces propriétés ont un coût. Le mimétisme HTTP/3 demande une pile QUIC en espace utilisateur, et nous l'avons écrite. La minimisation des données demande une infrastructure conçue pour ne rien retenir, et nous l'avons bâtie. Ce qui reste devant nous, la parité de fingerprint parfaite, l'architecture l'attend sans devoir être refaite.

Le code fait foi. Le moteur et le kit client sont ouverts et auditables, et les propriétés décrites ici y sont vérifiables directement. C'est là que se juge ce que ce document affirme.

Annexe A. Glossaire

BBR : algorithme de contrôle de congestion fondé sur l'estimation de la bande passante et du délai, robuste aux pertes.

BIP39 : standard de représentation d'une graine cryptographique sous forme d'une liste de mots mémorisables.

DAITA : défense contre l'analyse de trafic par ajout de padding et de délai, brouillant la forme du trafic.

Datapath : le chemin que suivent les paquets de l'utilisateur, du périphérique tunnel jusqu'au serveur de sortie. À distinguer du plan de contrôle, qui gère les abonnements et la découverte des serveurs.

DPI (Deep Packet Inspection) : inspection du contenu des paquets par un équipement réseau, pour identifier ou bloquer un protocole.

Ed25519 : schéma de signature à courbe elliptique, rapide et sûr, utilisé pour l'identité Warren.

Fail-closed : comportement d'un système qui, en cas de défaillance, bloque plutôt que de laisser passer.

Fingerprint : signature observable d'un protocole ou d'un logiciel (tailles de paquets, ordre des extensions TLS, cadence), qui permet de l'identifier même chiffré.

Golden vectors : fichiers de référence figeant un format binaire au bit près, rejoués par chaque implémentation pour garantir qu'elles parlent exactement le même protocole.

Handshake : échange initial qui établit une connexion, négocie le chiffrement et authentifie les parties.

Heartbeat : message périodique par lequel un serveur signale au plan de contrôle qu'il est vivant.

HPKE (RFC 9180) : chiffrement hybride à clé publique, utilisé pour sceller le trafic multi-hop de bout en bout.

Kill-switch : mécanisme qui empêche toute fuite de trafic hors du tunnel quand celui-ci tombe.

Leurre (*decoy*) : faux service (ici un vrai serveur HTTP/3) présenté à un sondeur pour qu'il ne distingue pas le VPN d'un site ordinaire.

MTU : taille maximale d'un paquet sur un chemin réseau donné.

NAT-PMP (RFC 6886) : protocole d'ouverture de port à travers une traduction d'adresses, utilisé pour le port forwarding.

No-log : absence de journalisation de l'activité des utilisateurs.

Non-custodial : modèle où le secret d'identité est détenu par l'utilisateur seul, jamais par le fournisseur.

Overlay : couche d'écriture superposée à un système de fichiers en lecture seule. Chez Warren elle n'existe qu'en mémoire vive, donc rien ne persiste au redémarrage.

Padding : octets de bourrage ajoutés à un flux pour masquer la taille réelle des données.

Privacy Pass (RFC 9578, RFC 9474) : jetons à signature aveugle permettant de prouver un droit sans être identifiable.

QUIC : protocole de transport moderne sur UDP, fondation de HTTP/3, intégrant TLS 1.3.

Quinn : implémentation Rust de référence de QUIC, dont Warren maintient un fork.

Raw Public Key (RFC 7250) : authentification TLS par clé publique brute, sans certificat X.509 ni PKI.

Reverse proxy : serveur placé devant un service pour recevoir le trafic à sa place (ici, il filtre les journaux d'accès et n'expose jamais le service directement).

Slot A/B : les deux emplacements racine d'un serveur de sortie. On met à jour l'emplacement inactif, on bascule au démarrage suivant, et on revient à l'ancien si le nouveau échoue.

Spin bit : bit d'en-tête QUIC destiné à la mesure du réseau ; Warren le neutralise car sa valeur pourrait servir de fingerprint.

Squashfs : système de fichiers compressé en lecture seule, utilisé comme racine immuable des serveurs de sortie.

SS58 : format d'encodage d'adresse emprunté à l'écosystème Substrate, ici utilisé comme simple encodage, sans chaîne de blocs.

Voucher : bon anonyme produit par un paiement, dont seul le hash est stocké, et qui s'échange ensuite contre un abonnement sans révéler l'identité du payeur.

WarrenGuard : le moteur VPN sur QUIC de Warren, générique et open source.

Wire format : la définition binaire exacte d'un message échangé sur le réseau. C'est le contrat entre implémentations, figé par les golden vectors.

Annexe B. Références

Sources primaires du retrait du port forwarding :

- Retrait du port forwarding chez Mullvad, 29 mai 2023, ports coupés le 1er juillet 2023. `https://mullvad.net/en/blog/removing-the-support-for-forwarded-ports`
- Retrait progressif du port forwarding chez IVPN, 29 juin 2023, retrait complet le 30 septembre 2023. `https://www.ivpn.net/blog/gradual-removal-of-port-forwarding/`
- Perquisition avec mandat au siège de Mullvad, 18 avril 2023, repartie sans rien saisir. `https://mullvad.net/en/blog/mullvad-vpn-was-subject-to-a-search-warrant-customer-data-not-compromised`

Standards :

- RFC 6886, NAT Port Mapping Protocol (NAT-PMP)
- RFC 7250, Raw Public Keys in TLS
- RFC 5705, Keying Material Exporters for TLS
- RFC 9000 et RFC 9221, QUIC et les datagrammes non fiables
- RFC 9180, Hybrid Public Key Encryption (HPKE)
- RFC 9312, propriétés opérationnelles de QUIC (spin bit)
- RFC 9474 et RFC 9578, signatures aveugles RSA et jetons Privacy Pass

Travaux académiques et industriels :

- Analyse de l'extracteur de SNI QUIC du grand pare-feu chinois, USENIX Security 2025
- TunnelCrack, désanonymisation et contournement de tunnel VPN, USENIX Security 2023
- TunnelVision (CVE-2024-3661), détournement de route par DHCP
- Maybenot, framework de défense contre l'analyse de trafic, université de Karlstad
- Littérature sur le fingerprinting de sites web (PETS, CCS)

Paysage QUIC (transport de VPN) :

- Mullvad, obfuscation QUIC (MASQUE) pour WireGuard, 2025. `https://mullvad.net/en/blog/introducing-quic-obfuscation-for-wireguard`
- Cloudflare, WARP sur MASQUE comme transport par défaut, 2024. `https://blog.cloudflare.com/zero-trust-warp-with-a-masque/`
- Apple iCloud Private Relay, tunnel HTTP/3 et MASQUE, 2021. `https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf`
- Google IP Protection (MASQUE et jetons Privacy Pass) dans Chrome. `https://github.com/GoogleChrome/ip-protection`