VNC® SDK security whitepaper

Version 1.3

Contents

Introduction

Customer security is of paramount importance to RealVNC®. As such, our security strategy is ingrained in all aspects of the VNC® SDK. We have invested extensively in our security, and take great pride in our successful track record.

In order to explain our strategy in more detail, this document is divided into four key topics: security architecture, cloud infrastructure, client security, and development procedures. Each are guided by four simple, but powerful, principles:

- You do not have to trust RealVNC in order to trust our software and services.
- We do not store your session data, and cannot decrypt it now or in the future.
- Every session is treated as though it is made in a hostile environment.
- The remote computer ultimately decides who is able to connect.

A note on terminology

The VNC SDK enables developers to create remote control apps. Typically, these consist of a **Server app** for the computer to be controlled, and a **Viewer app** for the device to control from. Together, these apps are known as **clients**.

Security architecture

Overview

At RealVNC we make a fundamental security assumption that the connection between a Viewer app and a Server app may traverse a hostile environment, and we have built our entire security architecture with this in mind. As such, the security features and strategies described below apply equally to cloud (‘VNC Cloud’) and direct (‘direct TCP’) connections.

When making a VNC Cloud connection, RealVNC cloud services only broker approved connections and act as a fallback transport mechanism for encrypted data. They explicitly **do not** enjoy a privileged security position. A developer can ensure that ultimate responsibility for authorizing remote access lies solely with the Server app running on computers to be controlled.

When making a direct TCP connection, a Viewer app performs no interaction with RealVNC cloud services at all (neither client needs access the Internet). Again, the Server app is ultimately responsible for granting authorized remote access.

As a developer, you can choose whether clients make VNC Cloud connections, direct TCP connections, or a mixture of both. For a summary, see this paper on cloud versus direct (written for VNC Connect, with which the VNC SDK shares a code base and the RealVNC cloud services).

Remote Framebuffer Protocol (RFB)

VNC technology uses the Remote Framebuffer Protocol, an Internet Standard protocol originally created by RealVNC as the first remote desktop protocol. RealVNC continues to actively maintain this and in June 2015 RFB version 5 was released, designed from the ground up to support cloud connectivity.

RFB 5 mandates the use of modern cipher suites and uses strong cryptography throughout. It is streamlined compared to TLS, making it much less prone to implementation vulnerabilities and misconfiguration. It offers very strong key exchange that is designed for cloud connectivity, by mixing in three sources of key material: the local client, the remote client and the cloud handshake. This puts the clients in full control of encryption keys, preventing tampering.

A detailed security analysis of RFB 5 can be found here.

Encryption

The VNC SDK uses AES-GCM encryption to ensure the secrecy and integrity of data while in transit. All encryption is end-to-end, ensuring no one can read the data in transit, including RealVNC.

Elliptic Curve Diffie-Hellman key exchange adds Perfect Forward Secrecy (PFS) on top of the RSA key exchange used for identity checking (refer to the *Identity checking* section, below). PFS provides a shared secret for each individual connection that can only be recovered while the connection is active, preventing leaked keys from being used to decrypt past or future connections.

Identity checking

The RealVNC identity-checking model allows a Viewer app to know it is connecting to the correct Server app, preventing impersonation. Each client is identified by an RSA key according to two different security models:

- For direct TCP connections, a “Trust on First Use” (TFU) model is employed. In this case, a developer can configure a Viewer app to mandate that the key is manually checked, and store keys to ensure that have not changed on reconnection. This guarantees that a malicious party has not intercepted the connection.
- For VNC Cloud connections, RealVNC cloud services assist in key exchange, meaning a Viewer app need not check the key manually. However, there is no need to trust the automated identity check. A developer can still configure a Viewer app to mandate that the key *is* manually checked.

When performing manual identity checking, a Server app’s identity is presented as an easily recognisable catchphrase. This makes it simple for users to notice changes to the key, preventing impersonation. The catchphrase is derived from the Server’s RSA key and is a human-readable representation of the key’s fingerprint.

Cloud negotiation process

VNC Cloud connections enable easy access to machines through NAT and firewalls. They do this by using an outbound connection to RealVNC cloud services from both Viewer and Server app. These services broker the connection, enabling them to exchange IP addresses for peer-to-peer connectivity, and may also relay data in the event that peer-to-peer connectivity is not possible. Once the connection is established, the same RFB security is applied, regardless of the transport mechanism. The clients always assume the transport could be hostile, even if transported through our services. RealVNC has no ability to read or tamper with any connection.

By avoiding the use of inbound TCP or UDP packets, clients do not require any ports to be opened on your firewall. This prevents unauthenticated access to your network from outside.

Negotiation of peer-to-peer connectivity

When using peer-to-peer connectivity with VNC Cloud connections, the standard Interactive Connectivity Establishment procedure (ICE) is employed. This involves opening a UDP port on each client, performing IP address discovery using STUN, and then sending packets directly to the peer. UDP ports are only opened by a Server app as needed when a connection is made, and are associated with a specific connection, reducing the attack surface to a minimum.

This operates at a lower level than RFB, so a separate security mechanism is used to sign every UDP packet, using a per-connection secret. This prevents attacks against the UDP port and interception of the connection at a lower level than the RFB data. In between the ICE and RFB layers, SCTP is used to add reliability to the data stream.

Cloud infrastructure

Overview

Our philosophy for VNC Cloud connections is that no one security measure is paramount. In other words, our security is layered; if RealVNC cloud services were to be breached, or if a malicious third party were to gain access to your RealVNC account, then providing the developer has implemented Server-side authentication checking, that party still cannot actually remotely access computers.

RealVNC cloud services

The VNC SDK initiates and maintains a VNC Cloud connection using our system of online services, which were designed in-house alongside our security team. These run on a security-hardened platform.

These services are protected by mandatory TLS, and all endpoints are suitably authenticated. All clients use TLS 1.2 when communicating with cloud services. We use rate-limiting and geographical distribution to defend against denial of service attacks. As our distributed architecture has no master nodes, our services are highly available. This ensures that service to clients is maintained during upgrades and failover. Data is replicated in real time to multiple data centers, and frequent database backups are kept on encrypted storage for recovery purposes.

Operations

Our technical operations team monitors RealVNC cloud services 24 hours a day, 365 days a year. These systems are regularly patched. Any critical vulnerabilities in upstream dependencies are assessed and patched outside of our regular patching schedule.

Access to service data is tightly controlled and granted on a case-by-case basis, with the approval of senior RealVNC management required. Every change made by the team is logged for audit purposes.

Servers containing sensitive data use host-based intrusion detection. If someone were to gain unauthorized access, automated alerts would be raised. Automated external port scanning is regularly performed on all Internet-facing servers. The output of this is reviewed regularly by our security team.

Public Key Infrastructure

Intra-service communication is secured using a closed Public Key Infrastructure (PKI). Industry best practice is used in the internal operation of this PKI, and best-in-class security devices (including hardware security modules and 256-bit AES encrypted hard drives) are used. By controlling the chain of trust end-to-end, we guarantee this is tightly controlled and not reliant on third party root certificate authorities.

Critically sensitive data is never stored in the cloud

A Server app does not delegate authentication to the cloud. Instead, it performs its own authentication checks, meaning that any breach of data from the cloud cannot be used to gain access to remote computers.

Your RealVNC account credentials are stored securely, using bcrypt hashes with a random salt. If you choose to delete your account, your details are completely removed from our servers.

RealVNC account management portal

Our online portal can be used to generate API keys and add-on codes for the VNC SDK.

Access to the portal itself is protected by mandatory TLS. Our website is graded A in the Qualys SSL Labs test.

We follow all best practices for secure web development, including using secure cookies and providing protection against content injection and cross-site scripting (XSS). To avoid session misuse, signed in users are automatically signed out after a period of inactivity.

Multi-factor authentication for your RealVNC account

By default, your RealVNC account is secured using email as a second factor. Each time you sign in online from a new device at a new location, you'll get an email requiring you to confirm the activity was by you. The email records the time, location and type of device attempting to access your account. This ensures that people cannot sign in to your account even if they discover or guess your RealVNC account credentials (email address and password).

You can further protect your RealVNC account by turning on 2-step verification on the **Security** page of the online portal. This means that each time you sign in online you are required to enter a TOTP token (an Internet standard used by Google Authenticator and similar apps) in addition to your account credentials. Effectively, TOTP replaces email as the second factor, and provides a higher level of security by requiring confirmation for every sign-in.

Client security

Overview

The VNC SDK is designed to delegate access control to a Server app running on the computer to be controlled. By contrast, some solutions use a gateway appliance or control mechanisms in the cloud to verify and grant access to a remote session. RealVNC never defers access control to an appliance or the cloud. Since the Server app remains in control at all times, the overall attack surface is lowered.

Audit logging

A developer can configure Viewer and Server apps to audit any quality of logging, to any destination. These logs can be used to inspect and audit individual connections and are extremely useful in supporting corporate compliance and regulatory governance requirements.

Blacklisting

A developer can configure a Server app to blacklist connecting Viewer apps failing to authenticate a particular number of times, to protect against brute force password attacks.

IP address filtering

For direct TCP connections, a developer can configure a Server app to prevent connections from particular IP addresses.

Cloud security transparency through identity checking

For VNC Cloud connections, it is not possible for an actor within RealVNC to mount a passive attack on relayed data, since all sessions are encrypted end-to-end. However, any cloud service architecture that performs key negotiation on behalf of its clients may be able to mount an active attack by exchanging false keys. This attack would enable impersonation of a Server app.

To protect against an attack from within RealVNC, the identity used to protect end-to-end encryption is made available to the VNC SDK, in the form of a memorable catchphrase. A developer can configure a Viewer app to mandate that this catchphrase is manually checked.

At RealVNC, we are transparent about our cloud security, and an actor within RealVNC cannot mount an attack against a client without leaving an auditable trace.

Gatekeeping

If a local user (that is, of a Server app) is likely to be physically present, a developer can configure the Server app to prompt that user to approve or reject each connection.

Multi-factor authentication for a Server app

A Server app is *always* responsible for authenticating Viewer apps requesting connections. This is true for both VNC Cloud and direct TCP connections. Authentication is never delegated to the cloud.

By default, connections are rejected unless a developer configures a Server app to request a user ID and password, and for a Viewer app to supply them. These credentials can be checked against any password store.

Contact RealVNC for information on APIs that will allow multi-factor authentication by a range of different mechanisms.

Session permissions

A developer can configure a Server app to grant session permissions to a Viewer app once authentication is complete.

For example, particular users or groups could be prevented from printing, transferring files, or copying and pasting text. Or entire sessions could be made view-only.

Root certificate pinning

When establishing a VNC Cloud connection, the clients check the TLS root (CA) certificate against a whitelist. This limits exposure to certificates issued by other certificate authorities in the OS trust store.

Development procedures

Overview

Our dedicated security team has been constantly involved during development of the VNC SDK, and remains actively involved in the decision-making process post-release.

Pre-development

We initially established a set of security and privacy requirements for the VNC SDK to adhere to. Our fundamental security principles – that you don't have to trust *us* in order to trust our software, that a Server app is in charge of any connection, that we cannot decrypt your connection data now or in the future, and that any active connection must be considered hostile at all times – were established during these early discussions. Exhaustive security and privacy risk assessments were then carried out, using threat modelling.

Any hardware used for VNC Cloud connectivity is owned and operated solely by RealVNC, and stored in remote data centers which meet today's strict industry certification and audit requirements. Only nominated RealVNC staff have the capability to configure this hardware. This helps keep any attack surface to a minimum.

Development

Our security team are constantly analysing our code, and continue to review new lines of code as new features are added. Code scanning tools, as well as manual static analysis, are used to ensure this code is kept to the highest possible security standards. Code reviews ensure that developers do not use functions or coding practices that are considered unsafe.

The code review process includes automated white-box fuzz testing, dynamic analysis using tools such as Valgrind and AddressSanitizer and a continuing attack surface review of the VNC SDK.

Release / post-development

Our code is subject to a final security review before each new release. We then archive all pertinent data. This is essential in order for us to perform post-release servicing tasks.

Every aspect of the VNC SDK has been designed with the highest level of security in mind. Despite this, it is incredibly important that any software vendor has an incident response plan. Because we understand that no security measure is unbeatable, we can and will release updates that protect our product from threats that emerge following its release.

This process is based on Microsoft's SDL Model. You can read more about this model at www.microsoft.com/sdl.

For over a decade, we have used CVE to archive any public-facing security vulnerabilities that we have fixed. We will continue to do so for the VNC SDK, in order to remain as transparent as possible to our customers.

Summary

The security features and processes that protect the VNC SDK are part of an in-depth strategy. This document has attempted to explain this strategy by separating it into four key categories.

We trust this has helped shed some light on our security and illustrated the integrity of our approach. However, given the inherent complexity of software security, we understand you may have specific enquiries. Please don't hesitate to contact us with your questions. We're only too happy to put you in touch with the appropriate member of our development team.

If you have any questions about the topics raised in this white paper, please contact us at enquiries@realvnc.com or visit realvnc.com/contact-us.

RealVNC