Lockdown & Confirmed VPN

Audit Report - Server & Client

Current Audit: July 23rd, 2020 | Initial Audit: June 9th, 2019
Openly Operated Certification v1.0

https://openlyoperated.org/report/confirmedvpn

Audit Date:

Auditor Information

Name:

Email:

LinkedIn:

Qualifications:

The auditor certifies that they have performed the requested verifications in the attached Audit Kit. The auditor provides the following executive summary of their findings:

☐ I am not an employee of, or an investor in the product being audited.

Auditor Signature

Audit Kit

This Audit Kit contains end-to-end, verifiable proof of how Lockdown and Confirmed VPN work. Since Lockdown Secure Tunnel uses Confirmed VPN's servers, all references to Confirmed VPN also apply to Lockdown Secure Tunnel. This kit lets auditors and anyone to fully audit or personally verify the privacy and security claims. Anyone can also report issues, ask questions, or voice concerns via the contact methods under the Identify Operation section.

History

July 23rd, 2020 - Second Audit

- Since the first audit, we launched Lockdown, a privacy app for blocking trackers that has a Secure Tunnel feature, powered by Confirmed VPN. Outside of that, Confirmed VPN's backend saw mostly security updates/patches, some bug fixes, and a few new public resources for transparency. See comprehensive list of changes in the "Update - July 2020" section of the Audit Kit.

June 9th, 2019 - First Audit

- Initial revision. The first end-to-end Openly Operated Certification of a live, consumer facing product: Confirmed VPN.

Read-Only Account

A read-only account allows access to viewing the operator's AWS Console for Confirmed VPN — it allows you to see a limited version of what the operators and employees can see. The read-only account helps users feel more in control of what is happening behind the scenes with their data, but for a comprehensive look, users should read through the proof-backed points, references, and Audit Reports in the rest of this Audit Kit.

Confirmed VPN's read-only accounts are currently by request only. Individual account credentials are created for each auditor or user who requests one — this is an extra precaution to monitor and prevent abuse.

If you have been contracted to perform an audit, your Console credentials should have been sent to you separately.

Highlighted Services

- CodePipeline: Deploys code to all backend servers, including `Main`, `Admin`, `Webhook`, `Support`, `Renewer`, `Helper`, `Partner`, and VPN (`Bandwidth` repository) servers. Check that the commit hashes in the CodeCommit first step matches the code referenced in the Open Source section, and that the code is successfully deployed to the servers.
- EC2: Controller and status panel for all backend servers. Check that none of the servers (or "instances") were launched with SSH keys — there should be nothing under the "Key Name" column.
- CloudFormation: The infrastructure dashboard. Check that the templates of the stacks match up with the infrastructure files in the CloudFormation public repository.
- EC2 Security Groups: Firewall rules for both internal and external access. Check that for EC2 resources, the attached security group follows the Principle of Least Privilege.

Auditor Verifications

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Claims With Proof

The following are selected claims made by Confirmed VPN regarding user privacy and security. These claims are backed up by specific proof from source code, infrastructure, and other references. These claims are only highlights — more details and comprehensive proof are provided in the Requirements section. Learn about Claims With Proof.

No Logging User Traffic

The Claim Confirmed VPN does not log user traffic or user IPs/PII (Personally Identifiable Information), making it impossible for its operators to monetize, abuse, steal, or sell user data to third parties.

The Proof

- **VPN Traffic Not Logged -** Confirmed's VPN servers' logging configuration is set to "absolutely silent" mode[1][2], so no user IPs, traffic/packets, requests, urls, or anything personally identifiable are logged. Bandwidth accounting[3], which is used to prevent degradation of the VPN service to other users, only processes the download and upload metrics[4] (rounded down to the nearest megabyte to further increase privacy), and does not log the user IP or any user traffic.
- **Server Backend Doesn't Log IPs or PII -** The main user-facing website only logs critical errors[5], and even in those cases, no IP or Personally Identifiable Information is logged[6]. Normal requests and successful requests, such as visits to the main website landing page, are never logged[5]. Infrastructure-level internet traffic logging is disabled[7][8][9].
- **No Third Party APIs -** Confirmed VPN's servers use no third party APIs for its website[10] or for its server/backend dependencies[11], with the sole exception of our payment processor[12], which has limited exposure to user data (no VPN traffic) and can be avoided completely. The client apps do not use any third party APIs, including analytics, advertising, and user tracking[13].

References

1. Confirmed VPN "CloudFormation" infrastructure code repository, VPN Server, Lines 1033-1080
2. StrongSwan User Documentation, Logger Configuration - Levels and Subsystems/Groups, June 9th, 2019 .
3. Confirmed VPN "CloudFormation" infrastructure code repository, VPN Server, Lines 1081-1094
4. Confirmed VPN "Helper" source code repository, Accounting Module, Line 16
5. Confirmed VPN "Main" source code repository, Primary App Module, Lines 37-41
6. Confirmed VPN "Main" source code repository, Primary App Module, Line 16
7. Audit Kit Revision 1, "Audit Trail - Infrastructure Audit Trail"
8. Open Watch macOS, Rule for detecting infrastructure traffic logging, Storyboard, Line 834
9. Open Watch macOS, Rule for detecting infrastructure traffic logging, Audit Engine, Lines 557-567
10. Confirmed VPN "Main" source code repository, Website Header
11. Confirmed VPN "Main" source code repository, Server/Backend Dependencies, Lines 15-41
12. Audit Kit Revision 1, Open Infrastructure - Third Party APIs
13. Audit Kit Revision 1, Open Source - Client Repositories - No User Tracking

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

No Access To User Data

The Claim Confirmed VPN's operators have no unaudited access to the minimal data that is stored to keep the service running. Any employee access is fully auditable and specific account queries to help resolve customer support issues automatically alerts the user involved.

The Proof

- **User Data Locked & Encrypted -** User data is stored in databases that operators do not have the password to[1][2][3][4]. Sensitive user data is also encrypted[5][6][7][8][9] or hashed[10][11][12][13][14][15] before being stored into the database, with encryption keys and salts that operators do not have access to[16][17]. Any operator attempt to access these passwords or keys is reported by an automated auditing tool[18][19].
- **No Direct Access -** The databases are configured with firewall rules preventing both external access and operator access[20][21]. Servers are only launched with no SSH key[22][23] and the SSH port[24] disabled, and with no other ways to issue unaudited commands[25][26].
- **Audited Indirect Access -** Customer support cases that require access to user data such as a user's subscription information, is serviced through an audited[27] Support dashboard which automatically alerts the user whenever this access occurs[28][29], and does not allow access to hashed passwords[30]. Database migrations are done through an audited[31] Admin database interface, which provides an audit trail[32][33][34] of what queries were executed on the database, preventing potential abuse.

References

1. Confirmed VPN "CloudFormation" infrastructure code repository, Shared, Line 1819
2. Confirmed VPN "CloudFormation" infrastructure code repository, Shared, Lines 924-925
3. AWS Systems Manager Documentation, Parameter Store SecureString, June 9th, 2019 .
4. Confirmed VPN "CloudFormation" infrastructure code repository, Shared, Lines 820-830
5. Confirmed VPN "Shared" source code repository, Secure Utility, Lines 63-71
6. Wikipedia, Advanced Encryption Standard, June 9th, 2019 .
7. Confirmed VPN "Shared" source code repository, User Model, Line 467
8. Confirmed VPN "Shared" source code repository, Source Model, Line 235
9. Confirmed VPN "Shared" source code repository, Subscription Model, Line 164
10. Confirmed VPN "Shared" source code repository, Secure Utility, Lines 12-22
11. Confirmed VPN "Shared" source code repository, Secure Utility, Lines 24-29
12. Confirmed VPN "Shared" source code repository, User Model, Line 466
13. Confirmed VPN "Shared" source code repository, User Model, Line 463
14. Wikipedia, Secure Hashing Algorithm 2, June 9th, 2019 .
15. Confirmed VPN "Shared" source code repository, Secure Utility, Lines 12-22
16. Confirmed VPN "CloudFormation" infrastructure code repository, StackSet-Shared, Lines 532-623
17. AWS Systems Manager Documentation, Parameter Store SecureString, June 9th, 2019 .
18. Audit Kit Revision 1, Audit Trail
19. Open Watch source code repository, Open Watch Engine, Lines 441-453
20. Confirmed VPN "CloudFormation" infrastructure code repository, Shared, Line 1836
21. Confirmed VPN "CloudFormation" infrastructure code repository, Shared, Line 2011
22. AWS CloudFormation Documentation, AWS::AutoScaling::LaunchConfiguration - KeyName, June 9th, 2019 .
23. Confirmed VPN "CloudFormation" infrastructure code repository, Main, Lines 572-589
24. Confirmed VPN "CloudFormation" infrastructure code repository, Main, Lines 96-107
25. AWS Systems Manager Documentation, "ssmmessages API calls", June 9th, 2019 .
26. Confirmed VPN "CloudFormation" infrastructure code repository, StackSet-Shared, Lines 117-123
27. Confirmed VPN "Support" source code repository, Primary App Module, Line 23
28. Confirmed VPN "Support" source code repository, Support Controller, Line 61
29. Confirmed VPN "Support" source code repository, Support Controller, Line 93
30. Confirmed VPN "Support" source code repository, Support Controller, Line 95
31. Confirmed VPN "Support" source code repository, Primary App Module, Line 25
32. Audit Kit Revision 1, Database Queries Audit Trail
33. Confirmed VPN "Admin" source code repository, Database Controller, Line 56
34. Confirmed VPN "Admin" source code repository, Database Controller, Line 58

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Modern Security & Encryption

The Claim Confirmed VPN uses modern security and encryption to protect both user VPN connections and the service itself from internal and external threats.

The Proof

- **Advanced Encryption -** User data stored on Confirmed is protected by modern encryption (AES)[1][2][3][4][5] and hashing (SHA-512, bcrypt)[6][7][8][9][10][11][12] algorithms, and user VPN connections use Certificate-Based Authentication[13][14] with stronger encryption than the settings designated by the non-profit Internet Engineering Task Force[15][16][17].
- **Network Security -** Confirmed VPN uses Security Groups as firewalls that block all traffic by default, and only whitelist specific ports and ingress as necessary[18]. Secure transit via HTTPS/TLS 1.2+ is required on all external connections[19][20][21][22], and also even enforced on communication between data center regions[23][24] and to file systems[25][26]. The VPN's automated network intrusion software and rulesets are used to keep the latest threats at bay[27][28][29].
- **Minimal Server, Database Permissions -** Both servers and databases follow the Principle Of Least Privilege[30], allowing only permissions that are absolutely necessary for their operation. Servers can only access parameters and resources that are directly used in their functionality[31][32], and their database roles can only read, write, modify, or delete specific table rows[33].
- **Security Updates -** The node.js framework, library dependencies, the operating system and its packages are kept updated[34][35][36][37] to ensure the latest security patches are installed. Alerts immediately notify the operator[38][39][40] when something is out of date, allowing them to perform the necessary updates as soon as possible.

References

1. Confirmed VPN "Shared" source code repository, Secure Utility, Lines 63-71
2. Wikipedia, Advanced Encryption Standard, June 9th, 2019 .
3. Confirmed VPN "Shared" source code repository, User Model, Line 467
4. Confirmed VPN "Shared" source code repository, Source Model, Line 235
5. Confirmed VPN "Shared" source code repository, Subscription Model, Line 164
6. Confirmed VPN "Shared" source code repository, Secure Utility, Lines 12-22
7. Confirmed VPN "Shared" source code repository, Secure Utility, Lines 24-29
8. Confirmed VPN "Shared" source code repository, User Model, Line 466
9. Confirmed VPN "Shared" source code repository, User Model, Line 463
10. Wikipedia, Secure Hashing Algorithm 2, June 9th, 2019 .
11. Confirmed VPN "Shared" source code repository, Secure Utility, Lines 12-22
12. Wikipedia, bcrypt, June 9th, 2019 .
13. StrongSwan Documentation, Security Recommendations - Certificate Based Authentication, June 9th, 2019 .
14. Confirmed VPN "CloudFormation" infrastructure code repository, VPN-IKEv2, Lines 1017-1018
15. Confirmed VPN "CloudFormation" infrastructure code repository, VPN-IKEv2, Lines 1011-1012
16. Internet Engineering Task Force, RFC 6379, "Suite B Cryptographic Suites for IPsec", October 2011
17. Wikipedia, Internet Engineering Task Force, June 9th, 2019 .
18. Audit Kit Revision 1, Open Infrastructure - Security Groups (Firewall)
19. Confirmed VPN "CloudFormation" infrastructure code repository, Main, Lines 523-534
20. Confirmed VPN "CloudFormation" infrastructure code repository, Main, Line 521
21. Wikipedia, Hypertext Transfer Protocol Secure, June 9th, 2019 .
22. Wikipedia, Transport Layer Security - TLS 1.2, June 9th, 2019 .
23. Confirmed VPN "CloudFormation" infrastructure code repository, Shared, Lines 2538-2974
24. AWS VPC Documentation, "What is VPC Peering?", June 9th, 2019 .
25. Confirmed VPN "CloudFormation" infrastructure code repository, Admin, Line 1019
26. AWS EFS Documentation, Encryption In Transit, June 9th, 2019 .
27. Wikipedia, Suricata (software), June 9th, 2019 .
28. Confirmed VPN "CloudFormation" infrastructure code repository, VPN-IKEv2, Lines 982-986
29. Emerging Threats Documentation, June 9th, 2019 .
30. Wikipedia, Principle of Least Privilege, June 9th, 2019 .
31. Confirmed VPN "CloudFormation" infrastructure code repository, Main, Lines 345-392
32. AWS Systems Manager Documentation, "Restricting IAM Permissions Using Hierarchies", June 9th, 2019 .
33. Confirmed VPN "Shared" source code repository, Database Schema, Lines 187-215
34. Confirmed VPN "Main" source code repository, Node Update Deployment Script, Lines 3-4
35. Confirmed VPN "CloudFormation" infrastructure code repository, Admin, Line 714
36. Confirmed VPN "CloudFormation" infrastructure code repository, Admin, Lines 746-766
37. Ubuntu Documentation, Automatic Security Updates, June 9th, 2019 .
38. Confirmed VPN "CloudFormation" infrastructure code repository, Admin, Lines 1280-1303
39. Confirmed VPN "CloudFormation" infrastructure code repository, Main, Lines 1226-1249
40. Confirmed VPN "CloudFormation" infrastructure code repository, StackSet-Shared, Lines 818-828

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Requirements

The following are requirements for being Openly Operated, and they exist to ensure a product's full, verifiable transparency. Without them, claims about security or privacy could not be made because there could be gaps, loopholes, or backdoors in these claims. Learn about Requirements.

Identify Operation

Product: Confirmed VPN

Privacy and Security App

Confirmed VPN is an app and service that increases the security and privacy of its users' internet connections. It has backend servers and infrastructure, as well as client apps on macOS, Windows, iOS, and Android.

website | contact

Location: United States

Delaware, California, and Florida

Confirmed VPN is operated by Confirmed Inc, which is incorporated in Delaware with filing number 6543601.

Corporate: 251 Little Falls Drive, Wilmington, DE 19808

San Francisco: 535 Mission St, 14th Floor, San Francisco, CA 94105

email | telegram | reddit | twitter

Co-Creator: Johnny Lin

San Francisco, CA

Johnny built and works on Confirmed VPN's Openly Operated backend services. He was an engineer at Apple iCloud and since then has built several apps featured by the App Store. Johnny is an alumnus of Brown University.

email | medium | telegram | linkedin | twitter

Co-Creator: Rahul Dewan

Miami, FL

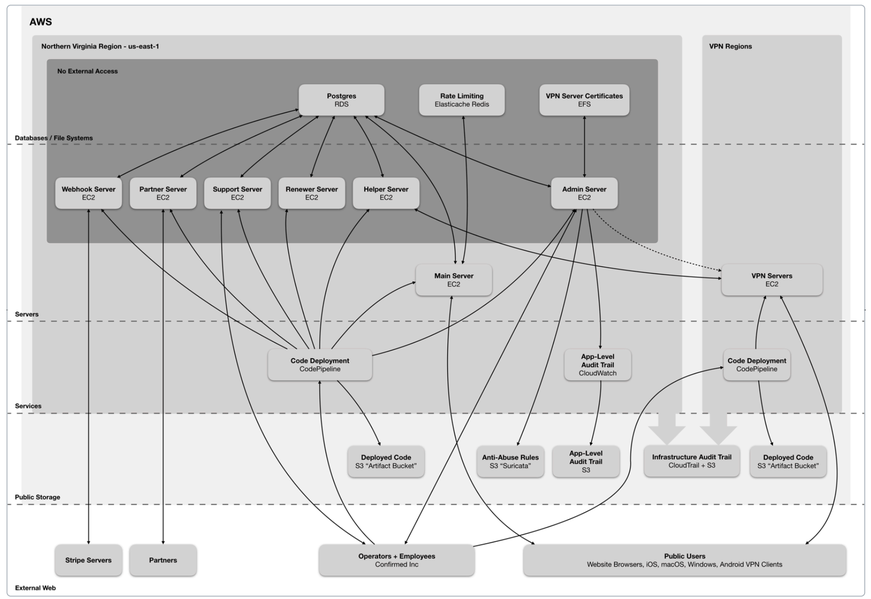
Rahul built and works on Confirmed VPN's clients on Mac, PC, iOS, and Android, and also handles business development. He also created Duet Display, a second screen for your iPad, and has previously worked at Apple. Rahul is an alumnus of Stanford University and Georgia Tech.

email | linkedin

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Open Infrastructure

Confirmed VPN is 100% open infrastructure, with one public infrastructure repository. All configuration and usages of third-party APIs are described below.



Infrastructure

AWS CloudFormation | GitHub | `commit cd181e99dafce996c5f6066ce437c6f138ef9450`

Confirmed VPN runs on Amazon Web Services, using CloudFormation to bring up almost every resource, from servers (which run Ubuntu) to DNS configurations to deployment pipelines. All the CloudFormation templates are public on Github. Confirmed VPN's main API region is in the `us-east-1` Northern Virginia region, and also has twelve identical VPN regions: `us-east-1`, `us-west-1`, `ap-south-1`, `ap-northeast-2`, `ap-southeast-1`, `ap-southeast-2`, `ap-northeast-1`, `ca-central-1`, `eu-central-1`, `eu-west-1`, `eu-west-2`, and `sa-east-1`.

Confirmed VPN mainly uses IKEv2 for encrypting user traffic currently, but has OpenVPN deployed on a small scale for testing.

Typical User Flow

To get a basic understanding of the infrastructure as it relates to a real customer user flow, you can follow the infrastructure diagram through the steps below, from sign up to connecting to the VPN. The following is only an example, user flows will look different depending on the type of subscription the customer signs up for, the platform they're on, and their payment type.

1. **Download Client** - The customer goes to the Confirmed VPN website and downloads the Mac client app, both of which are hosted on the `Main` server.
2. **Signup** - The Mac app guides the customer through account creation and subscription signup, also all hosted on the `Main` server.
3. **Configure VPN** - The Mac app requests a client VPN certificate from `Main`, which ensures that the account has a valid subscription, and returns a client VPN certificate to the user's Mac app. This VPN certificate is installed on the user's system.
4. **Connect to VPN** - The Mac app then connects the VPN to point to the user's selected region/endpoint, such as US East. This connection is made to the VPN server in the specified region.
5. **Validate Connection** - The VPN server activates the `Bandwidth` script, which makes a request to the `Helper` server, which checks that the user has a valid subscription and has a valid account. If `Helper` says the user account looks fine, then the connection is accepted.
6. **Connection Encrypted** - The user's internet traffic is now encrypted through Confirmed VPN.

The user flow above only involves `Main`, VPN/ `Bandwidth`, and `Helper`, because the typical user will only interact with these servers, not the others. The other servers (such as `Admin` and `Renewer`) are not externally accessible and have administrative/support functions to keep the service running smoothly. For details on what each of them do, see the Open Source section below.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	

Security & Privacy

- **Security Groups (Firewall)** - Every server type's firewall rules follows the Principle of Least Privilege, so in case of any external compromise or internal threat, damage is minimized.
  - `Admin` - This server is only for administrators, so it only allows ingress from a single CIDR, or range of IPs. The only other use case for ingress to Admin is when a new VPN server instance is created and the VPN instance requests the Server VPN certificate from Admin using a special internal-only API endpoint. This is an ingress rule that is by default disabled, and must be manually enabled using the Admin Dashboard.
  - `Support` - This server is only for support staff, so it only allows ingress from a single CIDR, or range of IPs.
  - `Renewer` - This server is never available externally or internally, because it's only responsible for running a once-daily subscription renewal job. There are no ingress groups allowed to Renewer.
  - `Helper` - This server communicates with VPN servers in all regions to keep bandwidth stats updated. It allows ingress from the internal local network.
  - `Partner` - This server is only available to Partners who have read-only access to anonymized data about their referrals, so it only allows ingress from a single CIDR, or range of IPs.
  - `Webhook` - This server only listens to webhook calls from Stripe, our payment processor, so it only allows ingress from the IPs that Stripe has specified for webhooks.
  - `Main` - This server is the main API server for all clients and browsers, so it allows ingress from any external IP, but limits it to port 80 (HTTP, automatically redirected to HTTPS) and port 443 (HTTPS).
  - `VPN` - These servers do the actual encrypt of user internet traffic for all clients, so it allows ingress from any external IP, but limits it to only UDP ports 500 and 4500 and IP protocol 50 and 51, which are the default ports and protocols for IKEv2 VPN.
- **VPN Server Certificate Security** - In order to encrypt user connections, a VPN must have the correct server certificate to prove to the client that it is indeed the legitimate VPN service that the user intends to connect to. If this server certificate's private key is not properly secured or mishandled, a malicious server could impersonate the user's VPN server, while actually intercepting and stealing the user's personal data — defeating the whole point of a VPN. Confirmed VPN secures the server certificate's private key in the following ways:
  - **Separate, Encrypted File System** - The root certificate, which is used to generate the server certificates, is stored on a separate file system, which is encrypted, to create an additional layer of protection. This file system's Security Group by default allows only access from the Admin server, which also enforces an encrypted connection to the file system.
  - **Manually Toggle Access** - In order for VPN servers to download the server certificate on initialization, the Admin Dashboard must manually enable internal network access to the Admin API for downloading the server certificate. This allows the the VPN server to temporarily have network access to the Admin server for the short duration that it's initializing (usually ~2 minutes).
  - **Internal IP Check** - In addition to the network-level firewall rule, the app-level API also ensures that any access must come from an internal IP, denying access and alerting administrators as well. This IP restriction also prevents operators from downloading the server certificate (insider threat), since operators access the Admin server through a non-internal IP.
  - **Access Secret Required** - The Admin API for downloading the server certificate also requires a Certificate Access Secret. This secret is randomly generated and stored separately in a Parameter Store value which the VPN's server role does not have access to. This key is manually input by Confirmed operators whenever a new instance is brought up.
  - **Audited Access** - Since the Server Certificate API should only be downloaded upon initialization of new VPN servers, a notification is sent to administrators whenever this API is successfully accessed. All accesses of this API also trigger an app-level Audit Log which records the action to both Cloudwatch Logs and S3.
  - **Certificate Rotation** - In the unlikely case that all infrastructure-, network- and app-level security measures are bypassed and the server certificate is compromised, the Admin Dashboard is able to generate a completely new certificate chain and point VPN servers to accept the new certificate chain. New clients are then released to connect to the new VPN servers, instead of the old, now-insecure servers.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

- **Maximize Internal Network Usage** - Whenever possible, communication between servers should use internal networks rather than routing through external networks, which may be untrusted. All servers, with the exception of the VPN servers, reside within the same VPC (Virtual Private Cloud) and communicate within the same internal network. To securely communicate with the VPN servers in different regions, VPC Peering is used, which is encrypted by default and stays within the AWS network backbone.
- **Redis Database Security** - The Redis database stores temporary user sessions (cookies) and is also used for preventing brute-force attacks, and it has at-rest encryption and transit encryption enabled. Its auth token is randomly generated and operators cannot access it. It is not accessible by the external web.
- **Postgres Database Security** - The Postgres database stores user account and subscription information. Its VPC Security Group prevents access by the external web. Its password is randomly generated and encrypted as a Secure String, so even the operators never see it. Any access or decryption of the database password triggers a tamper-proof and publicly viewable CloudTrail event, which are reported as a violation by Open Watch.
    - **Limited Database Roles** - Since each server type only needs to read, write, modify, and delete specific tables, each server type is granted only the permissions that it needs, following the Principle of Least Privilege. Database roles and permissions for each server type are here, so that in the case of a server being compromised, the damage is limited. Operators do not have access to any of these roles' passwords, as they are randomly generated upon initialization — example here.
    - **Only Audited Access** - Since databases sometimes require schema migration and other mandatory manual maintenance, non-direct queries to the database are allowed, but these queries are logged for auditing purposes, and can only be executed only through the Admin dashboard. This prevents potential internal and external abuse of this database access — see the Audit Trail requirement for complete explanation.
- **VPN Anti-Abuse Rules** - In order to protect Confirmed VPN's customers and users from potentially malicious users, automated anti-abuse rules are enforced for VPN traffic. These rules are enforced by Suricata, an open source intrusion detection and prevention system. Confirmed is fully transparent on exactly what these rules are — they're based on the open Emerging Threats ruleset, with our VPN's specific customizations publicly available here. To access a specific rule file, simply append the file name that's listed to the same url — for example, the direct link to access the confirmed.rules file is this url.
    - **User Privacy** - The anti-abuse rule triggers are performed algorithmically and no individual has manual access to or processes any user traffic. Even in the case that an anti-abuse rule is triggered, the VPN user's IP address is not logged, maintaining the user's privacy.
- **No Direct Server Access** - Direct, unaudited server access is a security and privacy risk because it allows changing code, installation of unauthorized/malicious applications, monitoring traffic, and more. Confirmed VPN disables all direct server access for everyone, including employees.
    - **Servers have no SSH key** - SSH is the most common way of accessing servers directly, but it requires an SSH key to be preloaded on the server. The KeyName property of the instances' launch configuration in all server types are not defined, so instances launch with no SSH key, making direct SSH access impossible — here's an example for the `Main` server.
    - **Firewall blocks SSH port** - AWS Security Group rules are configured to not allow traffic on all server types for SSH's port 22, so even if an SSH key managed to get onto a server somehow, it still couldn't be used to directly access the server — here's an example of the firewall configuration for `Main`.
    - **Session Manager is disabled** - AWS's Session Manager is Amazon's brand of direct server access. This feature is blocked by infrastructure role permissions. Additionally, any Session Manager session creation is recorded by the Infrastructure Audit Trail, and the open source Open Watch app (described below) detects these cases automatically.
- **No Access To Keys Or Salts** - In addition to using strong encryption/hashes, operators have no access to the encryption keys or salts, because they're randomly generated on initialization, and encrypted as a Secure String, so even the operators never see it. Any access or decryption of the database password triggers a tamper-proof and publicly viewable CloudTrail event, which are reported as a violation by Open Watch.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

- **HTTPS Required** - HTTPS is a basic necessity for encrypting internet connections, reducing the chances of eavesdropping and preventing certain attacks. Confirmed VPN requires HTTPS by redirecting all HTTP requests to HTTPS. Additionally, Confirmed VPN requires TLS 1.2, a modern type of HTTPS encryption.
- **Limited Server Roles** - Roles for server instances abide by the Principle Of Least Privilege, by limiting access to only what is necessary — for instance, the `Main` server role cannot access the `Admin` server's database user password, because the `Admin` database password is stored under a different section of the Parameter Store Hierarchy.
- **Updated Security Patches** - All servers run the latest version of Ubuntu 16 LTS, and security patches are automatically installed as they're released using Ubuntu's unattended-upgrades package. Operators are alerted when patches require a system restart, at which point the restarts and/or reprovisioning of servers is done at the earliest possible time.

Third Party APIs

Since other companies that run third party APIs (such as analytics tools, advertisement targeting tools, etc) are likely not Openly Operated nor provably transparent, usage of third party APIs should be minimized in order to reduce risks to user data. While using third party APIs would have been faster and easier for many features like analytics and monitoring, Confirmed VPN built those features from the ground up so that we can provide end-to-end proof of the privacy and security of user data.

Confirmed VPN uses one third party API (which the user may skip by opting out — see below): Stripe, for direct payment processing. Stripe stores user payment data, but does not have access to user emails, which reduces user data exposure. Furthermore, Stripe does not have access to any VPN traffic, as there is no logging (see Proof Of Claims) and Stripe's servers only have access to the `Webhook` server.

Confirmed VPN users who wish to opt out of Stripe may subscribe to Confirmed VPN via the iOS or Android app, which allows the user to provide payment through the App Store or Play Store. This way, no user information at all is transmitted to Stripe.

Code Deployment

Code is deployed using AWS CodePipeline, which consists of CodeCommit (a hosting service for code repositories), CodeBuild (building and testing code), and CodeDeploy (deploying code to servers) — see the relevant infrastructure code here.

To deploy code, code is `git push` ed to the CodeCommit repository, where it is automatically built by CodeBuild and uploaded into an S3 "Artifact Bucket" (aka "Deployed Code" on the Infrastructure diagram). CodeDeploy then deploys the code in the artifact bucket to the servers. So, the code in the S3 "Artifact Bucket" is the same as the code in production.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Open Source

Confirmed VPN is 100% open source, and has both servers and clients. Servers are the actual VPN servers that encrypt and route user traffic, as well as support/administrative servers. Clients are downloadable apps on macOS, Windows, iOS, and Android that connect to the VPN servers. Architecture and code for everything is detailed below, with bullet points highlighting security and privacy features.

Server Architecture

The majority of Confirmed VPN's backend is written in Javascript, using Node.js on the Express framework. It currently has the following backend code repositories: `Main`, `Admin`, `Webhook`, `Support`, `Renewer`, `Helper`, and `Partner` all depend on the `Shared` repository, and there is a separate `Bandwidth` repository that is deployed to all VPN servers. Backend servers are hosted on Amazon Web Services, with infrastructure brought up via CloudFormation. See Open Infrastructure for details.

The latest NodeJS Long Term Support ("LTS") version is installed on all servers, and operators are alerted when a new update is available. The updates are manually reviewed for compatibility, then installed via a simple deployment.

Server Repository: `Shared`

Shared Node.js Modules, Models, Utilities | GitHub | `commit 73ee624e79331a7dc2831b7af767040d6a6ea1ad`

This exists to reduce code duplication between the `Main`, `Admin`, `Webhook`, `Support`, `Renewer`, `Helper`, and `Partner` servers, and contains common shared code like logging, email, database access, models, security, and other utilities. Updating `Shared` requires a redeployment of dependent servers for changes to take effect.

- **User Data Secured** - Sensitive user data should always be secured with strong encryption or strong hashes. This ensures that internal operators can't read user data, and also makes it extremely difficult for hackers to read user data in the unlikely case of a database compromise.
  - **Strong Encryption** - User data that needs to be retrieved are encrypted using AES-256, which is the same encryption the United States government uses for top secret information. View the encryption code and its usages for user email, VPN certificates, and In-App Purchase receipts.
  - **Strong Hash** - User data that does not need to be retrieved (but still requires checking) are salted and hashed using SHA-512 (for faster performance/lookup) or bcrypt (for strongest security). View the SHA hashing code, bcrypt hashing code, and their usages for user email, brute force prevention, and user password.

Server Repository: `Main`

Node.js on Express | GitHub | `commit f9736c992107064ed47296a0c7e6980662a0b5a4`

This is the main public-facing website, which also hosts the API backend that clients connect to. It has static pages describing what Confirmed VPN is, a user dashboard for account management, and serves VPN certificates for subscribed users. This and the `VPN` servers are the only servers that are accessible externally and not whitelisted.

- **Secure User Sessions** - When a user logs in to access their account, they are given a cookie to keep them logged in. This cookie is secured by a random session secret, and expires every two hours, or when the user logs out. A valid session is required to access user account pages and other user-only endpoints.
- **Strong Password Requirement** - Users signing up via email are required to have a strong password, which is at least 8 characters in length and has at least one capital letter, lowercase letter, number, and special character.
- **Email Confirmation Required** - To prevent unauthorized signups, users signing up via email are required to confirm that they own the email by clicking a confirmation link in their inbox. The confirmation code is randomly generated.
- **Input Validation and Sanitization** - All inputs received by `Main` and its API must pass validation checks to ensure valid data is being received, as well as to prevent SQL injection and other potential attacks. Instead of using values from POST bodies directly, every value is passed through validation middleware, and only valid values are set into a new request.values object, which is then safe to use. Examples of validation include checking for a valid email address on sign-in, ensuring that In-App Receipts are base-64 encoded, and ensuring that email confirmation codes are alphanumeric.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

- **Brute Force Prevention** - Each user is limited to a certain number of requests per time window, preventing malicious users from spamming the website and API. See code for this feature here, and examples of its usage here and here.
- **No User Tracking** - Apps and services should have as little user tracking as possible, for both privacy and security reasons.
  - **No Third Party Analytics** - Using third party analytics tools may be convenient, but they're also a privacy risk. Some tools are hosted by advertisement companies (like Google Analytics), and other tools save obscene amounts of user activity onto third party servers, like full screen recording and tracking. Confirmed VPN uses no third party analytics on its website nor in its backend dependencies, minimizing the risk of user privacy violations.
  - **No Access To User Data** - As described in `Shared`, user data is encrypted or hashed by keys and salts that the operators Confirmed do not have access to. The Open Infrastructure section below additionally demonstrates that there's no direct, unaudited operator access to the database, making it extremely difficult for operators to view, much less exploit, user data.
  - **Minimal Logging With No IPs/PII** - Only errors are logged for debugging purposes, and even in those cases, no IP or Personally Identifiable Information is logged. Normal requests and successful requests, such as visits to the main website landing page, are not logged.

Server Repository: `Admin`

Node.js on Express | GitHub | `commit 20f0bed9faa6998542b652c4505f3b7c10c2e53d`

This is a private dashboard for Confirmed VPN, and is only accessible to administrators connecting via whitelisted IPs. It is heavily monitored and logged and is only intended to be used in specific circumstances, such as managing anti-abuse rules, uploading new client versions, deleting users, and other special cases.

- **Domain-Specific Login Required** - The Admin Dashboard requires registration and email confirmation of an account with an address ending in @confirmedvpn.com, preventing anyone else from accessing the dashboard.
- **Secure Admin Sessions** - When an admin user logs in to access the Admin Dashboard, they are given a cookie to keep them logged in. This cookie is secured by a random session secret, and expires every one hour, or when the admin user logs out. A valid session is required to access the Admin Dashboard pages and all its features.
- **Admin Email Alerts** - Access to the Admin Dashboard is heavily monitored and logged, because only operators should be accessing it. Email alerts are sent to the administrator email whenever a successful or failed login attempt occurs.
- **Database Queries Publicly Audited** - The Admin Dashboard is able to make queries to the database, with the intended purpose of database migrations and other rare circumstances. To keep the operator honest and ensure the operator doesn't attempt to export user data (even though sensitive user data is encrypted and hashed by keys unavailable to the operator, as shown in the `Shared` section above), all database queries are automatically recorded to CloudWatch Logs and a public S3 Bucket, where auditors and the public can examine the queries. The Audit Trail section in this Audit Kit has more about how this Database Audit Trail works and how to view it.
- **Server Certificate Security** - VPN servers require a Server Certificate in order to properly encrypt clients VPN connections. The Admin server generates these Server Certificates (referred to as "source" in the code) and dispenses them to the VPN servers through an API. Since these Server Certificates encrypt user connections, it's extra important that they're accessed by unauthorized entities. So, the Admin server requires a secret key to access the source-dispensing API, and also requires that the Admin Dashboard explicitly toggle this API to "enabled". Finally, every time the Server Certificate API is called, an email alert is sent to the administrator, and the event is logged to CloudWatch Logs and an S3 Bucket.
- **Server Certificate Rotation** - In the unlikely case that the Server Certificate is compromised, the Admin Dashboard is able to generate a new Server Certificate and corresponding Client Certificates. The "source" is then switched to the new Server Certificate, and new versions of the client VPN apps are released to connect to the new servers.

Server Repository: `Support`

Node.js on Express | GitHub | `commit 59ec7b6f0db125fa43149978e1101a3c52089581`

This is a private support dashboard for Confirmed VPN and is only accessible to support employees connecting via whitelisted IPs. It's a limited version of the `Admin` server, and is used for more frequent customer support requests such as diagnosing subscription and account issues.

- **Users Alerted On Data Access** - When a user has a problem with their account and reaches out to Confirmed VPN's support, the support agent may require temporary access to the user's basic account information, such as email address, in order to diagnose and fix the problem. To prevent abuse of this feature, the user is automatically alerted whenever a lookup of their account occurs.
- **Limited Access To User Data** - Sensitive user information such as password hashes and encrypted In-App Purchase receipts are not accessible via the Support dashboard, because this data is never necessary to diagnose a user's issues. Only this select whitelist of account details are accessible to support agents.
- **Domain-Specific Login Required** - The Support Dashboard requires registration and email confirmation of an account with an address ending in @confirmedvpn.com, preventing anyone else from accessing the dashboard.
- **Secure Support Sessions** - When a support user logs in to access the Support Dashboard, they are given a cookie to keep them logged in. This cookie is secured by a random session secret, and expires every two hours, or when the support user logs out. A valid session is required to access the Support Dashboard pages and all its features.
- **Support Email Alerts** - Access to the Support Dashboard is heavily monitored and logged, because only operators should be accessing it. Email alerts are sent to the administrator email whenever a successful or failed login attempt occurs.

Server Repository: `Webhook`

Node.js on Express | GitHub | `commit 3b2a08cc7d45fc6bd78f28c4511a99f735edfd5b`

This is a single-API server to receive requests from Stripe, Confirmed's payment processor for desktop subscriptions. It is only accessible by Stripe's whitelisted IPs. The requests from Stripe help Confirmed keep its subscription database updated and accurate, and also helps properly credit referral bonuses.

- **Verify Stripe Request Authenticity** - Even though the Webhook server's firewall security group only allows requests from Stripe's server IPs (see Open Infrastructure section for details), we take the extra cautious step of verifying that incoming requests are indeed authentic Stripe requests by validating them with Stripe's provided Webhook signature.

Server Repository: `Renewer`

Node.js on Express | GitHub | `commit 9368e54c76141419544e9d840f56cc51a7f2bab7`

This is a private server that is not accessible by any external IP. The Renewer server runs once a day and checks that users subscriptions are updated and accurate for App Store, Play Store, and desktop customers. Since the firewall security group does not allow any IP to access this server, and the only API calls are for renewing subscriptions and do not return user data, no additional security measures are necessary.

Server Repository: `Partner`

Node.js on Express | GitHub | `commit 6d91344b48b85767804d37a94c607e3e7227591a`

This is a private dashboard for Confirmed VPN's referral partners, who can accurately track their referrals without having access to any actual user information. The Partner dashboard is built as an example of how to build third-party partnerships that both maximize user privacy and can prove it. This server is only accessible to whitelisted IPs.

- **Anonymized Referral Data** - The Partner Dashboard is not able to access any sensitive user data, such as email addresses. The only data accessible to the Partner Dashboard are anonymized subscriptions and their details, such as the date the subscription was created, as well as aggregate referral data (referred to as "snapshot" in the code).
- **No Access To Other Partners' Data** - Each Partner has their own Partner Code, which is unique to them, and they're only able to access the anonymized referral data of their own Partner Code.
- **Secure Partner Sessions** - When a Partner user logs in to access the Partner Dashboard, they are given a cookie to keep them logged in. This cookie is secured by a random session secret, and expires every two hours, or when the partner user logs out. A valid session is required to access the Partner Dashboard pages and its features.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Server Repository: `Helper`

Node.js on Express | GitHub | `commit 1641ad2cf8641a6dbaa06a413c282640e785bf0b`

These are private servers that VPN servers communicate with in order to verify whether or not to enforce throttling measures, as well as to anonymously track bandwidth abuse. It has an API that checks if a user's subscription is active, revoked, and/or throttled. Since Helper is only accessible to the internal network of CIDR range 172.16.0.0/12, and the only API calls it has are to do accounting for bandwidth in kilobytes and tell VPN servers whether or not to throttle an expired or abusive client, no additional security is necessary.

Server Repository: VPN / `Bandwidth`

Node.js, Bash | GitHub | `commit bd9d6f933cf1615eca990bd3319afc6166817548`

The VPN servers are powered by StrongSwan, an open source VPN server that uses strong encryption. The servers also install a lightweight anti-abuse `Bandwidth` script. When a client connects, `Bandwidth` checks with `Helper` servers to see if anti-abuse throttling measures should be enforced. If so, that client's bandwidth is anonymously throttled for the duration of their session.

- **No Logging User Traffic Or IPs** - The VPN is configured for absolutely silent logging (level -1) as per the official StrongSwan configuration documentation, so no user IPs, traffic/packets, requests, urls, or anything personally identifiable are logged. Bandwidth accounting to prevent abuse/degradation of the VPN service only processes the download and upload metrics (rounded down to the nearest megabyte to further increase privacy), not the user IP or any user traffic.
- **Strong VPN Encryption** - Confirmed VPN uses Certificate Based Authentication (configuration here), which is inherently stronger than the Pre-Shared Keys that many other popular VPNs today rely on. Confirmed VPN's cryptographic suite uses stronger encryption settings than the settings designated by the non-profit Internet Engineering Task Force in RFC 6379, which provides for modern security while maintaining interoperability with iOS, macOS, Windows, and Android clients.
- **Anti-Abuse System** - Confirmed VPN has automated enforcement of anti-abuse rules, which are designed to prevent malicious VPN users from abusing and degrading the experience and speed of other users.
  - **Rules For Preventing Abuse** - To protect our users, anti-abuse rules are enforced by Suricata, an open source intrusion detection and prevention system. Confirmed is fully transparent on exactly what these rules are — they're based on the open Emerging Threats ruleset, with our VPN's specific customizations publicly available here. To access a specific rule file, simply append the file name that's listed to the same url — for example, the direct link to access the confirmed.rules file is this url.
  - **User Privacy** - The anti-abuse rule triggers are performed algorithmically and no individual has manual access to or processes any user traffic. Even in the case that an anti-abuse rule is triggered, the VPN user's IP address is not logged, maintaining the user's privacy.

Client Repository: `Confirmed-iOS`

Swift | GitHub | `commit fae95016d0a6772626c1c22b61b13912403d3442`

This is the iOS client for Confirmed VPN. Its primary purpose is to log in to Confirmed `Main` Servers and request VPN configurations to install the VPN on iOS devices for encrypting and protecting user traffic.

- **No User Tracking** - Many apps today harvest user data (location, contacts, web traffic, metadata, screen capture, etc) for profit, analytics, and advertisement targeting. Confirmed iOS contains no user data harvesting, and uses no third-party analytics libraries or APIs, eliminating this exposure of user data. All of the libraries used by Confirmed iOS are open source, and listed here and here.
- **iOS App Sandbox** - iOS apps reside within an App Sandbox, preventing other apps from accessing the data stored within other apps. This means on iOS, the user email, password and In-App Receipt stored in Confirmed VPN has an additional layer of protection against other potentially malicious apps on your device.
- **Secure API Connections** - Confirmed iOS makes HTTPS connections to the Main server API, which encrypts connection data and also reduces the risk of man-in-the-middle attacks.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	

Client Repository: `Confirmed-Mac`

Swift | GitHub | `commit 12410ec08a0171aded8e3574ace36863ffb4c6f5`

This is the macOS client for Confirmed VPN. Its primary purpose is to sign in to Confirmed `Main` Servers and request VPN configurations to install the VPN on Mac devices for encrypting and protecting user traffic.

- **No User Tracking** - Many apps today harvest user data (location, contacts, web traffic, metadata, screen capture, etc) for profit, analytics, and advertisement targeting. Confirmed macOS contains no user data harvesting, and uses no third-party analytics libraries or APIs, eliminating this exposure of user data. All of the libraries used by Confirmed macOS are open source, and listed here and here.
- **Secure API Connections** - Confirmed macOS makes HTTPS connections to the Main server API, which encrypts connection data and also reduces the risk of man-in-the-middle attacks.

	Choose Result ▼	
	Choose Result ▼	

Client Repository: `Confirmed-Android`

Java | GitHub | `commit d58cea087d7af2523e4a29fd659fb73113924b87`

This is the Android client for Confirmed VPN. Its primary purpose is to sign in to Confirmed `Main` Servers and request VPN configurations to install the VPN on Android devices for encrypting and protecting user traffic.

- **No User Tracking** - Many apps today harvest user data (location, contacts, web traffic, metadata, screen capture, etc) for profit, analytics, and advertisement targeting. Confirmed Android contains no user data harvesting, and uses no third-party analytics libraries or APIs, eliminating this exposure of user data. All of the libraries used by Confirmed Android are open source, as it's a forked version of the open source StrongSwan Android client.
- **Secure API Connections** - Confirmed Android makes HTTPS connections to the Main server API, which encrypts connection data and also reduces the risk of man-in-the-middle attacks.

	Choose Result ▼	
	Choose Result ▼	

Client Repository: `Confirmed-Windows`

C# | GitHub | `commit f5a8ecf5672d7de3febe48deda07b248be175816`

This is the Windows client for Confirmed VPN. Its primary purpose is to sign in to Confirmed `Main` Servers and request VPN configurations to install the VPN on Windows devices for encrypting and protecting user traffic.

- **No User Tracking** - Many apps today harvest user data (location, contacts, web traffic, metadata, screen capture, etc) for profit, analytics, and advertisement targeting. Confirmed Windows contains no user data harvesting, and uses no third-party analytics libraries or APIs, eliminating this exposure of user data. Confirmed Windows is 100% open source.
- **Secure API Connections** - Confirmed Android makes HTTPS connections to the Main server API, which encrypts connection data and also reduces the risk of man-in-the-middle attacks.

Audit Trail

Audit trails show all operator actions since the creation of the Openly Operated product. Their purpose is to provide an unbiased, verifiable source of truth. For Confirmed VPN, since its Infrastructure Audit Trail does not record the operator's manual database actions, there is an additional audit trail called the Database Audit Trail. Since it is critical to keep the VPN Server Certificate secure, a third audit trail called the Actions Audit Trail records whenever the VPN Server Certificate and its private key is accessed.

Infrastructure Audit Trail

The infrastructure audit trail provides a log of all major operator actions in bringing up and maintaining the infrastructure. They are usually automatically generated by the platform that the product is built on, and should have a way to verify integrity. Confirmed VPN uses AWS's CloudTrail service for its infrastructure audit trail.

- **Impartial** - This audit trail is generated automatically by AWS CloudTrail. AWS CloudTrail logs all operator actions on AWS, and operators are not able to alter or delete any CloudTrail logs.
- **Authentic** - Confirmed VPN's CloudTrail logs have Log File Integrity Validation enabled, which generates an unforgeable checksum alongside every log file, preventing modification without detection. Even deletion of logs without detection is impossible, for two reasons: One, the checksum of each log is based partially on the previous file's checksum, and two, even if there is no operator activity, CloudTrail periodically generates a "no-op" checksummed log file, preventing the operator from claiming there are no logs because nothing has occurred. Verification of authenticity is discussed in the section below.
- **Comprehensive** - CloudTrail is enabled first, before bringing up any infrastructure, and it's never disabled, so logs cover the entire lifetime of the product. Any gaps would be detected by the verification mechanism below. For finding potentially dangerous operator actions, see the "Detect Dangerous Operator Actions" section below.

Log File Integrity Verification

AWS provides a manual way to check the integrity of CloudTrail logs against their checksums, but since there are tens of thousands of audit log files, Openly Operated built a free, open source Mac tool that automatically does this integrity check for CloudTrail logs in an AWS account, called Open Watch.

To use Open Watch, download the open source Open Watch tool and either use the precompiled build release, or follow the instructions in the README to build and run it. Then, use the following settings:

- Rules: Check all checkboxes
- Start: The start date you wish to run Open Watch from. Confirmed VPN's environment was initialized on November 18th, 2018, so you may use that date to do a comprehensive scan — be warned, this will take a long time.
- End: Today's Date
- AWS Key: `AKIAIV7RW2OU25DOM7EA`
- AWS Secret: `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`

Click "Audit" to begin. This may take a significant amount of time, since Open Watch is downloading tens of thousands of audit logs and verifying each of them, so it may make sense to launch this process as the last thing you do in the day, and verify the results in the morning (but do not allow your computer to sleep or hibernate). If any integrity issues are found, they'll be displayed in the main whitespace area after verification is complete. Please note that it's better to keep the app in the foreground during auditing, because macOS may heavily throttle applications that are in the background and cause the process fail.

Note: The AWS Key and Secret above are limited to only permissions necessary to perform the Open Watch audit — attempting to use them for any other API calls will fail.

A second option, if the Audit Trail download is not progressing as quickly as you like, is to download the .zip file in the Manual Analysis section below, unpacking it, and pointing Open Watch to the unpacked directory with the "Choose" button labeled "Optional: Choose local folder for logs" — this may be more efficient than querying the AWS API directly.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Detect Dangerous Operator Actions

To fulfill the Comprehensive section of the Audit Trail requirement, it can be difficult to sift through the thousands of CloudTrail logs to find potentially dangerous operator actions. Fortunately, the Open Watch tool above also performs detection of several common types of dangerous operator actions, displayed as "Rules" in the Open Watch interface, with explanations of each rule.

Manual Analysis

To manually analyze the infrastructure audit trail, download it below — warning: this file is extremely large and will take significant time to download and unpack.

Confirmed VPN Infrastructure Audit Logs - June 9th, 2019 - Download

Database Audit Trail

Confirmed VPN has a custom-built Administrator dashboard (the `Admin` repository) that allows the operator to send manual database queries, in order to perform the necessary administrative task of database migrations. However, because these queries are arbitrary, that could be abused by the operator to retrieve or modify user records, putting user privacy at risk. These queries are not tracked or logged by the Infrastructure Audit Trail (CloudTrail), so Confirmed VPN tracks this with the Database Audit Trail for later auditing.

Right before a database query is executed, the `Admin` server logs the query to two places: 1) to a Cloudwatch Logs LogGroup called AdminAudit viewable to the read-only account and auditors, in a Log Stream called PostgresQueries, accessible here (you will need an auditor read-only account to access this) and 2) to S3 in a log file prefixed with `PostgresQueries-` in a public bucket, accessible to everyone here. This way, the public and auditors get to see all database queries that the operator executes, keeping the operator honest.

- **Impartial** - The Database Audit Trail is generated by custom code in the `Admin` server, referenced above. The Infrastructure Audit Trail, along with the Establish Provenance section below, ensures that the referenced code is indeed the same code that's deployed in production.
- **Authentic** - Modification of Database Audit Trail log lines is not possible because CloudWatch Logs is not capable of modifying log lines. However, it is possible to delete the entire log stream or log group and re-create it, creating the illusion that there never were any log entries. This tampering can be detected by Open Watch, which searches the Infrastructure Audit Logs for the DeleteLogGroup and DeleteLogStream API calls.
- **Comprehensive** - The same verification for Authenticity can be used to ensure comprehensiveness. As long as the Database Audit Trail log stream and group haven't been deleted, they should contain all database queries issued since the creation of Confirmed VPN.

Detect Dangerous Operator Actions

If you're an auditor with a read-only account, use the read-only account to view the CloudWatch log stream `PostgresQueries` in the `AdminAudit` log group to see a complete list of all database queries manually executed by the operator. If you don't have a read-only account, check the public S3 bucket AdminAuditBucket for files prefixed with `PostgresQueries-` . To view the contents of individual files, just append the filename to the S3 bucket url. The queries displayed, if any, should be for database migrations or simple tests, not queries for extracting rows from the database. For example, a `CREATE TABLE` query is fine, but a `SELECT * FROM USERS` is not.

Actions Audit Trail (VPN Server Certificate Audit)

In addition to securing the VPN Server Certificate and its private key as described in the "VPN Server Certificate Security" section above, Confirmed logs the time and IP whenever the Server Certificate and private key is downloaded, since it should only be accessed by internal VPN servers.

When the VPN Server Certificate is accessed, the `Admin` server logs the query to two places: 1) to a Cloudwatch Logs LogGroup called AdminAudit viewable to the read-only account and auditors in a Log Stream called Actions, accessible here (you will need an auditor read-only account to access this) and 2) to S3 in a log file prefixed with `Actions-` in a public bucket, accessible to everyone here. This way, the public and auditors get to see all the IPs that have ever downloaded the VPN Server Certificate and private key.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

- **Impartial** - The Actions Audit Trail is generated by custom code in the Admin server, referenced above. The Infrastructure Audit Trail, along with the Establish Provenance section below, ensures that the referenced code is indeed the same code that's deployed in production.
- **Authentic** - Modification of Actions Audit Trail log lines is not possible because CloudWatch Logs is not capable of modifying log lines. However, it is possible to delete the entire log stream or log group and re-create it, creating the illusion that there never were any log entries. This tampering can be detected by Open Watch, which searches the Infrastructure Audit Logs for the DeleteLogGroup and DeleteLogStream API calls.
- **Comprehensive** - The same verification for Authenticity can be used to ensure comprehensiveness. As long as the Actions Audit Trail log stream and group haven't been deleted, they should contain all database queries issued since the creation of Confirmed VPN.

Detect Dangerous Operator Actions

If you're an auditor with a read-only account, use the read-only account to view the CloudWatch log stream `Actions` in the `AdminAudit` log group to see a complete list of all database queries manually executed by the operator. If you don't have a read-only account, check the public S3 bucket AdminAuditBucket for files prefixed with `Actions-` . To view the contents of individual files, just append the filename to the S3 bucket url. The logs displayed should indicate downloads only originating from internal network IPs, in the CIDR (IP Range) `172.16.0.0/12` , which are expected requests from VPN servers being initialized.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Establish Provenance

Provenance is proof that both the code and infrastructure presented match the code and infrastructure in the actual users' production environment.

Infrastructure Provenance

The infrastructure presented above matches the infrastructure in production because it's brought up by CloudFormation, which is AWS's "code-as-infrastructure" service.

To verify this, auditors with read-only accounts can use the Read-Only Account to log into the console and check the CloudFormation service in the `us-east-1` region. Under each stack, check the template tab and compare it with the infrastructure files in the CloudFormation public repository. Then, any differences between the template and the current infrastructure can be detected with the Detect Drift functionality.

If you don't have an auditor read-only account, all infrastructure actions are logged by the Infrastructure Audit Trail, so you can use the zipped logs under "Manual Analysis" to see everything that has been done to bring up and maintain the infrastructure. The logs are in standard JSON format.

Auditors should also verify that the domain that users are connecting to is indeed the one auditors are examining, by checking the Route 53, which is AWS's DNS service. Verify that the one of the nameservers in the confirmedvpn.com Hosted Zone matches the nameserver returned by a host nameserver lookup from your local machine. On macOS, the command is `host -t soa confirmedvpn.com` .

For further proof of infrastructure provenance and detailed analysis of the raw logs, download Confirmed VPN's full infrastructure audit logs above, under the "Infrastructure Audit Trail - Manual Analysis" section.

Source Code Provenance - Client Code

To verify provenance, as with any open source software, anyone can build and run their own client locally — the build instructions are provided in each client repository.

The source code in the client Github repositories linked in the Open Source section above are the exact repositories used to build the binaries that users download.

Source Code Provenance - Backend & Server

The source code presented above matches the source code in the production environment because they're both `git remote` s of the same repository, which is built and deployed without modification to production servers via CodePipeline.

Manual modification of source code in production is not possible, because direct access to the servers is disabled and blocked — see the "Open Infrastructure - Security & Encryption - No Direct Server Access" section above.

To verify provenance, use the Read-Only Account and examine the CodePipeline service. Check that the code in the CodeCommit matches the code referenced in the repository, and that the code is successfully deployed to the servers. CodePipeline and CodeDeploy also displays the history of all the previous code deployments, allowing verification of any previous version of source code in production.

Alternatively, the Open Watch tool automatically parses through the Infrastructure Audit Trail to find the latest deployments for each server type and links to their corresponding source code. When the Open Watch tool has completed its audit, auditors can simply click on the entries in the right hand column to download the exact source code deployed onto that server.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Update: July 2020

In order to re-audit Confirmed VPN (and now Lockdown Secure Tunnel) in July 2020, all updates to the Requirements since the previous audit in June 2020 are documented below. To update their previous audit, auditors review these changes to confirm that the requirements are still fulfilled to quality as Openly Operated. By doing so, auditors verify that the original privacy and security claims made under the "Claims With Proof" section are still valid and accurate.

Identify Operation

Most of the operation has stayed the same, except we added a new product: Lockdown. Lockdown has two components, a Firewall and a Secure Tunnel (VPN). The firewall is entirely on-device and of course open source, and the Secure Tunnel uses the same VPN backend as Confirmed VPN, with very minimal tweaks. All the new code and infrastructure are covered in the next sections.

Open Infrastructure

The main infrastructure stayed mostly the same, as we didn't run into any scalability or other issues with our existing infrastructure. There are a few exceptions, highlighted below.

Create the Lockdown site @ lockdownhq.com

- We initially put this site in its own, independent environment, but then realized that the better choice was to put it in the same environment as Confirmed VPN, since Lockdown Secure Tunnel uses Confirmed's backend. And so if we wanted to send emails (such as Email Confirmation, Forgot Password, etc), we would want to send them from a @lockdownhq.com sender, instead of @confirmedvpn.com, so that the user isn't confused. We didn't want the domain ownership of lockdownhq.com to be split between two different environments, so we put the Lockdown site in Confirmed VPN's environment.
- For now, the site itself is completely static and doesn't access any user data and doesn't access the database. commit

Create the Privacy Review site @ privacyreview.co

- This is a simple website that shows the tracking behavior of popular apps. It reads from the `reviews` and `trackers` database tables, which only contain data that we manually entered to review popular apps - basically a blog, except more structured. These tables do not contain any user data.
- The infrastructure bringup code is available here: commit

Migrate VPN Instances from EC2 to Lightsail

- We moved VPN instances from EC2 to Lightsail because Lightsail instances were much less complex to manage. During our initial launch, we chose EC2 because Lightsail lacked specific health monitoring capabilities, such as CPU utilization, network utilization, and alarms. Without a way to automatically monitor health of production instances, we couldn't use Lightsail. In February 2020, Lightsail added built-in capability for resource monitoring and alarms. This allowed us to eliminate dozens of custom metrics and alarms that we had to manage for our EC2 instances, and switch to Lightsail.
- We designed this transition to comply with the same Openly Operated tenets. We disable SSH ports before the instance is accessible, and to be extra safe, we also automatically delete all SSH keys on the instance on launch, and also disable the SSH service. This can be verified in two ways:
  - The audit account has access to Lightsail, which can verify that the SSH ports are disabled
    - Due to a limitation in AWS, audit accounts cannot view Lightsail information through the AWS Console, and instead must view Lilghtsail information using the AWS CLI. See Configuring AWS CLI. After configuring the CLI, you can use the command `aws lightsail get-instances` to view the Lightsail instances and their open ports to confirm port 22 (SSH) is not accessible.
  - The initial launch script for these instances, which disable the SSH service and remove the instance's SSH keys, is logged in the tamper-proof CloudTrail audit logs. The updated Open Watch v1.0.4 has an updated "Safe Bringup" check, which scans for Lightsail's `CreateInstances` API call, and ensures that the `userData` (launch script) disables SSH and removes the SSH keys. Documentation Reference

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Open Source

All code changes can be viewed on our public repositories, but they are also summarized below in more detail.
Auditors: Verify the code, and also ensure the commit hash in CodePipeline matches the public commit hash.

Server Repository: `Admin` `commit 1b92c011a519269d21fee3eab9360523f23797e8`

This repository saw a few minor bugfixes, and a creation of an admin dashboard for the new Privacy Review website at privacyreview.co. To inform our existing users about Privacy Review, we built a privacy-preserving email sending system which doesn't allow us to view user email addresses directly when sending emails. It also respects do-not-email/unsubscribe requests, as well as bounces and complaints.

- commit - BUGFIX: Fix issue with updating PM2, which is our process manager for Node.js.
- commit - UPDATE: Update NPM dependencies
- commit - ENHANCEMENT: Allow issuing commands to the Redis database to balance mitigation of potential brute force attacks with server performance.
- commit - SECURITY: Additional layer of security preventing operator from accessing certificate secrets
- commit - FEATURE: Create an admin backend to manage the new Privacy Review website at privacyreview.co
- commit - FEATURE: Process email bounces and complaints from users who receive our newsletter email about Privacy Review
- commit - FEATURE: Ability to send emails to users about updates to Lockdown, without viewing users' actual email addresses so as to preserve privacy
- commit - PRIVACY: Don't log successful email sends (no email campaign had been sent out at this point yet, only test emails)
- commit - ENHANCEMENT: Ability to set do-not-email for users from Admin, for users who didn't want to click the "unsubscribe" button themselves
- commit - UPDATE: Update NPM dependencies

Server Repository: `Bandwidth` `commit 54eb77969a19c9e6c0caef13210008eb7ac6003d`

This repository was mostly untouched as it is internal-only and not public or customer-facing.

- commit - UPDATE: Update NPM dependencies

Server Repository: `Helper` `commit e89252a4a5d64a0bc6553d3605e4611490b20225`

This repository was mostly untouched as it is internal-only and not public or customer-facing.

- commit - UPDATE: Update NPM dependencies

Server Repository: `LD-Main` `commit 1bfea51ebac11ce187b56c3c3f9271fe493162c7`

This is a new repository for the Lockdown landing page. It's a static website that has no access to any user data. It links to the Lockdown iOS and Mac apps, as well as press mentions of the app.

- View the repository.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Server Repository: `Main` `commit 474d9470dc318b61101b6820aac7b1b5f77611dc`

This repository saw various bug fixes, site updates, but the main change was adding support for Lockdown Secure Tunnel.

- commit COPY: Fix links to Audit Kits
- commit UI: Refresh home page and other static pages
- commit BUGFIX: Don't send cancellation email if user has other active subscription
- commit UI: Update meta-image, screenshots, contacts
- commit BUGFIX: Fix issue with updating PM2, which is our process manager for Node.js.
- commit STORAGE: Rotate PM2 logs so they don't fill up the disk. These logs do not contain any personally identifiable information.
- commit ENHANCEMENT: Support password autofill on iOS
- commit FEATURE: Support Lockdown Secure Tunnel
- commit UI: Copyright year update
- commit BUGFIX: Confirm Email would sometimes error on "Missing Email"
- commit BUGFIX: Support + sign in emails
- commit BUGFIX: Count "_" as special character in password
- commit BUGFIX: Use the newest subscription as the correct one instead of just choosing the first one
- commit FEATURE: Unsubscribe to update emails
- commit BUGFIX: Don't automatically redirect to app on confirm email, in case the user does not yet have the app installed or is on a different device
- commit UPDATE: Update NPM dependencies

Server Repository: `Partner` `commit 8a5861acd30ffbc8e4d1b5dd4b169338c43b04b4`

This repository was for our referral program, which is no longer in use and has been decommissioned.

- commit BUGFIX: Fix issue with updating PM2, which is our process manager for Node.js.
- commit UPDATE: Update NPM dependencies

Server Repository: `PR-Main` `commit 345b47b1c1a90e0d5ad4bdcacbae919a8fedd373`

This is a new repository for the Privacy Review website. The site, located at privacyreview.co only has access to Privacy Review's reviews, and it never accesses any user data.

- View the repository.

Server Repository: `Renewer` `commit d2e4ace4fc11527f5c642920c76c49635d60cf76`

This repository was mostly untouched as it is internal-only and not public or customer-facing.

- commit BUGFIX: Fix issue with updating PM2, which is our process manager for Node.js.
- commit UPDATE: Update NPM dependencies

Server Repository: `Shared` `commit eb33fa1d2d8182fcae6f90d1d41a5e82294e4a06`

Since this is the shared or "common" code between the other repositories, changes here are mostly changes to support the changes in other repositories.

- commit BUGFIX: Don't send cancellation email if user has other active subscription
- commit UPDATE: Support Google Play Store (Android users) v3 API
- commit ENHANCEMENT: Support Lockdown Secure Tunnel
- commit BUGFIX: Support + sign in emails
- commit UPDATE: Support sending emails from the Lockdown domain
- commit FEATURE: Support Privacy Review site
- commit FEATURE: Support sending emails about updates to Lockdown
- commit UPDATE: Update NPM dependencies

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Server Repository: `Support` `commit bf41a81a180c76ef3539a76d1b3444b0cb84c34d`

This repository is for servicing customer support requests, and is internal-only, so didn't have many changes.

- commit BUGFIX: Fix issue with updating PM2, which is our process manager for Node.js.
- commit UPDATE: Update NPM dependencies

Server Repository: `Webhook` `commit 1b6cd7372bb0a5c40bf2ad6bdff543710bc3b1db`

This repository is only open to Stripe server IPs. Only very minor changes were made to it.

- commit UPDATE: Deprecate referral program
- commit BUGFIX: Fix issue with updating PM2, which is our process manager for Node.js.
- commit UPDATE: Update NPM dependencies

Client Repository: `Confirmed-iOS`

There were minimal changes to the iOS Confirmed VPN app since the previous audit.

- commit COMPATIBILITY: iOS 13 update and whitelist update
- commit PRIVACY: Remove pasteboard usage in anticipation of iOS 14. Pasteboard was only used for now-deprecated referral program. To be sure of user privacy, it only used the pasteboard if its contents started with a unique code ("266347633"), so any user data from the pasteboard was never copied.

Client Repository: `Confirmed-Mac`

There were minimal changes to the Mac Confirmed VPN app since the previous audit.

- commit BUGFIX: Fix issue with whitelist where entries would be out of order
- commit BUGFIX: Fix issue where clicking a different region would sometimes not dismiss the country list

Client Repository: `Confirmed-Android`

There were minimal changes to the Android Confirmed VPN app since hte previous audit.

- commit BUGFIX: Compatibility with newer Android versions

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Client Repository: `Confirmed-Windows`

There were no changes to the Windows Confirmed VPN app.

Client Repository: `Lockdown-iOS`

This is the new Lockdown app for iOS. It has two main features: Firewall, and Secure Tunnel (VPN). The Firewall blocks trackers, ads and badware for all apps, and to maximize user privacy, does this entirely on-device. The Secure Tunnel uses the same backend VPN servers as Confirmed VPN. Like Confirmed iOS, Lockdown is privacy-first and security-first:

- **No User Tracking** - Many apps today harvest user data (location, contacts, web traffic, metadata, screen capture, etc) for profit, analytics, and advertisement targeting. Lockdown iOS contains no user data harvesting, and uses no third-party analytics libraries or APIs, eliminating this exposure of user data. All of the libraries used by Lockdown iOS are open source, and listed here and here.
- **Secure API Connections** - Lockdown iOS makes HTTPS connections to the Main server API, which encrypts connection data and also reduces the risk of man-in-the-middle attacks.

Client Repository: `Lockdown-Mac`

This is the new Lockdown app for Mac. It's almost identical in functionality to Lockdown iOS, and also mirrors its UI. Like Confirmed Mac, Lockdown Mac is privacy-first and security-first:

- **No User Tracking** - Many apps today harvest user data (location, contacts, web traffic, metadata, screen capture, etc) for profit, analytics, and advertisement targeting. Lockdown iOS contains no user data harvesting, and uses no third-party analytics libraries or APIs, eliminating this exposure of user data. All of the libraries used by Lockdown iOS are open source, and listed here.
- **Secure API Connections** - Lockdown Mac makes HTTPS connections to the Main server API, which encrypts connection data and also reduces the risk of man-in-the-middle attacks.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Audit Trail

Infrastructure Audit Trail

We still verify our Infrastructure Audit Trail by using the Open Watch tool to automate validation of CloudTrail logs. Simplified instructions below:

1. Download the .zip file of the audit logs by clicking "Download" below, since it's much faster to download one file and decompress it, instead of downloading tens of thousands of files separately.
   Download - Infrastructure Audit Logs - July 23rd, 2020
2. Decompress the downloaded file into a folder.
3. Get the updated Open Watch v1.0.4 either through compiling its source or using the precompiled build release.
4. Open Open Watch and use the following settings:
   - Rules: Check all checkboxes
   - Start: June 9th, 2019 (This was the date of the previous audit)
   - End: July 23rd, 2020
   - AWS Key: `AKIAIV7RW2OU25DOM7EA`
   - AWS Secret: `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`
5. Click "Choose" in Open Watch and select the Infrastructure Audit Logs you decompressed.
6. Click "Audit" to begin the audit.

Remember that this may take a significant amount of time, so it may make sense to launch this process as the last thing you do in the day, and verify the results in the morning (but do not allow your computer to sleep or hibernate). If any integrity issues are found, they'll be displayed in the main whitespace area after verification is complete. Please note that it's better to keep the app in the foreground during auditing, because macOS may heavily throttle applications that are in the background and cause the process fail.

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	

Database Audit Trail

Since operators don't have direct access to the database, in order to make schema updates to the database, operators use an Admin tool to send commands. The Admin tool records all the database commands that are issued to ensure the operator does not access user data. The Database Audit Trail is available to auditors here, and we've copied its contents in chronological order below, along with explanations and justification for each command.

- Create a dashboard for referral partners to view referral performance as anonymized, aggregated metrics. The referral program has been deprecated and is no longer in use.
    - `CREATE TABLE partners ( id SERIAL, title text NOT NULL, code text NOT NULL, percentage_share smallint NOT NULL ); ALTER TABLE ONLY partners ADD CONSTRAINT partners_code_pkey PRIMARY KEY (code);`
    - `CREATE TABLE partner_snapshots (id SERIAL, create_date timestamp with time zone DEFAULT now() NOT NULL, partner_code text NOT NULL, summary text NOT NULL, campaigns text NOT NULL); ALTER TABLE ONLY partner_snapshots ADD CONSTRAINT partner_snapshots_partner_code_fkey FOREIGN KEY (partner_code) REFERENCES partners(code) ON UPDATE CASCADE;`
    - `ALTER TABLE users ADD COLUMN partner_campaign text;`
    - `CREATE TABLE partner_users (id SERIAL, email text NOT NULL, password text NOT NULL, partner_code text NOT NULL); ALTER TABLE ONLY partner_users ADD CONSTRAINT partner_users_email_pkey PRIMARY KEY (email); ALTER TABLE ONLY partner_users ADD CONSTRAINT partner_users_partner_code_fkey FOREIGN KEY (partner_code) REFERENCES partners(code) ON UPDATE CASCADE;`
    - `CREATE USER partner WITH ENCRYPTED PASSWORD 'PG_PARTNER_PASSWORD'; GRANT SELECT(id, create_date, referred_by, delete_date, delete_reason, banned, email_confirmed, partner_campaign) ON users TO partner; GRANT SELECT(user_id, receipt_id, receipt_type, plan_type, expiration_date, cancellation_date, in_trial, failed_last_check, updated) ON subscriptions TO partner; GRANT SELECT ON partners TO partner; GRANT SELECT ON partner_users TO partner; GRANT SELECT ON partner_snapshots TO partner; GRANT UPDATE(password) ON partner_users TO partner;`
    - `SELECT partner_campaign FROM users WHERE partner_campaign IS NOT NULL;`
    - `SELECT count(*) FROM subscriptions INNER JOIN users ON (subscriptions.user_id = users.id) WHERE users.partner_campaign IS NOT NULL AND subscriptions.expiration_date > now() AND subscriptions.cancellation_date IS NULL;`
    - `SELECT partner_campaign FROM subscriptions INNER JOIN users ON (subscriptions.user_id = users.id) WHERE users.partner_campaign IS NOT NULL AND subscriptions.expiration_date > now() AND subscriptions.cancellation_date IS NULL;`
- Add columns to subscription table to support feature to send subscription cancellation confirmation emails to users
    - `ALTER TABLE ONLY subscriptions ADD COLUMN expiration_intent_cancelled boolean DEFAULT false NOT NULL;`
    - `ALTER TABLE ONLY subscriptions ADD COLUMN sent_cancellation_email boolean DEFAULT false NOT NULL;`
    - `GRANT SELECT(id, email, email_encrypted, stripe_id, create_date, referred_by, delete_date, delete_reason, banned, month_usage_megabytes, month_usage_update, email_confirmed, do_not_email, do_not_email_code) ON users TO renewer;`

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	

- Get the total count of new users in various time periods (count only, no actual data retrieved).
  - `select count(*) from users where create_date > now() - INTERVAL '1 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '1 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '12 HOUR'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '12 HOUR'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '1 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '1 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '12 HOUR'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '4 DAY' AND create_date < now() - INTERVAL '3 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '1 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '12 HOUR'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '1 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '12 HOUR'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '12 HOUR'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '1 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '1 DAY'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '12 HOUR'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '12 HOUR'`
  - `SELECT count(*) FROM users WHERE create_date > now() - INTERVAL '1 DAY'`
- Add ability to distinguish between a Confirmed VPN signup and Lockdown signup
  - `ALTER TABLE users ADD COLUMN lockdown boolean DEFAULT false NOT NULL;`
- Create tables for Privacy Review website's content
  - `CREATE TABLE reviews (id serial PRIMARY KEY, name text NOT NULL UNIQUE, display_name text NOT NULL, tagline text NOT NULL, num_trackers integer NOT NULL, num_attempts integer NOT NULL, rating text NOT NULL, date date NOT NULL DEFAULT now(), platforms text NOT NULL DEFAULT 'iOS', ranking text, icon_url text, disclaimer text, data_required_info text[] NOT NULL DEFAULT '{}', data_required_access text[] NOT NULL DEFAULT '{}', data_optional text[] NOT NULL DEFAULT '{}', screenshot_url text, test_method text NOT NULL, test_description text NOT NULL, test_open text NOT NULL, test_consent text NOT NULL, test_background text NOT NULL, test_notes text NOT NULL, summary_url text, published boolean NOT NULL DEFAULT false);`
  - `CREATE TABLE trackers ( id serial PRIMARY KEY, name text NOT NULL, display_name text NOT NULL, tagline text NOT NULL, categories text[] NOT NULL DEFAULT '{}', connections text[] NOT NULL DEFAULT '{}', collected_data text NOT NULL );`
  - `CREATE TABLE reviews_trackers ( id SERIAL PRIMARY KEY, review_id integer NOT NULL REFERENCES reviews(id) ON DELETE CASCADE, tracker_id integer NOT NULL REFERENCES trackers(id) ON DELETE CASCADE );`
  - `GRANT SELECT ON trackers TO main; GRANT SELECT ON reviews_trackers TO main; GRANT SELECT ON reviews TO main;`
  - `ALTER TABLE reviews DROP COLUMN data_required_access;`
  - `ALTER TABLE reviews DROP COLUMN data_optional;`
  - `ALTER TABLE reviews ADD COLUMN rating_reason TEXT; ALTER TABLE reviews ALTER COLUMN test_notes DROP NOT NULL; ALTER TABLE reviews ALTER COLUMN test_background DROP NOT NULL;`
- Retrieve count of total users (no data, only a single count).
  - `select count(*) from users where email_confirmed = true;`
  - `select count(*) from users where lockdown = true and email_confirmed = true`
  - `select count(*) from users where email_encrypted is not null and email_confirmed = false`
- Add capability to send user updates via email, and capability to user to instantly unsubscribe.
  - `ALTER TABLE users ADD COLUMN newsletter_subscribed boolean NOT NULL DEFAULT true;`
  - `ALTER TABLE users ADD COLUMN newsletter_unsubscribe_code text NOT NULL DEFAULT upper("substring" (md5(random()::text), 0, 20));`
  - `CREATE TABLE campaigns ( id SERIAL PRIMARY KEY, name text NOT NULL, from_address text NOT NULL, subject text NOT NULL, html text NOT NULL, plaintext text NOT NULL, create_date timestamp without time zone NOT NULL DEFAULT now(), last_sent_date timestamp without time zone );`
  - `CREATE TABLE campaign_emails ( id SERIAL PRIMARY KEY, campaign_id integer NOT NULL REFERENCES campaigns(id), email_encrypted text NOT NULL, unsubscribe_code text NOT NULL, sent boolean NOT NULL DEFAULT false, failed boolean NOT NULL DEFAULT false );`
  - `CREATE UNIQUE INDEX campaign_id_email_unique ON campaign_emails(campaign_id int4_ops,email_encrypted text_ops);`

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	
	Choose Result ▼	

Establish Provenance

Establishing provenance provides verification that the server and client code are indeed what is published publicly in our repositories. For our existing server and client repositories, this has not changed from before. For the newer Lockdown and Privacy Review repositories, the same verification steps can be used from the original audit's Establish Provenance section

Describe Verification (Required)	Result (Required)	Comments/Issues/Questions
	Choose Result ▼	

Issues, Comments, Questions Not Addressed In Audit Kit