Getscreen.me

Security Statement

This document describes the security features and principles used to protect remote access to devices through Getscreen.me software. It covers the security measures applicable to both the SaaS and Self-Hosted versions of the product.

November 16, 2026

Table of contents

1. Security Statement 5
1.1 Introduction 5
1.2 Product and Vendor Information 5
1.3 Security Architecture and Principles 5
1.3.1 Deployment Models 5
1.3.1.1 Cloud (SaaS) 5
1.3.1.2 Self-Hosted 6
1.3.2 Shared Responsibility Model 6
1.4 Connection Security 6
1.4.1 Connection Participants 6
1.4.1.1 Device 7
1.4.1.2 Technician 7
1.4.1.3 Server 7
1.4.2 Connection Establishment and Data Transmission 7
1.4.2.1 WebRTC Channel 8
1.4.2.1.1 Host 8
1.4.2.1.2 Server-Reflexive 9
1.4.2.1.3 Relay 10
1.4.2.2 WebSocket Channel 12
1.4.3 Data Encryption 12
1.4.4 Network Requirements and Port Filtering 13
1.5 Authentication and Account Security 13
1.5.1 Password Security 13
1.5.2 Two-Factor Authentication (2FA) 14
1.5.3 Login by Client Certificate (mTLS) 14
1.5.4 Single Sign-On (SSO/SAML) 14
1.5.5 External Authentication (LDAP) 14
1.5.6 IP Address Whitelisting 14
1.5.7 Dashboard Session Control 15
1.5.8 Automatic Logout 15
1.5.9 Inactive Account Removal 15
1.6 Access Control and Session Security 15
1.6.1 Access Rights and Roles 15
1.6.2 Device Access Permissions 15
1.6.3 Feature Restrictions 16

1.6.4 Session Recording 17
1.6.5 Team Audit Logs 18
1.6.6 Access Link Expiration 18
1.7 Remote Device Security 18
1.7.1 Code Signing 18
1.7.2 Software Update Security 19
1.7.3 Function Transparency 19
1.7.4 Group Policy 19
1.7.5 Black Screen and Input Blocking 20
1.7.6 Session Duration Limits 20
1.7.7 Inactive Session Termination 20
1.7.8 Active Connection Indicator 20
1.8 Server Security 20
1.8.1 CAPTCHA 20
1.8.2 HSTS (HTTP Strict Transport Security) 20
1.8.3 CSP (Content Security Policy) 21
1.8.4 Brute Force Intrusion Protection 21
1.9 Data Security and Privacy 21
1.9.1 Data Minimization 21
1.9.2 Data Access Control 22
1.9.3 Data Storage and Retention 22
1.9.4 Data Deletion 22
1.9.5 Data Residency 23
1.9.6 Backups and Recovery 23
1.10 Infrastructure Security 23
1.10.1 Reverse Proxy 24
1.10.2 Server Hardening 24
1.10.3 Intranet (LAN-Only) Deployment 24
1.10.4 Infrastructure Monitoring 24
1.10.5 Data Center Security 24
1.11 Secure Development and Vulnerability Management 25
1.11.1 Secure Development Practices 25
1.11.2 Vulnerability Management 25
1.11.3 Security Updates 26
1.12 Security Incident Response 26
1.13 Security Assurance and Compliance 27
1.13.1 Regulatory Compliance 27

1.14 Security Contact and Related Documents 27
1.14.1 Security Contact 27
1.14.2 Related Documents 28

1. Security Statement

1.1 Introduction

This document describes the security features and principles used to protect remote access to devices through Getscreen.me software. It covers the security measures applicable to both the SaaS and Self-Hosted versions of the product.

Security features and their availability may vary depending on the deployment model, subscription plan, operating system, and software version.

1.2 Product and Vendor Information

Getscreen.me is remote desktop software that allows users to remotely access and control computers and mobile devices over the Internet or a local network. It is designed for corporate network administration, technical support, and personal use.

The product has been available on the market since 2020.

Getscreen.me is developed by Point B Ltd, a private limited liability company based in Limassol, Republic of Cyprus. The Getscreen.me trademark is protected under applicable intellectual property laws in the jurisdictions where it has been granted legal protection.

Public company information:

> **Point B Ltd**
> Address: Vasileiou Vryonidi, 6, Gala Court Chambers, Floor 3, Office 304C, 3095, Limassol, Cyprus
> Registration Number: HE 430957
> VAT Number: CY10430957Q
> Email: info@getscreen.me

1.3 Security Architecture and Principles

Getscreen.me incorporates security controls across its application architecture, communication channels, authentication mechanisms, access management, and infrastructure operations.

Security responsibilities are shared between Getscreen.me and its customers according to the selected deployment model.

1.3.1 Deployment Models

Getscreen.me supports two deployment models: **Cloud (SaaS)** and **Self-Hosted**. Both provide remote access and administration capabilities, but differ in infrastructure ownership, operational management, and security responsibilities.

1.3.1.1 Cloud (SaaS)

In the Cloud deployment model, Getscreen.me operates and maintains the server infrastructure required to provide the service. This includes server administration, infrastructure monitoring, network security, maintenance, and deployment of server-side security updates.

Customers access the service without having to deploy or maintain their own Getscreen.me servers. They remain responsible for managing user accounts, authentication settings, access permissions, and the security of their endpoint devices.

1.3.1.2 Self-Hosted

The Self-Hosted deployment model allows organizations to deploy Getscreen.me server components within their own infrastructure, including private networks and isolated environments.

Customers maintain control over the hosting environment, network configuration, server access, TLS certificates, backups, and infrastructure security. They are also responsible for deploying server updates and security patches made available by Getscreen.me.

Getscreen.me provides the application software and relevant technical documentation, while the customer manages the infrastructure on which it operates.

1.3.2 Shared Responsibility Model

Security responsibilities are distributed between Getscreen.me and the customer depending on the selected deployment model.

In the **Cloud (SaaS)** model, Getscreen.me is responsible for protecting and maintaining the service infrastructure, while customers are responsible for securely configuring and using their accounts, devices, and access permissions.

In the **Self-Hosted** model, customers assume additional responsibility for securing, maintaining, and monitoring their own server infrastructure.

Regardless of the deployment model, customers are responsible for:

- Managing technician accounts, authentication methods, and access permissions.
- Configuring security policies appropriate to their organization.
- Protecting endpoint devices and operating systems.
- Controlling access to remote devices and sensitive information.
- Reviewing relevant security settings and audit logs.

Getscreen.me is responsible for developing, maintaining, and providing security updates for its software components.

The effectiveness of the overall security configuration depends on both the security mechanisms implemented in Getscreen.me and their appropriate deployment, configuration, and use by the customer.

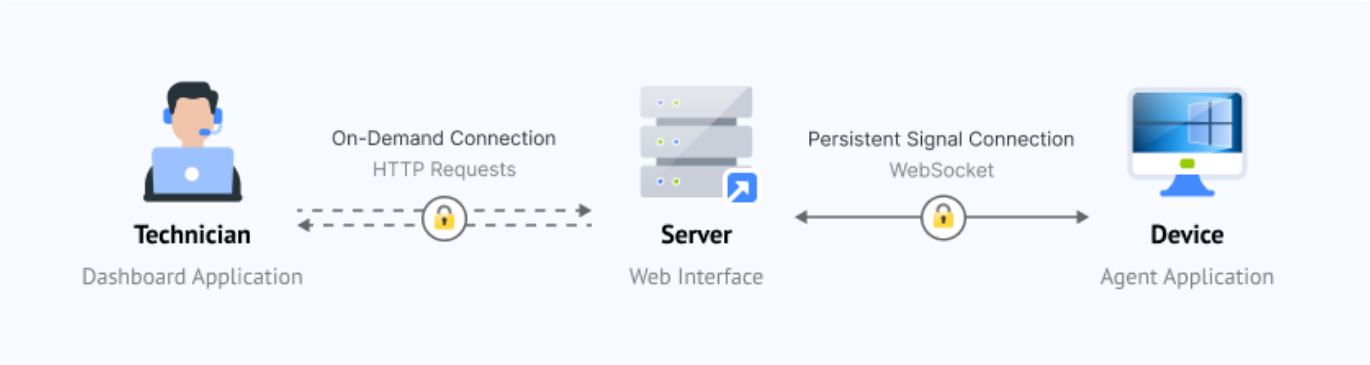
1.4 Connection Security

To establish a secure End-to-End connection for remote device control, the system must first identify the connection participants and coordinate the steps required to establish the connection.

1.4.1 Connection Participants

Three participants are involved in establishing a remote control session:

1. **Device** — The remote device being controlled.
2. **Technician** — The remote user who connects to and controls the device.
3. **Server** — The communication system responsible for coordinating and establishing the connection between the Device and the Technician.



1.4.1.1 Device

The Agent Application must be installed and running on the remote device to enable remote control. Its primary functions are:

- Capturing and transmitting the device's screen and status information to the Technician.
- Receiving and executing commands from the Technician.

While running, the Agent maintains a **persistent signaling channel** with the Server over a **WebSocket** connection secured with TLS encryption

This channel is used to transmit device identification information, connection availability status, and configuration data required to establish remote sessions.

1.4.1.2 Technician

To control a remote device, the Technician must use either the desktop Dashboard Application or the Server's web interface (Web Dashboard).

Both interfaces provide two main functions:

- Requesting a connection to a remote device.
- Accessing the remote control interface and its available tools.

Unlike the Agent, the Technician's application does not maintain a persistent connection to the Server. Instead, it **communicates on demand** through **Short-Lived HTTPS Requests**, with data protected by TLS encryption

Access to the system requires authentication and authorization. Authentication session data is stored in session cookies managed by the browser or Dashboard application.

1.4.1.3 Server

The Server coordinates communication between the Technician and the Device, enabling them to establish an **End-to-End Connection**.

1.4.2 Connection Establishment and Data Transmission

Before the connection can be established, access permissions must be verified on both the Server and Device sides:

- **Server-Side Access Check**. When the Technician requests a connection, the Server checks the Technician's access permissions and assigned role to determine whether the connection is allowed.

  If authorized, the Server forwards the connection request to the Agent.
- **Device-Side Access Check**. Upon receiving the request, the Agent checks the local access permissions configured by the device user and reports its decision to the Server.

If approved, the Server informs the Technician's application of the connection method and data channel selected by the Agent.

During an active connection between the Device and the Technician, data can be transmitted through one of two communication channels: **WebRTC Channel** or **WebSocket Channel**.

The Agent selects the preferred channel based on the Device's network environment, giving priority to WebRTC. If the selected channel is not suitable for the Technician's network environment, the Dashboard application can automatically switch to another channel. The Technician can also manually change the communication channel during an active session.

1.4.2.1 WebRTC Channel

This channel uses WebRTC (Web Real-Time Communication) technology to establish Peer-to-Peer (P2P) connections between endpoints, even when they are located in different networks behind NAT (Network Address Translation) or firewalls.

WebRTC uses ICE (Interactive Connectivity Establishment) to find a suitable network route between the participants. Each participant gathers potential network addresses and ports that can be used to establish a connection. This process is called ICE Candidate Gathering.

The participants then exchange these addresses as ICE Candidates through the Server using their existing signaling channels. They also exchange SDP (Session Description Protocol) messages to negotiate connection parameters. ICE performs connectivity checks and selects a working network route. In this system, the Agent acts as the controlling ICE participant and makes the final route selection.

There are three main types of ICE candidates:

- **Host** — A direct connection using the device's local IP address. Typically used when both participants are on the same local network.
- **Server-Reflexive** — A direct connection over the internet using public IP addresses and ports discovered through STUN (Session Traversal Utilities for NAT).
- **Relay** — A connection through an intermediate TURN server that forwards data between the participants.

Once the network route is established, the participants perform a DTLS Handshake using DTLS (Datagram Transport Layer Security) to establish secure communication. During this process, they:

- Exchange the required cryptographic parameters.
- Verify each other's certificates against the fingerprints provided in the SDP messages.
- Establish a shared secret using a cryptographic key exchange mechanism.
- Derive encryption keys to protect subsequent data transmission.

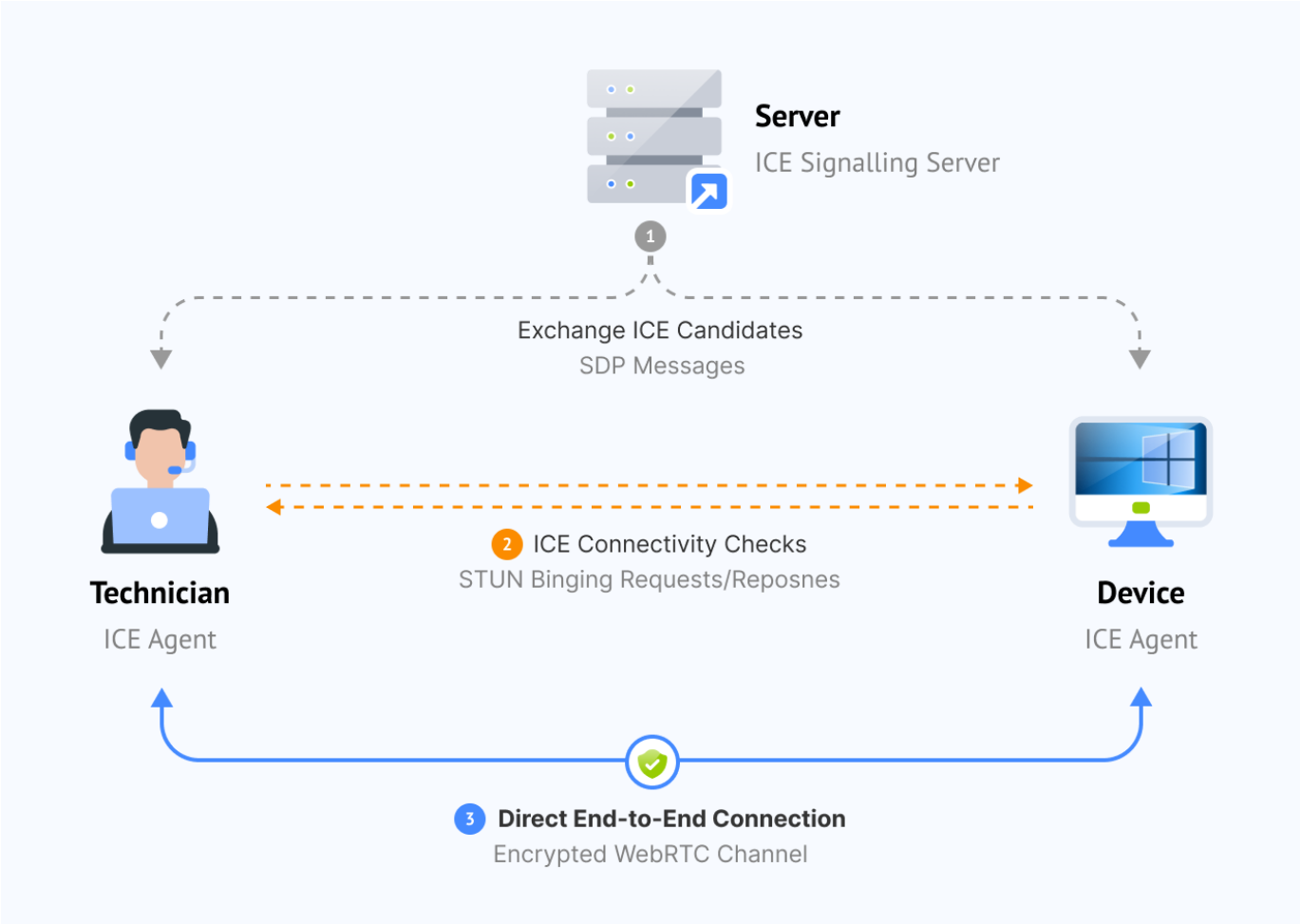
Media streams are encrypted using SRTP (Secure Real-time Transport Protocol), a protocol specifically designed for secure real-time multimedia transmission. It uses RTP sequence numbers and timestamps to handle packet loss and out-of-order delivery without necessarily waiting for retransmission. The encryption keys for SRTP are derived during the DTLS Handshake.

WebRTC primarily uses UDP (User Datagram Protocol) to minimize latency and support real-time data transmission. If a direct UDP connection is unavailable, WebRTC can use TCP (Transmission Control Protocol) or TCP/TLS (Transmission Control Protocol secured with Transport Layer Security), typically through an intermediate TURN server. This allows connections to be established even in networks with strict restrictions.

Regardless of the network protocol or route used, media streams remain protected by SRTP with keys established through DTLS, while WebRTC data channels are protected directly by DTLS.

1.4.2.1.1 HOST

This connection route is used when the participants can reach each other directly through their network interface addresses. This is typically possible when both participants are on the same local network or on different networks with routing between them enabled.

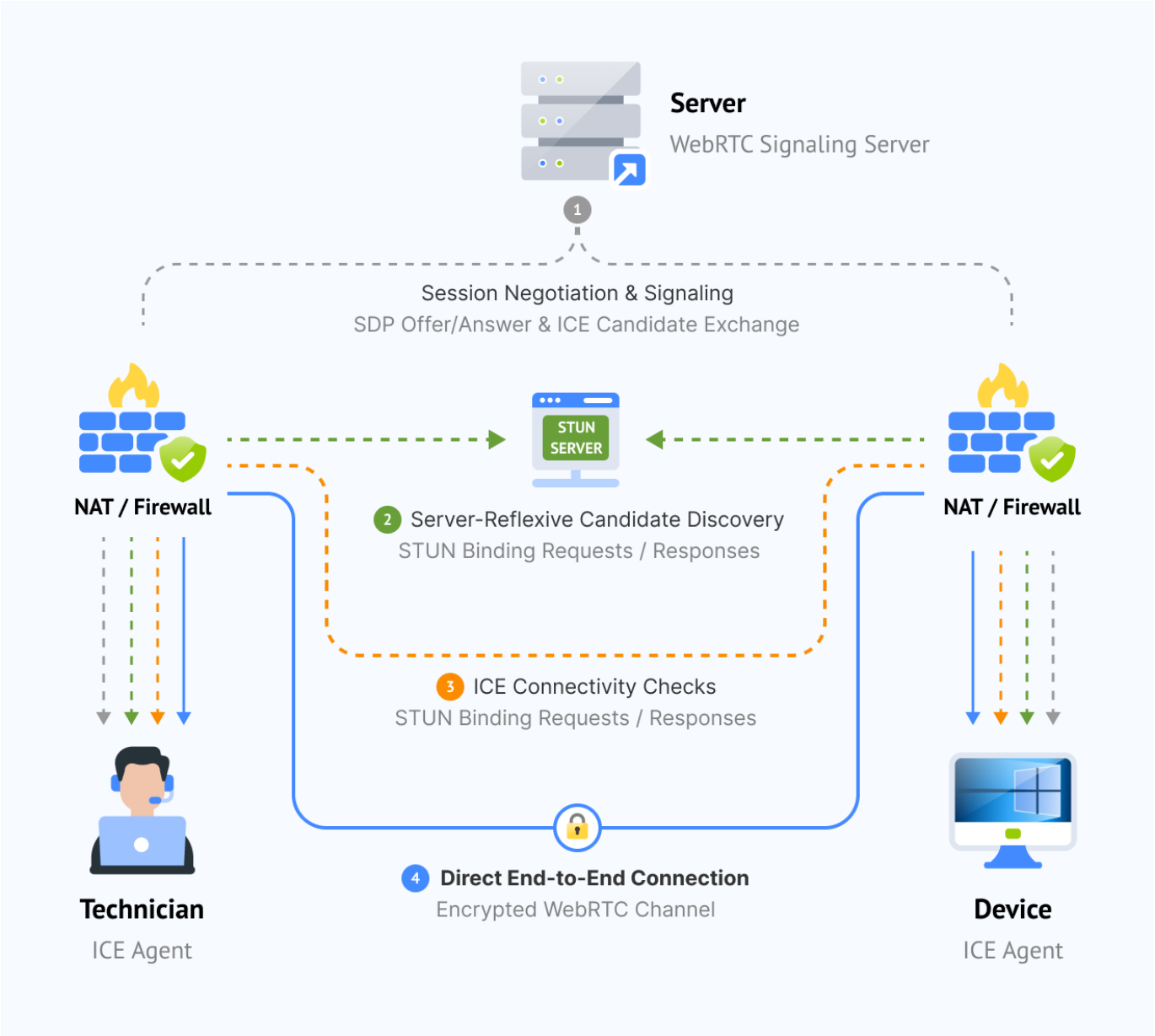


The connection is established through the following steps:

1. Each participant collects its network interface addresses as Host Candidates and exchanges their candidates through the signaling server (gray line in the diagram).
2. The ICE mechanism on each side performs Connectivity Checks between candidate pairs using STUN Binding Requests (orange line).
3. If the checks succeed, ICE selects a working candidate pair, and a direct, **encrypted End-to-End connection** is established between the participants (thick blue line).

1.4.2.1.2 SERVER-REFLEXIVE

This connection route is used when participants can communicate directly through their external IP addresses and ports. This is typically possible when participants are behind different NAT devices or Firewalls that allow direct communication using dynamically assigned external addresses and ports.



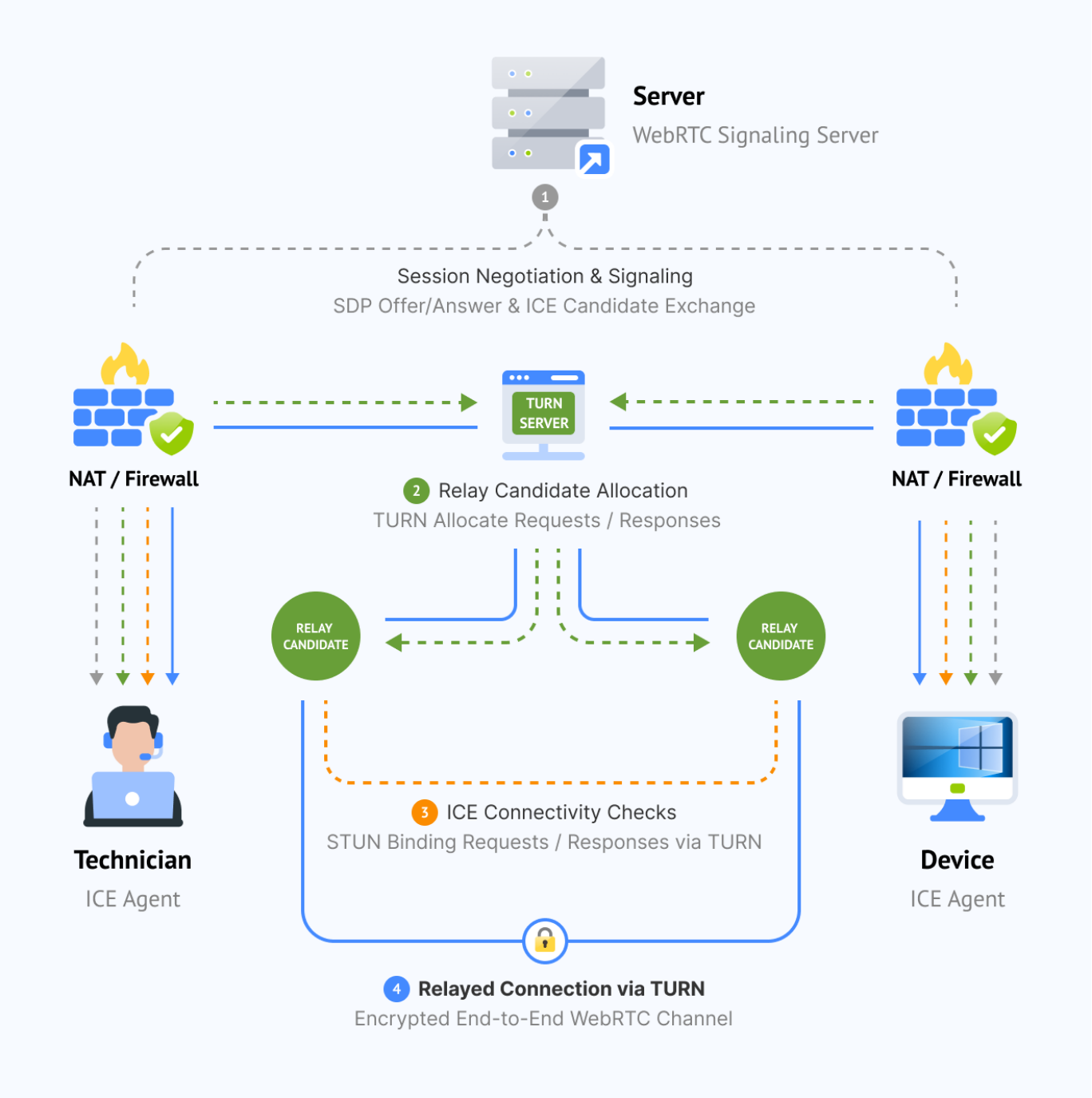
The connection is established through the following steps:

1. The participants exchange their candidates through the signaling server (gray line in the diagram).
2. Each participant contacts an auxiliary STUN server to discover the external IP address and UDP port assigned by its NAT device. These addresses are collected as Server-Reflexive Candidates (green line).
3. The ICE mechanism on each side performs Connectivity Checks between candidate pairs using STUN Binding Requests (orange line).
4. If the checks succeed, ICE selects a working candidate pair, and a direct, **encrypted End-to-End connection** is established between the participants (thick blue line).

1.4.2.1.3 RELAY

This connection route is used when a direct connection through Host or Server-Reflexive Candidates is unavailable. This typically happens when NAT or Firewall restrictions prevent direct communication between the participants.

In this case, data is transmitted through an intermediate TURN server, which relays encrypted traffic between the participants.



The connection is established through the following steps:

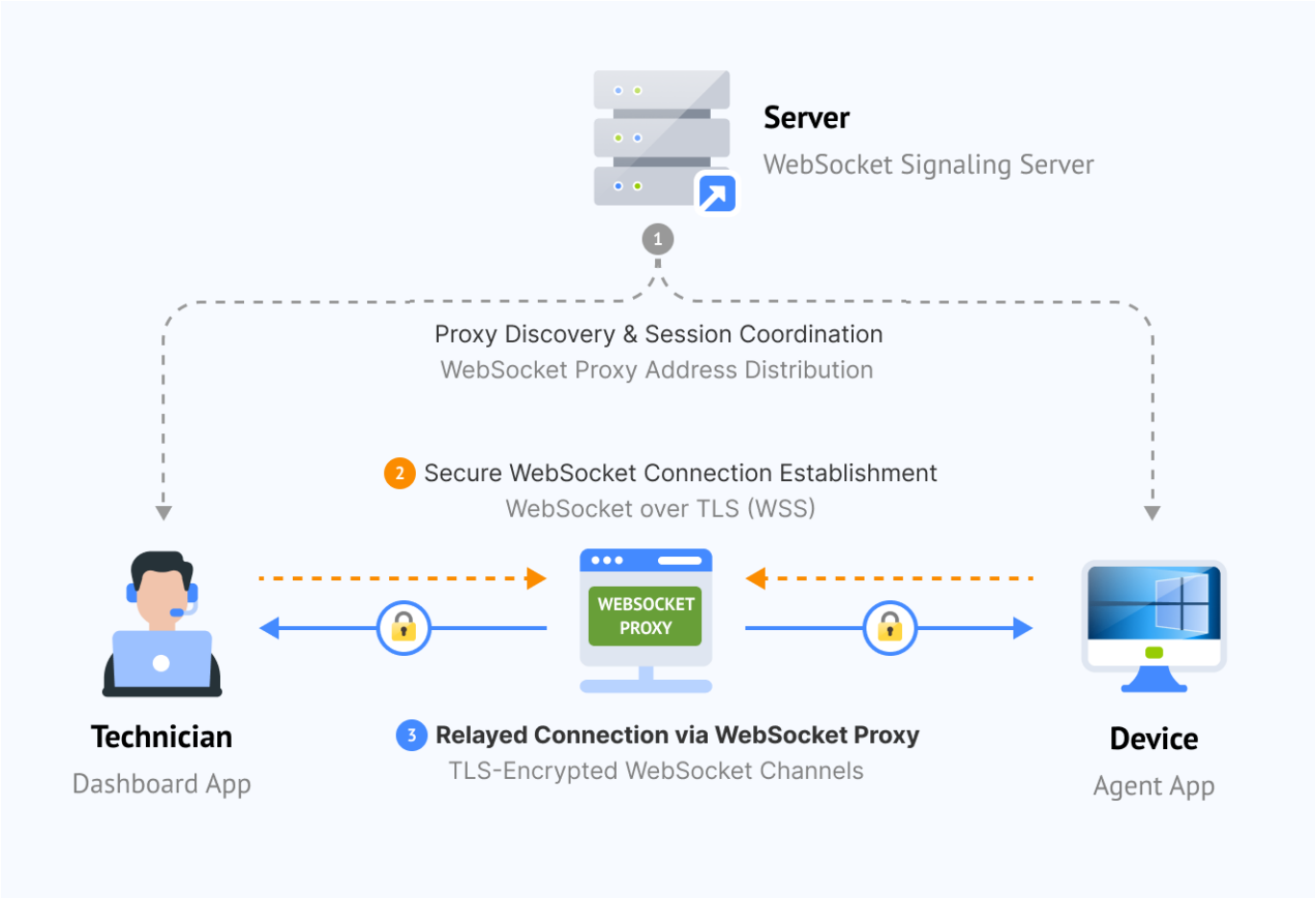
1. The participants exchange their candidates through the signaling server (gray line in the diagram).
2. One or both participants contact an auxiliary TURN server, which allocates an IP address and port for relaying traffic. These addresses are collected as Relay Candidates (green line in the diagram).
3. The ICE mechanism on each side performs Connectivity Checks using STUN Binding Requests through the TURN server (orange line).
4. If the checks succeed, ICE selects a working candidate pair, and an encrypted End-to-End connection is established through the TURN server (thick blue line). The TURN server only forwards the **encrypted WebRTC traffic** and cannot access its encrypted content.

Communication with the TURN server can use UDP, TCP, or TLS over TCP, allowing connections to be established in networks with different Firewall restrictions.

1.4.2.2 WebSocket Channel

This is a fallback communication channel that uses the WebSocket protocol to maintain a persistent connection and exchange streaming media data between the participants.

Each participant establishes a separate **TLS-encrypted WebSocket connection** to a dedicated proxy server, which relays data between them. TLS encryption protects data in transit between each participant and the proxy server.



The connection is established through the following steps:

1. The participants request the proxy server address from the signaling server (gray line in the diagram).
2. Both participants establish secure WebSocket connections to the same proxy server using the address provided by the signaling server (orange line).
3. The participants send and receive data through the proxy server, which relays the traffic between them (blue line).

Communication with the proxy server uses outbound TCP port 443, the same port commonly used for HTTPS. This allows WebSocket connections to work through most NAT devices and Firewalls without requiring additional network configuration.

1.4.3 Data Encryption

All data transmitted between the Device and the Technician is encrypted to protect its confidentiality and integrity during transmission. The encryption mechanisms depend on the communication channel used:

- **WebRTC Channel** — Uses DTLS (Datagram Transport Layer Security) to establish secure communication and negotiate encryption keys. Media streams are encrypted using SRTP (Secure Real-time Transport Protocol), while WebRTC data channels are protected directly by DTLS. Supported encryption algorithms include AES-128 and AES-256, depending on the negotiated cipher suite. Encryption remains active even when traffic is transmitted through an intermediate TURN server.

- **WebSocket Channel** — Uses TLS (Transport Layer Security) to establish encrypted connections between the participants and the proxy server. Depending on the TLS version and negotiated cipher suite, encryption may use algorithms such as AES-128-GCM, AES-256-GCM, or ChaCha20-Poly1305.

The specific encryption algorithm is negotiated automatically during connection establishment, based on the cryptographic capabilities and configuration of the communicating endpoints.

1.4.4 Network Requirements and Port Filtering

Getscreen.me uses defined network ports and protocols to establish secure connections between remote devices, technicians, and servers.

The following ports and protocols are used by Agent and Technician Applications in both SaaS and Self-Hosted deployments:

Direction	Port	Protocol	Purpose
Outbound	443	TCP	Secure communication with Getscreen.me servers and WebSocket proxy connections
Inbound / Outbound	40000–60000	UDP	WebRTC Peer-to-Peer connections between devices and technicians
Outbound	3478	UDP	WebRTC connectivity establishment using STUN/TURN servers

WebRTC Channel

WebRTC primarily uses UDP for low-latency communication. Direct Peer-to-Peer connections may require inbound and outbound UDP traffic, while STUN/TURN servers assist with connectivity establishment and relay traffic when necessary.

WebSocket Channel

WebSocket connections use outbound TCP port 443 with TLS encryption. Both the Agent and Technician Applications establish secure connections to a proxy server, which relays traffic between them.

For SaaS deployments, communication with Getscreen.me servers and WebSocket proxies may require allowing the following domain pattern in the firewall configuration:

`*.getscreen.me`

When WebRTC connectivity is unavailable due to network restrictions, Getscreen.me can use the WebSocket channel as an alternative without requiring additional inbound ports.

For Self-Hosted deployments, server addresses, domain names, and firewall requirements depend on the organization's infrastructure configuration. The exact ports and external services required should be verified against the Self-Hosted deployment requirements.

These network requirements help organizations configure firewall rules, restrict unnecessary network access, and maintain secure remote connections even in restricted network environments.

1.5 Authentication and Account Security

Getscreen.me provides multiple authentication methods and account security features to protect user accounts and prevent unauthorized access. Administrators can configure authentication requirements and access restrictions according to their organization's security policies.

1.5.1 Password Security

Password-based authentication requires technicians to provide valid credentials before accessing the Dashboard. Passwords are protected during transmission using TLS encryption and stored as salted cryptographic hashes.

Administrators can configure password security policies, including:

- **Password Complexity** — Requirements for password length and character composition.
- **Password Expiration** — Configurable password validity periods requiring regular password changes.
- **Password Change on First Login** — Mandatory password change when a technician signs in for the first time.

These features help protect technician credentials, enforce organizational password policies, and reduce the risk of unauthorized account access.

1.5.2 Two-Factor Authentication (2FA)

Two-Factor Authentication adds an additional layer of security by requiring technicians to provide a one-time verification code in addition to their regular login credentials.

Verification codes can be received through one of the following methods:

- **Email** — A one-time verification code sent to the technician's registered email address.
- **Authenticator Application** — A time-based one-time password (TOTP) generated by a compatible authenticator application.
- **Telegram** — A one-time verification code delivered through the Telegram messaging application.

This feature reduces the risk of unauthorized access even if a technician's password is compromised.

1.5.3 Login by Client Certificate (mTLS)

Client Certificate Authentication allows technicians to securely access the Dashboard using digital certificates issued by a trusted Certificate Authority (CA). During the TLS handshake, the server verifies the client's certificate before granting access.

Administrators can configure trusted Certificate Authorities and manage revoked client certificates to control which certificates are accepted.

This feature provides certificate-based authentication without requiring technicians to enter a username and password.

1.5.4 Single Sign-On (SSO/SAML)

Single Sign-On allows technicians to authenticate through their organization's external Identity Provider (IdP) using the SAML 2.0 protocol.

Authentication is handled by the Identity Provider, allowing organizations to apply centralized authentication policies and manage user access through their existing identity management infrastructure.

This feature simplifies account management and enables centralized control over technician authentication.

1.5.5 External Authentication (LDAP)

LDAP integration allows organizations to authenticate technicians using an external directory service, such as Microsoft Active Directory.

Technician credentials are validated against the configured directory, allowing organizations to use their existing corporate accounts for Dashboard access.

This feature helps centralize credential management and reduces the need to maintain separate passwords for technician accounts.

1.5.6 IP Address Whitelisting

Administrators can configure a whitelist of trusted IP addresses and networks from which technicians are allowed to access the Dashboard.

Login attempts originating from IP addresses outside the configured whitelist are automatically rejected.

This feature helps prevent unauthorized access by restricting Dashboard authentication to trusted networks and locations.

1.5.7 Dashboard Session Control

Technicians can view and manage their active Dashboard sessions across different devices, including both the Web Dashboard and Desktop Dashboard Application.

Each session can be individually terminated, allowing technicians to remotely revoke access from devices they no longer use or recognize.

This feature helps technicians maintain control over their authenticated sessions and reduces the risk of unauthorized access from unattended or untrusted devices.

1.5.8 Automatic Logout

Administrators can configure automatic logout policies for technicians in their team. Technician sessions in both the Web Dashboard and Desktop Dashboard Application are automatically terminated after a specified period of inactivity, requiring technicians to authenticate again.

This feature helps administrators enforce session security policies and reduces the risk of unauthorized access through unattended Dashboard sessions.

1.5.9 Inactive Account Removal

Technician accounts that remain inactive for an extended period can be automatically removed according to the organization's configured policies.

This feature helps maintain an up-to-date list of authorized users and reduces security risks associated with unused accounts.

1.6 Access Control and Session Security

Getscreen.me provides flexible access control and session management features to help organizations control who can access remote devices, which actions they can perform, and how remote sessions are monitored and recorded.

1.6.1 Access Rights and Roles

Getscreen.me uses a Role-Based Access Control (RBAC) system to manage technician permissions. Administrators can assign predefined roles or create custom roles with specific access rights.

Roles determine which features and administrative functions are available to each technician, allowing organizations to distribute responsibilities according to their internal security policies.

This feature helps enforce the principle of least privilege and prevents technicians from accessing functions beyond their assigned responsibilities.

1.6.2 Device Access Permissions

Getscreen.me provides several types of device access to support different organizational structures and remote support scenarios:

- **Team Access** — Allows administrators to control which remote devices are accessible to individual technicians or departments within a team. Devices can be organized into groups, with access permissions assigned at the individual device or group level.
- **Public Access** — Allows external technicians to access designated remote devices without joining the team, using protected access links.

- **One-Time Access** — Provides temporary, single-use access links for remote support without granting permanent access to the device.
- **Cross-Team Access** — Enables controlled access to remote devices between separate Getscreen.me teams, supporting collaboration across organizations or departments.

These features help prevent unauthorized connections by ensuring that remote device access is granted only to the intended technicians.

1.6.3 Feature Restrictions

Administrators can restrict the features available to technicians by configuring predefined or custom roles with granular permissions.

Permissions are organized into three categories based on their purpose: **Devices and Groups**, **Invitations**, and **Administration**.

Permission	Category	Description
Screen View	Devices and Groups, Invitations	View the remote screen without control
Screen Control	Devices and Groups, Invitations	Control the remote screen
File Manager	Devices and Groups, Invitations	Access the remote file system
Terminal Mode	Devices and Groups, Invitations	Access the remote terminal
Reboot	Devices and Groups, Invitations	Reboot remote devices
Calls	Devices and Groups, Invitations	Make calls during sessions
Chat	Devices and Groups, Invitations	Exchange text messages
Sync Clipboard	Devices and Groups, Invitations	Synchronize clipboard contents
Screen Preview	Devices and Groups, Invitations	Preview remote device screens
Video Recording Delete	Devices and Groups, Invitations	Delete session recordings
Wake Up	Devices and Groups	Wake devices using Wake-on-LAN
Lock/Unlock	Devices and Groups	Lock the remote OS account
No Confirmation	Devices and Groups	Connect without remote user approval, where applicable
Black Screen and Input Lock	Devices and Groups	Block the remote screen and input
Connection History	Devices and Groups	View connection history
Software Inventory	Devices and Groups	View installed software
Modify Notifications	Devices and Groups	Manage device notifications
Device Modify	Devices and Groups	Modify device properties
Device Delete	Devices and Groups	Delete remote devices
Own Invitations	Invitations	Access personal invitations
All Invitations	Invitations	Access all team invitations
Installation Request	Invitations	Request Agent installation
Audit Log	Administration	Access team audit logs
Technicians and Team Settings	Administration	Manage technicians and team settings
Branding Settings	Administration	Manage branding settings
Billing Settings	Administration	Manage billing settings
API Settings	Administration	Manage API keys
Permanent Access	Administration	Access the Permanent Access section

This feature helps organizations enforce the principle of least privilege by restricting access to remote control tools, device management functions, and administrative settings according to each technician's responsibilities.

1.6.4 Session Recording

Getscreen.me supports optional recording of remote sessions for subsequent review. Session recording can be enabled by the technician and is available for Screen View, Screen Control, File Manager, and Terminal Mode.

Session recordings allow authorized users to review activities performed during remote connections.

This feature improves accountability, supports security investigations, and helps organizations review remote access activities.

1.6.5 Team Audit Logs

Getscreen.me maintains audit logs that record significant actions performed by technicians within a team, providing a detailed history of team activities.

Logged events cover several categories:

Category	Examples of Logged Events
Authentication and Account Security	Login and logout, 2FA verification, password and account changes
Remote Connections	Session start and termination, device wake-up requests
Device Management	Device registration, linking and unlinking, changes to device access
File Operations	File uploads, downloads, deletions, and moves
Access Permissions	Changes to technician permissions and device sharing settings
Team Administration	Technician invitations and removal, department and team settings changes
Temporary Access	Quick Support invitations, one-time access links
Security Operations	Black screen and input lock activation, session recording deletion

Administrators can review audit logs and filter events by date, event type, device, or technician. Access to audit logs is controlled through team permissions.

This feature improves accountability, provides transparency into team activities, and supports security audits and incident investigations.

1.6.6 Access Link Expiration

Public access links and invitations can have configurable expiration periods. Once the specified time limit is reached, the link or invitation becomes invalid and can no longer be used to establish new connections or accept access invitations.

This mechanism helps prevent temporary access permissions from remaining available indefinitely.

1.7 Remote Device Security

Getscreen.me provides security features through its Agent Application to protect remote devices, control remote access permissions, and ensure transparency during remote sessions. These features allow device owners and administrators to enforce security policies and maintain control over how remote devices are accessed and managed.

1.7.1 Code Signing

Getscreen.me Agent Application for Windows and macOS is digitally signed using code-signing certificates to verify the software publisher and protect the integrity of distributed application files.

- **Windows** — The application is signed by **POINT B LTD** using an Extended Validation (EV) Code Signing certificate issued by **GlobalSign**. Digital signatures use SHA-256 and include trusted timestamps to support long-term signature validity.
- **macOS** — The application is signed using an Apple Developer ID certificate, allowing macOS to verify the developer's identity and application integrity.

This feature helps protect users against tampered application files and reduces the risk of installing software from untrusted sources.

1.7.2 Software Update Security

Getscreen.me regularly releases updates for the Agent Application to address security vulnerabilities, fix bugs, and improve software reliability.

The Agent Application supports automatic updates, helping remote devices stay protected with the latest security fixes and improvements.

Updates are distributed through official Getscreen.me infrastructure, with digital code signing providing additional authenticity and integrity protection for Windows and macOS applications.

This feature helps reduce security risks associated with outdated Agent versions and protects remote devices against known software vulnerabilities.

1.7.3 Function Transparency

The standard Getscreen.me Agent Application does not contain hidden monitoring or surveillance features. Its remote access capabilities are transparent and designed for legitimate remote support and administration, rather than covert user monitoring.

The device owner can configure local access permissions through the Agent Application settings, allowing or restricting specific remote access capabilities:

Permission	Description
Mouse and Keyboard Control	Allow remote control of the device's mouse and keyboard
Screen Sharing	Allow remote connections through one-time access links
File Transfer	Allow uploading and downloading files
Terminal Mode	Allow remote terminal access
Audio Calls	Allow voice communication during remote sessions
Lock Input and Black Screen	Allow blocking local input and hiding the device screen
Unconfirmed Connections	Allow remote connections without requiring local user confirmation
Prevent from Sleeping	Prevent the device from entering sleep mode during remote access

These local permissions take precedence over permissions granted to technicians through the Dashboard, unless overridden by centrally enforced Group Policies.

This feature ensures transparency and gives device owners control over how their devices can be accessed and managed remotely.

1.7.4 Group Policy

Getscreen.me provides a built-in Group Policy mechanism that allows administrators to centrally configure and enforce Agent Application settings across groups of devices registered for Permanent Access.

Group Policies enable administrators to define consistent security settings and access permissions for multiple devices. When a Group Policy is applied, its configured settings take precedence over corresponding local Agent Application settings and cannot be overridden by device users.

This mechanism operates independently of Microsoft Windows Group Policy (GPO) and is managed directly through Getscreen.me.

This feature helps organizations enforce consistent endpoint security policies, prevent unauthorized configuration changes, and maintain centralized control over remote access permissions.

1.7.5 Black Screen and Input Blocking

Getscreen.me provides Black Screen Mode and Input Lock Mode to protect remote sessions from unauthorized viewing or local interference.

- **Black Screen Mode** — Hides the remote session activity from the device's physical monitor while allowing the technician to continue working remotely.
- **Input Lock Mode** — Temporarily blocks local keyboard and mouse input while allowing the technician to maintain remote control.

These features help protect sensitive information from unauthorized viewing and prevent local interference during remote administration.

1.7.6 Session Duration Limits

Administrators can configure limits on the maximum duration of remote connections. When the configured time limit is reached, the remote session is automatically terminated.

This mechanism helps prevent remote access sessions from remaining active longer than necessary.

1.7.7 Inactive Session Termination

Remote connections can be automatically terminated after a configurable period of inactivity.

This mechanism helps prevent forgotten or unattended sessions from remaining open and reduces the risk of unauthorized access.

1.7.8 Active Connection Indicator

The Agent Application displays a visual indicator on the remote device whenever an active remote connection is established.

This indicator allows the device user to see when their computer is being accessed remotely, providing transparency and helping to identify potentially unauthorized connections.

1.8 Server Security

Getscreen.me provides server-side security mechanisms to protect its applications against unauthorized access, automated attacks, and common web security threats. In SaaS deployments, Getscreen.me also manages the security of the hosting infrastructure. In Self-Hosted deployments, customers are responsible for securing and configuring the underlying servers and network environment.

1.8.1 CAPTCHA

Getscreen.me uses CAPTCHA verification to protect account registration, authentication, and password recovery processes from automated attacks.

The system supports multiple CAPTCHA providers, including Google reCAPTCHA, Cloudflare Turnstile, and hCaptcha, depending on the server configuration.

This feature helps prevent automated account abuse, credential stuffing, and unauthorized authentication attempts.

1.8.2 HSTS (HTTP Strict Transport Security)

Getscreen.me supports HTTP Strict Transport Security (HSTS), which instructs compatible browsers to use HTTPS for subsequent connections to the server and prevents insecure HTTP access while the HSTS policy is in effect.

This feature helps prevent protocol downgrade attacks and reduces the risk of sensitive information being transmitted over unencrypted connections.

1.8.3 CSP (Content Security Policy)

Getscreen.me implements Content Security Policy (CSP) to restrict which resources can be loaded and executed within the Web Dashboard and remote connection interfaces.

CSP rules define permitted sources for scripts, styles, and other browser resources. These policies can be configured to accommodate trusted external domains when required.

This feature helps mitigate Cross-Site Scripting (XSS), malicious content injection, and other browser-based security threats.

1.8.4 Brute Force Intrusion Protection

Getscreen.me implements protection mechanisms to detect and restrict repeated unauthorized access attempts, including password guessing and brute-force attacks against temporary access links.

The system limits repeated authentication attempts and applies restrictions when suspicious activity is detected.

This feature helps protect technician accounts and remote access links against automated credential guessing and unauthorized access.

1.9 Data Security and Privacy

Getscreen.me implements data protection practices to ensure the secure processing, storage, and management of information generated through its remote access services.

These practices include data minimization, defined retention policies, controlled data deletion, geographic data residency, and encrypted backups to help protect information throughout its lifecycle.

1.9.1 Data Minimization

Getscreen.me follows data minimization principles by limiting the collection and storage of information to what is necessary for providing remote access services, managing accounts, and maintaining system security.

The following table summarizes the main categories of data stored by the system:

Data Category	Description
Account Information	Account identifiers, contact information, and authentication-related data
Technician and Team Information	Team members, assigned roles, departments, and access permissions
Device Information	Device identifiers, operating system details, connection status, and configuration
Connection History	Remote session metadata, including participants, devices, connection times, and session duration
Team Audit Logs	Records of significant actions performed by technicians within a team
Server Logs	Technical events, errors, and operational information required for maintaining the service
Session Recordings	Recorded content of remote sessions, stored only when recording is explicitly enabled
Device Previews and Software Inventory	Desktop previews and installed software information, when the corresponding features are enabled

Remote session content is not recorded or stored by default. Session recordings are created only when explicitly enabled by the technician.

The system also provides configuration options to limit the collection of certain device information, such as desktop previews and software inventory data.

These practices help reduce unnecessary collection and storage of potentially sensitive information while providing transparency and control over the data processed by Getscreen.me.

1.9.2 Data Access Control

For SaaS deployments, access to stored user data is restricted to authorized Getscreen.me personnel responsible for operating and maintaining the service infrastructure.

Access permissions are assigned according to operational responsibilities to limit unnecessary exposure of sensitive information.

For Self-Hosted deployments, customers manage access to databases, storage systems, and server infrastructure under their control.

These restrictions help protect stored information from unauthorized access and disclosure.

1.9.3 Data Storage and Retention

Getscreen.me stores operational data required to provide remote access services, including account information, device configurations, connection history, audit logs, and session recordings.

For Getscreen.me SaaS deployments, data retention periods depend on the type of information:

Data Category	Retention Period
Account Information	For the lifetime of the account
Device and Team Information	For the lifetime of the account
Connection History	For the lifetime of the account
Team Audit Logs	For the lifetime of the account
Session Recordings	Until deleted by the user, the account is closed, or the storage limit is reached
Server Logs	3 years
Backups	Up to 1 year

Connection history and team audit logs remain available throughout the lifetime of the account to support remote access management, historical reviews, and security investigations.

For Self-Hosted deployments, customers manage data storage and retention within their own infrastructure. Actual retention depends on the application's behavior, available configuration options, and the customer's data management practices.

These practices support operational continuity, security investigations, and the availability of historical records for auditing and administrative purposes.

1.9.4 Data Deletion

For SaaS deployments, Getscreen.me provides mechanisms for deleting user accounts and managing stored information.

When an account is deleted, its personal information and primary account record are removed from the active account database. The association between the account ID and the account holder's personal information is removed.

Connection history and team audit logs are retained for operational and security purposes. These records may continue to contain technical metadata, including internal account identifiers, IP addresses, and connection timestamps.

This process removes personal account information while preserving historical records needed for auditing and security investigations.

1.9.5 Data Residency

For SaaS deployments, Getscreen.me operates a geographically distributed server infrastructure to provide remote access services while maintaining control over data storage locations.

The infrastructure includes core service servers, data storage systems, and proxy servers used to relay remote connections.

The following hosting locations and providers are used:

Country	Hosting Provider	Purpose
Germany	Hetzner	User data hosting, core services, and proxy infrastructure
Finland	Hetzner	User data hosting and supporting services
Germany	OVHcloud	Proxy infrastructure
United States	Hetzner, Vultr	Proxy infrastructure
India	DigitalOcean, Vultr, Gcore	Proxy infrastructure
Australia	Vultr	Proxy infrastructure
Israel	Vultr	Proxy infrastructure
Brazil	Vultr	Proxy infrastructure
South Korea	Vultr	Proxy infrastructure

User data hosting is located in Germany and Finland. Proxy infrastructure is distributed across multiple geographic regions to support remote connections and improve network performance.

Getscreen.me also offers a Self-Hosted deployment model, allowing organizations to host the server infrastructure within their own environment and control the geographic location of their data storage.

This architecture provides transparency into data hosting locations and offers deployment options to help organizations meet their internal data residency and regulatory requirements.

1.9.6 Backups and Recovery

For SaaS deployments, Getscreen.me performs daily backups of server data to support service recovery in the event of infrastructure failures, data corruption, or accidental data loss.

Backup copies are encrypted and stored in the same geographic region and with the same hosting provider as the primary server. Backups are retained for up to one year.

For Self-Hosted deployments, backup configuration, scheduling, storage, and recovery procedures are managed by the organization operating the infrastructure.

These measures help reduce the risk of permanent data loss and support service recovery following unexpected incidents.

1.10 Infrastructure Security

Getscreen.me implements infrastructure security measures to protect server environments, restrict unauthorized network access, and maintain reliable operation of remote access services.

The SaaS infrastructure is managed by Getscreen.me, while the Self-Hosted deployment model allows organizations to operate the system within their own infrastructure and apply their internal security policies.

1.10.1 Reverse Proxy

For Self-Hosted deployments, Getscreen.me supports operation behind a Reverse Proxy, allowing organizations to route incoming connections through an intermediate server before forwarding them to the application server.

This configuration allows administrators to integrate Getscreen.me with their existing network infrastructure and control how application services are exposed to external networks.

Reverse Proxy deployment can provide an additional layer of network isolation and reduce direct exposure of application servers to the Internet.

1.10.2 Server Hardening

Getscreen.me applies server hardening measures to its SaaS infrastructure by disabling unnecessary system services and restricting administrative access to authorized personnel.

These measures reduce the number of potential entry points for attacks and limit access to critical server resources.

For Self-Hosted deployments, organizations are responsible for securing the underlying operating system, managing administrative access, and maintaining their server infrastructure.

These practices help reduce the server attack surface and minimize the risk of unauthorized access and exploitation of system vulnerabilities.

1.10.3 Intranet (LAN-Only) Deployment

The Getscreen.me Self-Hosted edition supports deployment within isolated corporate networks without requiring access to the public Internet during normal operation.

Organizations can use internal IP addresses and private domain names to make the system accessible only within their corporate network.

This deployment model allows remote access connections and related server operations to remain within the organization's controlled network environment.

This feature helps isolate remote access infrastructure from external networks and supports organizations with strict network security and data isolation requirements.

1.10.4 Infrastructure Monitoring

Getscreen.me uses automated monitoring systems to continuously track the availability and operational status of its SaaS infrastructure.

Monitoring operates 24/7 and generates alerts when service disruptions or abnormal operating conditions are detected. Both public and internal monitoring systems are used to provide visibility into service availability and infrastructure health.

Public service availability information and status history are accessible through the Getscreen.me Status Page.

Internal monitoring systems provide additional operational information to authorized infrastructure administrators.

These measures help detect infrastructure problems, support timely incident response, and maintain reliable service operation.

1.10.5 Data Center Security

Getscreen.me SaaS infrastructure is hosted by established providers, including Hetzner, OVHcloud, Vultr, DigitalOcean, and Gcore.

Physical security of hosting facilities is managed by the respective data center operators. Documented security measures at facilities operated by these providers may include:

- **Physical Access Control** — Restricted access to data center facilities.
- **Video Surveillance** — Monitoring of entrances and restricted areas.
- **Environmental Protection** — Fire detection and protection systems.
- **Power and Cooling Infrastructure** — Systems designed to support continuous operation of data center equipment.

Hosting providers maintain recognized information security certifications and independent audit reports, including ISO/IEC 27001 and SOC 2 Type II, depending on the provider, service, and data center location.

Official security and compliance documentation is available from Hetzner, OVHcloud, Vultr, DigitalOcean, and Gcore.

Specific security controls and certification scopes vary by hosting provider and data center location.

For Self-Hosted deployments, organizations are responsible for selecting and securing their own hosting environments.

These measures help protect physical infrastructure and provide independent assurance of hosting providers' security practices.

1.11 Secure Development and Vulnerability Management

Getscreen.me incorporates security practices into its software development and maintenance processes to identify potential vulnerabilities, reduce security risks, and maintain the reliability of its remote access services.

These practices include code reviews, automated security checks, vulnerability assessment and remediation, and regular software updates.

1.11.1 Secure Development Practices

Getscreen.me follows a structured development process that includes mandatory code reviews before changes are integrated into the main codebase.

Automated development and security checks are integrated into CI/CD workflows using GitHub Actions and tools such as:

- **Dependabot** — Monitors software dependencies and identifies available security updates.
- **GitHub Advisory Database (npm audit)** — Checks npm dependencies for known security vulnerabilities using GitHub's security advisory database.
- **Trivy** — Scans software components and dependencies for known vulnerabilities.
- **Linters** — Perform static code analysis to identify code quality issues and potentially unsafe coding patterns.

These checks complement manual code reviews and help identify potential issues during the development process.

These practices help detect vulnerabilities and coding errors early, reduce security risks, and maintain consistent software quality.

1.11.2 Vulnerability Management

Getscreen.me identifies potential security vulnerabilities through automated scanning, internal development processes, and reports received from independent security researchers.

Independent security researchers also test Getscreen.me for vulnerabilities and report their findings in exchange for potential rewards. Vulnerabilities can be reported through the regular Getscreen.me technical support channels.

Reported vulnerabilities are reviewed and prioritized based on their severity, exploitability, and potential impact on the service and its users.

Getscreen.me establishes the following response and remediation targets:

Severity	Response Time	Remediation Time
Critical	1 business day	7 calendar days
High	2 business days	30 calendar days
Medium	3 business days	90 calendar days
Low	5 business days	180 calendar days

- **Response Time** — The maximum target time to acknowledge a vulnerability report and begin its initial review, measured from receipt of the report.
- **Remediation Time** — The target time to develop, verify, and release a security fix, measured from confirmation of the vulnerability and assignment of its severity.
- **Business Days** — Monday through Friday, excluding official public holidays in Cyprus.
- **Calendar Days** — Consecutive days, including weekends and public holidays.

Vulnerability severity is assessed based on technical impact, exploitability, and the potential consequences for affected users and systems.

Critical vulnerabilities that are actively exploited receive immediate priority for investigation and risk mitigation, regardless of standard response targets.

Remediation targets may be adjusted when additional investigation or technical complexity requires more time. Where appropriate, temporary mitigation measures may be implemented before a permanent fix becomes available.

This process helps identify, prioritize, and remediate security vulnerabilities within defined timeframes, reducing exposure to known threats.

1.11.3 Security Updates

Getscreen.me regularly releases security updates and software improvements for its Server, Agent, and Dashboard Applications.

- **SaaS** — Server-side security updates are deployed centrally by Getscreen.me, ensuring that the hosted service receives fixes without requiring customer intervention.
- **Self-Hosted** — Server updates are provided by Getscreen.me and installed by customers responsible for maintaining their own deployments.
- **Client Applications** — Security fixes for Agent and Dashboard Applications are distributed through application updates.

Security updates are prioritized according to the severity of the issues they address. Critical fixes may be released independently of the regular software release schedule.

Information about software releases, security fixes, and other improvements is published in the Getscreen.me Changelog.

Regular security updates help address known vulnerabilities and maintain the security and reliability of Getscreen.me components.

1.12 Security Incident Response

For SaaS deployments, Getscreen.me uses infrastructure monitoring, automated alerts, and technical support channels to identify potential security issues and service disruptions.

Reported security issues affecting Getscreen.me software or managed infrastructure are reviewed by the technical team to assess their nature, potential impact, and appropriate corrective actions.

For Self-Hosted deployments, customers are responsible for monitoring and responding to security incidents within their own infrastructure. Getscreen.me can review reported issues relating to its software through its technical support channels.

Depending on the circumstances, corrective actions may include restricting affected functionality, addressing identified vulnerabilities, deploying software fixes, and restoring affected services.

Where an incident involves personal data, Getscreen.me evaluates applicable obligations under data protection laws, including notification requirements under the GDPR.

Customers can report suspected security incidents through the regular Getscreen.me technical support channels.

These measures help identify security issues, support technical investigation, and reduce their potential impact on customers and services.

1.13 Security Assurance and Compliance

Getscreen.me supports organizational security and compliance requirements through technical safeguards, access controls, secure development practices, and the use of established infrastructure providers.

Security responsibilities depend on the deployment model and applicable regulatory and contractual requirements.

1.13.1 Regulatory Compliance

Getscreen.me operates with consideration for applicable data protection and privacy requirements, including the European Union General Data Protection Regulation (GDPR).

Technical and organizational security measures, including encryption, access controls, audit logging, data retention policies, and infrastructure protection, are described in the relevant sections of this document.

For SaaS deployments, Getscreen.me manages the security of its hosted service infrastructure. For Self-Hosted deployments, customers are responsible for securing their hosting environments, maintaining server configurations, and implementing appropriate organizational policies.

Compliance with specific regulatory requirements depends on the customer's use of the service, deployment configuration, and applicable legal obligations.

These measures help organizations implement data protection controls and support their regulatory compliance responsibilities.

1.14 Security Contact and Related Documents

1.14.1 Security Contact

Security vulnerabilities, suspected security incidents, and other security-related concerns can be reported to Getscreen.me through the following contact:

Email: info@getscreen.me

Reports are reviewed by the technical team and handled according to their severity and potential impact.

1.14.2 Related Documents

The following documents provide additional information about Getscreen.me security, privacy, and technical requirements:

Document	Description
Privacy Policy	Personal data collection, processing, and protection practices
Cookie Policy	Information about cookies and similar tracking technologies
Terms of Use	Terms and conditions governing the use of Getscreen.me

Last Updated: October 29, 2026