

## Remote Desktop Software as a forensic resource

Jonathan Manson

To cite this article: Jonathan Manson (2022): Remote Desktop Software as a forensic resource, Journal of Cyber Security Technology, DOI: [10.1080/23742917.2022.2049560](https://doi.org/10.1080/23742917.2022.2049560)

To link to this article: <https://doi.org/10.1080/23742917.2022.2049560>



© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



[View supplementary material](#)



Published online: 14 Mar 2022.



[Submit your article to this journal](#)



[View related articles](#)



[View Crossmark data](#)

# Remote Desktop Software as a forensic resource

Jonathan Manson 

School of Computing, Edinburgh Napier University, Edinburgh, UK

## ABSTRACT

Remote Desktop Software (RDS) enables the controlling of a computer system without the need for physical access. Operations are sent to the remote machine and executed as if performed by a local user. With an unprecedented shift to remote working due to the COVID-19 Pandemic, more people are working on home devices without enterprise IT support and therefore reliant upon this software to collaborate and keep their systems available and secure. RDS complicates a Forensic Investigation as any person with remote access privileges or knowledge of bypassing them could be responsible for an action. Despite its importance and prevalence, forensic research into RDS is minimal. As a market-leading solution for Windows, TeamViewer is an impactful starting point to demonstrate that such software is forensically-valuable to explore. This paper shows that with suitable evidence, an Investigator can identify which machines have performed remote control or been controlled, transferred files and have been remotely rebooted, among other events. We also highlight a potential privacy concern due to inadequate uninstallation processes. To illustrate the value of our findings we publish a Python module for Autopsy that automatically locates, processes and visualises key TeamViewer artefacts for an Investigator.

## ARTICLE HISTORY

Received 20 August 2021

Accepted 1 March 2022

## KEYWORDS


Forensics; TeamViewer;  
remote desktop; RDP;  
windows

## 1. Introduction

### 1.1. Application forensics

The forensic auditing of applications is vital for analysing evidence gathered during a Forensic Investigation. Using this information, an Investigator can discover and interpret captured evidence with a degree of certainty and present well-supported conclusions. Research can be used to develop automated tools, able to operate at-scale and quickly triage large datasets.

**CONTACT** Jonathan Manson  [40454033@live.napier.ac.uk](mailto:40454033@live.napier.ac.uk)  School of Computing, Edinburgh Napier University, 10 Colinton Road, Edinburgh, UK

 Supplemental data for this article can be accessed [here](#)

© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way.

The understanding of forensic artefacts is paramount from a privacy perspective. Users should be conscious of the data applications store, how it is protected and how it can be removed. Even privacy-focused applications like Tor leave noteworthy traces following uninstallation [1].

## **1.2. Remote desktop software**

Remote Desktop Software (RDS) allows users access to the desktop environment of a remote machine, usually via a network connection. Inputs, such as mouse and keyboard events, are sent via this connection and executed on the remote environment. The results are returned and rendered for the controlling user. The controlling of a computer system without the need for physical access has a wide range of applications, including technical support and administering hard-to-reach devices [2]. Commercial RDS often includes features beyond the visualisation of a remote user's desktop, such as integrated voice communication, increased security and auditing, and access management.

With a major shift to remote working due to the COVID-19 Pandemic [3], more people are working on home devices without enterprise IT support and are therefore reliant upon this software to collaborate and access machines previously only operated physically.

RDS requires significant privileges on host systems, for example, access to network traffic, user sessions and filesystems. As a result, such software provides a sizeable attack vector for malicious actors. The recent Colonial Pipelines hack upon US infrastructure has been posited to be the result of the compromise of TeamViewer [4]. RDS is also used as Malware directly and referred to as Remote Access Trojans (RATs) in this context. Scammers steal sensitive data by persuading users into installing a RAT to access their device under false pretences [5].

## **1.3. TeamViewer**

TeamViewer is one of the most popular RDS offerings, providing support for all common Operating Systems and claiming a market share of over a third. It is used by more than 90% of Fortune 500 companies [6] and relied upon within critical sectors, such as healthcare [7]. Some data sources classify TeamViewer as the market leader, with around double the market share as their closest competitors [8].

## **1.4. Aims and scope**

This paper explores the extent to which on-disk artefacts created by RDS can be a resource for Forensic Investigators through their identification, examination and evaluation. This is accomplished through the design and execution of a well-structured experiment upon the TeamViewer Windows RDS and based upon methodology from similar research.

We used the latest free, non-commercial version at the time of experimentation. Even the free version of TeamViewer includes a wide range of individual features and customisation. It would be infeasible to scrutinise all of these, particularly in combination. Therefore, we focused on activities we believe are of most benefit for an Investigator and are likely to be in widespread use – TeamViewer core features.

## 2. Theoretical context

Despite the prevalence of RDS and its privileges on host systems, there is little formal research on artefacts for any product. Formal analysis of Desktop applications, in general, is comparatively rare. For example, Skype has been a mainstay Windows application for many years, but the most recent in-depth forensic research was by Yang et al. [9] in 2016.

The appetite for forensic knowledge of TeamViewer is made clear through open-source Internet articles discussing the topic. Blogs by Lee [10] and Haq [11] contained no descriptions of their methodologies, specific versions tested and lacked many explanatory notes. But they did seem to show that connection details and other important artefacts may be available.

Kerai [12], in an unpublished Thesis, reviewed RDP and VNC-based artefacts and included some investigation into TeamViewer. The specific version of TeamViewer was '5.0.8232', which is so ancient as not to be listed on TeamViewer 2021 support pages [13]. A later conference paper by Kerai and Vekariya [14] found Registry and file artefacts containing personal information, such as email addresses and usernames, on a later (but unnamed) version of TeamViewer. Little detail is provided regarding the methodology used to identify the artefacts and there is no explanation of specific structures.

## 3. TeamViewer technical summary

Due to its status as proprietary software, source code is not available [15]. However, the company maintains an active, open forum for customer queries and an extensive set of guides and manuals which shed some light on the internals.

### 3.1. Features

TeamViewer is free for personal use and provides solutions for all major Operating Systems. Major features include:

- Access and control of Computers remotely, both attended and unattended.
- File sharing.
- Meetings.

- Text chat.
- Session recording.

Commercial or paid licences provide additional features, such as an increase in the number of concurrent meeting participants and remote printing. The TeamViewer [16] Manuals provide much of the following information unless specified.

### **3.2. Identifiers**

TeamViewer devices (Clients) are uniquely identified via an ID, known as a TeamViewer ID or Client ID. IDs are auto-generated for each device on installation based on 'hardware characteristics' and reportedly do not change, even following software reinstallation. Using these IDs, devices running TeamViewer can request control over each other. In LAN scenarios, IDs are not used. IP addresses serve as unique identifiers instead.

TeamViewer IDs are entirely numerical and can have either 9 or 10 digits [17]. When a meeting is started, a unique Meeting ID is generated of the form 'mXX-XXX-XXX'. 'Static' Meeting IDs, which do not change, allow for recurrent or scheduled meetings.

Passwords for authentication for remote control are by default auto-generated on each start of the application, therefore changing relatively regularly. Permanent passwords can be configured to enable unattended access.

### **3.3. Connections**

TeamViewer servers broker connections between devices, which register their assigned IDs to such servers, known as 'TeamViewer Master Servers'. Sessions can be over TCP or UDP, chosen by the Master Server. Initially, both parties connect to the Master for a handshake procedure and most often then connect directly. In circumstances where this is not possible, connections via other TeamViewer servers, known as 'Routers', can be used.

One of the key selling points of TeamViewer is its zero-configuration setup, requiring just installation and no opening of ports on endpoint or gateway devices. However, firewall rules may need to be modified to allow connections from the application from the local machine. The preferred outbound TCP/UDP port is 5938 but, when this is not responsive, TCP ports 443 and 80 are ordered backups.

### 3.4. Security

TeamViewer traffic is secured using both public-key (RSA) and symmetric key (AES) encryption. Devices generate an RSA public/private pair and upload the public key to the Master Server. This key is encrypted in transit with the public key of the Master itself. To connect to each other, devices first request each other's public keys from the Master, encrypted in-transit by their own public key and digitally signed by the Master.

A signed 256-bit AES key is then generated by the initiating device (Controller) and shared via the Master between devices, encrypted with the public key of the other device (Slave). At this point, the Slave authenticates the Controller, and the AES key, before an encrypted session is started using said key. In this reported protocol, private keys are never transmitted, and therefore, not even TeamViewer servers can decrypt the data between devices.

There is no public information available on the specifics of the cryptography involved in this exchange protocol. Notably, we do not know:

- Key formats and public key bit lengths.
- Operating mode of the AES encryption.
- If and when keys are changed or recreated, even during a session.

A significant threat is the brute-forcing of TeamViewer passwords once an ID is known. To prevent such attacks, TeamViewer exponentially increases the latency between connection attempts. This should make attacks, particularly on auto-generated passwords, unfeasible. Older versions of TeamViewer are vulnerable to a bypass of this enforced wait, listed as CVE-2018-16,550 [18].

TeamViewer provides no information on the location of or the storage format of userchosen permanent passwords. Open-source research has previously shown that password artefacts are stored in the Windows Registry encrypted with a constant key and IV, allowing for quick decryption [19]. It is unknown whether the latest versions of TeamViewer have changed this. A further documented vulnerability in TeamViewer versions prior to and including 13.1.1548, CVE-2018-14,333 [20], allows attackers to extract passwords from the process memory of the TeamViewer application.

### 3.5. Custom file types

Sessions are recorded in the proprietary 'TVS' format but can later be converted to AVI video files. Only TeamViewer software can play TVS files. The TVS format is entirely undocumented by TeamViewer, and little work has been done on its specifics.

A blog post by Dennis [21] showed that previous iterations of its file header contained easily extractable metadata, such as versions, dates and various identifiers. The body of the file requires further insider information to process but likely includes the session recording data itself in a custom compressed format.

## 4. Methodology

### 4.1. Design considerations

Since research into our subject area is thin, the analysis of live memory captures was out of scope, and our focus was on the filesystem and Registry. Windows Event Logs were also not of interest. Although TeamViewer may have one or more Providers that deposit Events into the Event Log, we believed it would be erroneous to consider these as purely 'Application Artefacts'. More general Windows Operating System artefacts, such as Prefetch Files, are also not viewed as a TeamViewer-specific resource.

TeamViewer, like all RDS, requires network connectivity. It may also make use of Internet access for automatic updates. We needed to consider the potential impact of connectivity during any experimentation carefully. If an update occurred during testing, we would be comparing the results of different versions of the program. Mahr et al. [22], when researching Zoom, accepted that not all testing could be conducted on the exact same software version due to the forcing of updates that was apparent only during the process.

A final consideration was timing. Applications often run scheduled updates or actions without requiring user input. It is possible that running TeamViewer with varying delays in artefact collection would result in different outputs. For instance, if we collected artefacts too soon following a scenario, there may not have been time for TeamViewer to log it appropriately. Some artefacts may also only be written to disk following termination of the application, the data held in RAM prior to that point.

To simulate timezone differences and detect whether local, UTC or other formats were used in any artefacts, we manually adjusted the clock on the Windows devices during experimentation.

### 4.2. Test protocol

We designed our test protocol to consider the issues described and based it on similar forensic experimentation performed on Windows applications. Like Mahr et al. [22], we began by setting up a test environment, followed by establishing the scenarios to be used, covered by *Setup* and *Scenario Creation* sections, respectively. The *Execution* stage describes how we performed the scenarios and acquired data

following their completion, using VM snapshots similar to Muir et al. [1] but without RAM captures. Our *Analysis* step was influenced by Yang et al. [9] and Majeed et al. [23] and codified how we investigated changes in the filesystem and Registry Hives between scenarios, balancing thoroughness and effort.

#### 4.2.1. Setup

Unlike many other applications audited in the literature, TeamViewer involves several machines. In our methodology, we refer to the machine being administered remotely as the **Slave** and the machine performing the actions as the **Controller**. TeamViewer has bidirectional functionality, so this naming does not feature in other literature. TeamViewer themselves in their literature consider the Controller the Client and Slave the Server, but we believe this terminology unclear regarding when trying to understand which machine is performing actions on the other.

We used Virtual Machines (VMs) throughout our work, based upon the recommendations of Quick and Choo [24]. Using VMs, we were able to quickly save and revert the states of machines in our experiments. We desired an environment as clean as possible to act as a baseline to build from; therefore, we created two 64-bit Windows 10 VMs using VirtualBox 6.1.18 r142142 from a fresh ISO file downloaded from Microsoft. One machine was designated the Slave and the other the Controller, with a single user created for each, named 'Slave' and 'Controller', respectively.

Both machines were connected via a VirtualBox 'Internal Network'. When internal connectivity was required, machines were manually assigned static IP version 4 addresses. An additional network interface was configured for VirtualBox 'NAT Networking' to allow Internet connectivity via our host machine. A shared folder was connected to enable the bi-directional transfer of files between the VMs and the host. The networking setup is illustrated in Figure 1.

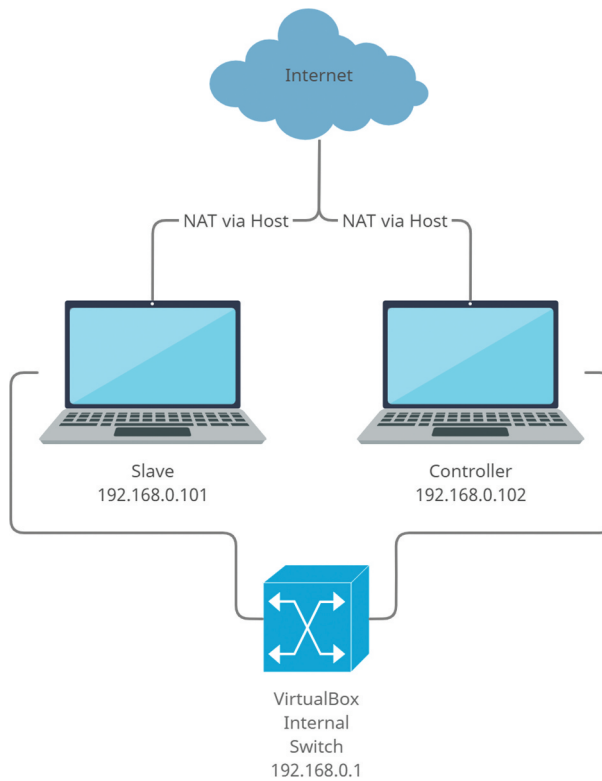
The same download of TeamViewer Full version 15.16.8 was transferred to both machines, as well as a RegistryChangesView (RCV) version 1.27 Windows binary. RCV is a free tool that allows for the snapshotting of the Windows Registry and comparisons between snapshots. TeamViewer and RCV were the latest versions at the time of writing.

The machines were seeded with text files on their user Desktops, called 'created\_locally.txt'. Once each VM had been set up and relevant files transferred, we took an initial ('Clean') snapshot.

#### 4.2.2. Scenario creation

Predefined scenarios are common in the literature [1,9,24–26]. We developed each scenario in line with similar work to imitate typical user activities of interest to an Investigator and numbered them for reference. Scenarios are listed in Table 1. All settings and operations can be reviewed in detail in the TeamViewer manuals [16].





**Figure 1.** Network diagram of experiment setup.

#### 4.2.3. Execution

Analysis of both machines following every listed scenario would have been arduous, and we expect that the extra effort would not have provided significant benefit. Instead, We took VM and Registry snapshots to capture sets of dependent scenarios. Following each node's completion, we left machines to run for between 1 and 5 minutes to allow TeamViewer to complete the updating of any artefacts. Once this time elapsed, we closed any open TeamViewer windows and processes and performed a second set of VM and Registry snapshots.

To provide further assurances that factors such as timing do not affect the scenarios, they were repeated at least three times each, at different dates and times of day in the same manner as that by Yang et al. [9].

#### 4.2.4. Analysis

Our host machine also acted as a Forensic Workstation for further analysis, which was set up with tools required to analyse collected artefacts. FTK Imager version 4.5.0 was used to extract filesystems from snapshots by adding

Virtual Hard Disk (VHD) files created by VirtualBox as evidence items. Snapshots were imported into Autopsy using their VHD files by cloning the machine's state into a new VM in VirtualBox and extracting VHD files.

We used the comparison capabilities of RCV to determine changes in the Registry between scenarios. WinMerge version 2.16.10.0 was used to compare changes in captured filesystems quickly. WinMerge is a free, open-source tool for comparing modifications to files and directories, which provides a straightforward interface for examining differences.

We needed to isolate the TeamViewer-specific actions for analysis, which could become complicated when looking at files that multiple applications modify. When reviewing the differences between snapshots, we did not study every artefact. Instead, we used keywords to focus on those TeamViewer-related (i.e. not likely to be altered by any other software), similar to Teing et al. [27]. Without this limitation, the process would have become much too time-consuming. Matches were case-insensitive and could appear at any point in the file or Registry path.

The keywords we used were as follows:

- TeamViewer
- TV
- Viewer
- Meeting
- Control
- Slave
- Session

We used the Microsoft SysInternals Strings program to extract information from artefacts that could not be easily rendered into Unicode or other formats.

#### 4.2.5. *Software and tools*

Table 2. Summary of software used during experiment.

### 5. Results

Following installation, the application starts immediately and presents a TeamViewerID and password. The displayed passwords are auto-generated upon each start of the application. Table 3 shows the TeamViewer IDs related to each machine.

Rather than provide an exhaustive list of all artefacts identified, we discuss only those we believe are of significant forensic value in terms of identifying:

- TeamViewer installation (i.e. the presence of TeamViewer).
- TeamViewer usage (i.e. what actions have been performed and when).
- Information on user identities and preferences.

**Table 1.** Summary of scenarios.

Scenario Number	Scenario Summary
S0	Clean Windows 10 machine.
S1	TeamViewer installed.
S2	Remote Control connection via WAN.
S3	Application usage.
S4	File creation and access.
S5	File transfer using drag and drop.
S6	Remote restart.
S7	File transfer using file transfer mode.
S8	TeamViewer Meeting.
S9	Settings configuration.
S10	Session recording.
S11	LAN connection.
S12	TeamViewer uninstalled.

**Table 2.** Shows a summary list of the software and tools used during the experimentation stage.

Role(s)	Product	Version
TeamViewer Installation File	TeamViewer Full	15.16.8
Forensic Workstation/Host machine	Windows 10 x64	N/A
VM software.	VirtualBox for Windows	6.1.18 r142142
VM OS installation.	Windows 10 × 64 ISO	Consumer Edition (2004), updated in February 2020.
Forensic analysis.	Autopsy for Windows	4.17
Data Acquisition.	FTK Imager	4.5.0
Registry Snapshots & analysis.	RegistryChangesView	1.27
Filesystem comparison.	WinMerge	2.16.10.0
String extraction.	SysInternals Strings	2.53

**Table 3.** TeamViewer IDs of machines within experiment.

Machine	TeamViewer ID
Slave	958,223,731
Controller	958,517,082

Where reasonable, we provide sample structures and examples of artefacts we have found, such as lines in log files and headers. These are often truncated (using), templated or otherwise modified to illustrate a point.

### 5.1. Firewall rules

Upon installation four new firewall rules were added that allowed inbound TCP and UDP from any remote address on Public profiles for the following programs:

- C:\Program Files (x86)\TeamViewer\TeamViewer.exe
- C:\Program Files (x86)\TeamViewer\TeamViewer Service.exe

## 5.2. Registry artefacts

The TeamViewer Registry key locations we found and their descriptions are shown in [Table 4](#).

### 5.2.1. HKCR artefacts

Subkeys were created under the Hive HKEY\_CLASSES\_ROOT, which contains file extension association information. These included .tvs, .tvc and .tvi, among others.

These artefacts provide a good indicator that TeamViewer is currently installed some-

where on the machine. They were removed when TeamViewer was uninstalled.

### 5.2.2. HKCU artefacts

Artefacts in the current user Hive location, HKEY\_CURRENT\_USER (HKCU), can reveal details regarding user preferences and historical usage. [Table 5](#) provides a summary of those we expect to be of the most value.

### 5.2.3. HKLM artefacts

HKEY\_LOCAL\_MACHINE (HKLM) values apply to the complete installation of TeamViewer and do not vary between different Windows accounts. [Table 6](#) provides a list of interesting keys, descriptions of them and investigative comments.

## 5.3. Filesystem artefacts

[Table 7](#) shows a summary of key filesystem locations and artefacts. In order to shorten path descriptions the following aliases are used:

**Table 4.** Identified teamviewer registry locations.

Location	Description
HKEY_CLASSES_ROOT (HKCR)	File extension information for TVspecific formats.
HKCU\SOFTWARE\TeamViewer	User preferences and configuration values.
HKLM\Software\WOW6432Node \TeamViewer	Machine data, such as certificates, password information and software details.

**Table 5.** Identified HKCU registry artefacts.

Key Name	Description
FT_Start_Directories	Recent local and remote directories used by the File Transfer utility, of the format: <Slave ID>?<Local Dir> <Remote Dir> This was only present on the Controller.
Meeting_Username	The configured username for Meetings. Automatically set to the Windows username.
Username	The configured Display Name set by the user. If this key does not exist then the Windows Computer Name is used in interactions and logs.

**Table 6.** Identified HKLM registry artefacts.

Key Name	Description
Always_Online	Indicates whether TeamViewer is always running and will start on Windows startup to allow for 24/7 access. 0 indicates false, 1 true.
BlacklistBuddy	Email addresses of accounts added to the block list.
Certificate	DigitalCertificatedetails.Unknownformat. Changes regularly.
CertificateKey	Key for certificate. Unknown format. Changes regularly.
ClientID	The TeamViewer ID in hexadecimal format.
ConnectionHistory	Each eight bytes is an entry of a connection performed. Even if the same machine is connected to, the entry differs each time. Unknown format. Only exists on the Controller.
InstallationDirectory	Location of the TeamViewer program files.
LanOnly	If existing and set to 1, only LAN connections are allowed.
LastLogoffTime	Unix epoch representation of the time the most recent session ended. Only exists on the Slave.
LastMACUsed	Lists MAC addresses of the most recent interfaces used. Exists even when no connections have been made. Includes significant whitespace at the start of the value.
PermanentPassword	Binary data of the Personal Password set. If not existing, no such password is allowed. Format unknown. If the same password is saved at different times, a different value is stored.
PermanentPasswordDate	UTC Timestamp of when the password was set in ISO 8601 combined date-time format. E.g. '20210409T122529'.
Restore_Session_Notify_ID	The TeamViewer ID to notify when a session needs to be restored. We found this to contain the Controller ID following a remote reboot. Only exists on the Slave.
Security_WinLogin	Determines if Windows logon is allowed. A value of 2 indicates 'Allowed for all users'. If the key does not exist or set to 0, it is not allowed.
Version	Version of TeamViewer installed.

**Table 7.** Identified TeamViewer filesystem artefacts.

Location	Description
C:\Program Files (x86)\TeamViewer	Installation directory, alias <ID>.
<ID>\Connections_incoming.txt	Incoming Connection log.
<ADR>\Connections.txt	Outgoing Connection Log.
<ID>\TeamViewer15_Logfile.log	Program Log.
<ADR>\MRU\RemoteSupport	Configuration file directory.
<ADL>\RemotePrinting\tvprint.db	Printing and chat databases in SQLite format.
<ADL>\Database\tvchatfile.db	

- <ADL> for c:\users\<username>\AppData\Local\TeamViewer
- <ADR> for c:\users\<username>\AppData\Roaming\TeamViewer

We discuss these artefacts in more detail in the following sections.

### 5.3.1. Connection logs

TeamViewer records details of incoming and outgoing connections in text files, with one line per connection. Lines are of similar formats in each, with white-space separating data points. Listing 1 describes incoming and outgoing log line formats.

Outgoing:

<Slave ID> <Start Date> <Start Time> <End Date> <End Time> <Current

Windows User> <Connection Type> <Unique Session ID>

Incoming:

<Controller ID> <Controller Display Name> <Start Date> <Start Time> <End Date>

<End Time> <Current Windows User> <Connection Type> <Unique

Session ID>

Listing 1: Connection Log formats

Listings 2 and 3 show examples from the Controller and Slave logs respectively.

---

```
958223731 05-04-2021 16:30:36 05-04-2021 16:32:37 Controller RemoteControl {
CCFBCE3E-49B1-420D-9589-BFBAD8EF70E6}
```

---

Listing 2: Example Connection Log line (Controller).

---

```
958517082 Controller 05-04-2021 16:30:40 05-04-2021 16:32:56 Slave
RemoteControl {CCFBCE3E-49B1-420D-9589-BFBAD8EF70E6}
```

---

Listing 3: Example Connection Log line (Slave).

We further discovered that:

- All times are in UTC.
- The Session ID is the same in both logs.
- The Connection Type for File Transfer Mode is 'FileTransfer'.
- If Connections are made via LAN, the TeamViewer IDs are '0'.
- By default, the display name is the Windows Computer Name (hostname).

### 5.3.2. Program log

The TeamViewer15\_Logfile.log file, which we refer to as the 'Program Log', is an incredibly verbose record of application activities written to continually whilst running. The challenge with a log this detailed was to extract only relevant information.

Upon startup, a summary of program, server and computer details are written to the log. An example is shown in Listing 4.

---

```

Start:2021/04/05 17:21:46.764 (UTC+1:00)
Version:15.16.8 (32-bit)
Version short hash: b0756eefb01
ID:958,223,731
Loglevel:Info (100)
License:10,000
Server:master4.teamviewer.com
IC:-116,641,248
CPU:Intel64 Family 6 Model 142 Stepping 11, GenuineIntel
CPU extensions: g9
OS:Win_10.0.19041_W (64-bit) IP:169.254.111.195,10.0.2.15
MIDv:2
Proxy-Settings:Type = 1 IP = User =
IE:11.789.19041.0
AppPath:C:\Program Files (x86)\TeamViewer\TeamViewer_Service.exe
UserAccount:SYSTEM

```

---

Listing 4: Example Program Log startup.

The IP field contains a comma-separated set of addresses of interfaces on the machine. In Listing 4 the first (169.\*) is the address of the default VirtualBox internal network and the second the address of the NAT interface used to access the Internet.

The Program Logs of our experimental machines as a whole contained some similarities, but were sufficiently different such that we analysed them separately.

All log lines (unless specified) are of the format detailed in Listing 5, where the Process ID is that of the TeamViewer process and Local Time is in 24-hour format. For brevity in the following examples and extracts, we omit all but the <Details> sections.

```
<YYYY/MM/DD> <Local Time> <Process ID> <Unknown> <Unknown>
<Details>
```

Listing 5: Program Log line format.

The following subsections highlight information we were able to consistently extract from the Program Logs of either machine, focusing specifically on the events covered as part of the experimental scenarios.

## Connection details

When a connection is received on a machine, it is noted by logs that show an incoming session, encryption negotiation and shows that UDP punching is employed.

CommandHandlerRouting [2]::CreatePassiveSession(): incoming session via GB-LON

-ANX-R017.teamviewer.com, protocol Tcp

```
CTcpConnectionBase [2]::ConnectEndpoint(): Connecting to endpoint
188.172.198.148:5938
```

```
Negotiating session encryption: client hello received from 958517082, RSA key length
= 4096
```

```
[...]
```

```
UDPv4: punch received a = 169.254.146.29:57,662: (*)
```

Listing 6: Program Log connection example (Slave).

UDP punching enables packets to be received from a remote endpoint, even if the local computer is behind a device implementing Network Address Translation (NAT) or a firewall [28]. In normal situations access to a device using their public IP is not possible without specifically configured router rules. From a forensics standpoint, what is most interesting about this is that we can identify the public IP of the Controller from the Slave and vice-versa. In Listing 6 this is 169.254.146.29. Due to the way our machines were networked, this is a private IP within the VirtualBox internal network.

We repeated this portion of the experiment using an Amazon EC2 instance hosted on the Internet to connect to one of our VMs. By searching for "punch received a =" we found the verifiable public IP of the EC2 machine within the logs.

However, we found the best and most reliable logs to identify session details contain the lines 'New Participant added in CParticipantManager' or 'was added with the role'.

Line details can be of either of the following forms in Listing 7, where <> indicates a parameter such as the TeamViewer ID or display name. Listing 8 shows an example log when the Controller connected to the Slave.

```
New Participant added in CParticipantManager (ID [<,<TVID>,<Unknown>])
```

Listing 7: Program Log incoming connection format.

---

```
CParticipantManagerBase participant Slave (ID [958,223,731,-775,734,905]) was added with the role 3
New Participant added in CParticipantManager Slave ([958,223,731,-775,734,905])
CParticipantManagerBase participant Controller (ID [958,517,082,546,167,381]) was added with the role 6
New Participant added in CParticipantManager Controller
([958,517,082,546,167,381])
```

---

Listing 8: Program Log connection extract (Slave).

We believe that the role numbers 3 and 6 relate to Slave and Controller respectively, but cannot confirm this without further effort.

There was no consistent log that defined when a connection was terminated, though lines with the pattern 'in session [0-9] has terminated' were found.



On the Controller, similar logs for connections were found, which can be identified through searching for 'was added with the role'. Notably, the Controller appears to be added to the session first from its own perspective.

```
CParticipantManagerBase participant Controller (ID [958517082,546167381]) was
added with the role 6
```

```
CParticipantManagerBase participant Slave (ID [958223731,-775734905]) was added
with the role 3
```

Listing 9: Program Log connection extract (Controller).

The Program Log is not a complete historical record and is eventually archived once it becomes greater in size than 1MB [29]. At any point in time there is only one active log file and one archived file. i.e. when a new file is created, the currently archived file is replaced. We found that this size is comparatively generous for an Investigator to understand substantial amounts of previous activity. The previous archived log file has \_OLD appended to the end of its filename. We did not produce enough data to determine whether several archives are kept.

## Remote reboot

Within the Slave's log, we found evidence that the remote reboot ordered by the Controller in scenario S6 had occurred. This also included the ID of the machine ordering the reboot (as 'PartnerID'), shown in Listing 10. No evidence of requesting a remote reboot could be seen on the Controller.

---

```
CRemoteReboot::Reboot (Reboot-Type = 1, PartnerID = 958,517,082,
InstantSupportSession = 0)
```

---

Listing 10: Program Log remote reboot example.

## Meetings

We found meeting IDs, participant IDs and names in the Slave's Program Log. Meetings were also designated a String in the format of a Globally Unique ID (GUID) (Listing 11). On the Controller, similar to the Slave (the meeting host), we found IDs and display names, looking for 'joined meetings' and 'added name' (Listing 12). Additionally, when a meeting session ended, we could sometimes identify the reason by searching for 'TerminateSession'.

---

```
Start meeting with MeetingID = m600-002-52
VolP: Meeting session created: MeetingID = m600-002-52, ParticipantID = [958,223,731,1,301,508,607],
MeetingGUID = {3BE8BF24-BEFC-4F0C-964C-
CB55B8FE6B9C} tvclientbase::blitz::ManagerImpl::AddParticipant(05E30C08): participant id
[958,517,082,915,592,460] added name 'Controller'
```

---

Listing 11: Program Log meeting example (Slave).

---

```
ConnectionGuard: joined meetings in sessions: 1(m600-002-52)
tvclientbase::blitz::ManagerImpl::AddParticipant(055A8D48): participant id
[958,223,731,1,301,508,607] added name 'Slave' tvclientbase::blitz::ManagerImpl::AddParticipant
(055A8D48): participant id [958,517,082,915,592,460] added name 'Controller'
[... ]
SessionControl::TerminateSession: Session termination reason
MeetingPresenterDisconnect
```

---

Listing 12: Program Log meeting example (Controller).

## Authentication attempts

Vital for intrusion investigations is the identification of any reconnaissance or failed access attempts. We found the details highlighted in Listing 13 when subsequent incorrect passwords were submitted to the Slave. Using search terms of 'attempt number' and 'password was denied' we could quickly locate relevant logs lines. Failed authentication attempt details did not appear in the Controller log.

```
AuthenticationBlocker::Allocate: allocate ok for DyngateID 958517082, attempt num-
ber 1
```

```
AuthenticationPasswordLogin_Passive::RunAuthenticationMethod: authentication
using dynamic password was denied
```

```
AuthenticationBlocker::Allocate: allocate ok for DyngateID 958517082, attempt num-
ber 2
```

Listing 13: Program Log failed authentication example.

## File transfers

File transfer details were available in the Program Log on the Controller. We could identify local and remote file locations, as well as the sizes of those moved.

Listing 14 shows summarised results of the Controller downloading 'created locally.txt' from the Slave and then uploading an empty file called "created remotely.txt".

---

```
Write file C:\Users\Controller\Desktop\created_locally.txt
Download from "C:/Users/Slave/Desktop/created_locally.txt" to "C:/Users/
Controller/Desktop/created_locally.txt" (13 Bytes)
[...]
Upload from 'C:\Users\Controller\Desktop\created_remotely.txt' to "C:/Users/
Slave/Desktop/created_remotely.txt" (0 Bytes)
```

---

Listing 14: Program Log file transfer examples.

A finding of note regarding the File Transfer scenarios is that there was no discernible difference in the logs between the use of the "Drag and Drop" and the "File Transfer Mode" tools. It is, therefore, likely these capabilities are backed by the same program code, and only the user interface shown to the operator differs. From a forensics standpoint, this may make investigation simpler, not having to cover two styles of transfer.

## Session recording

On the Controller we could identify when a session recording was completed via a search for 'StoreSessionFile: File', which included details on the filenames being written. Listing 15 shows an extract of such a log following scenario S10.

---

```
StoreSessionFile: File 'C:\Users\CONTRO~1\AppData\Local\Temp\TeamViewer
TeamViewerSession 2021-04-06 13.14.17.tmp.tvs' moved successfully to 'C:\Users\Controller\Desktop\Slave
(958,223,731)_2021-04-06 13.14.tvs'.
```

---

Listing 15: Program Log session recording example.

The log also shows that session recordings are initially stored in a temporary directory. Session files are further discussed in a later section.

No relevant logs could be found on the Slave (the device being recorded) that detailed that the session was being recorded by the Controller.

### 5.3.3. TeamViewer configuration files

TeamViewer configuration files are in Windows INI file format but with a tvc file extension. We found these on the Controller following connections and meetings, under <ADR>\MRU\RemoteSupport. Remote Control configuration files were found named with the Slave TeamViewer ID and Meeting configuration files with the Meeting ID.

The first line of configuration files contains a simple header. The body of the files contained two ' = '-separated fields of 'targetID' and 'action', of either a TeamViewer or Meeting ID and type of connection respectively. Examples are shown in Listing 16, retrieved from the Controller.

```
File: 958,223,731.tvc
===== [TeamViewer Configuration]
targetID = 958,223,731 action = RemoteSupport
File: m600-002-52.tvc
===== [TeamViewer Configuration]
targetID = m600-002-52 action = Meeting
```

Listing 16: TeamViewer Configuration File content examples.

#### 5.3.4. Database files

We found SQLite database files in the following locations:

- <ADL>\RemotePrinting\tvprint.db
- <ADL>\Database\tvchatfile.db

Their names imply they store chat and printing data but were both empty in all scenarios.

#### 5.3.5. Session recording files

Recorded sessions are by default saved with filenames of the following format when taken by the Controller:

<Slave Display Name> (<Slave ID>)\_<YYYY-MM-DD> <HH>.<SS>.tvs

Examining the file, we found a human-readable header indicating useful metadata, such as versions, times, TeamViewer IDs of both participants and a display name. The first line includes 'TVS', which is likely used for file format validation purposes. The GUID reported in this header was not the same as any aforementioned Session GUID we could find. Similarly, the 'LocalParticipantID' number also did not appear elsewhere, and therefore we were unable to determine their meaning.

```
TVS
Version 5
TVVersion15.16.8
Date2021-04-06 13.14
ClientID958517082
ServerIDSlave (958,223,731)
LocalParticipantID4116799522416294559
GUID{50FB8ABF-4E7F-485D-AF56-6AF08A2572E7}
StreamTypes2
```

ScreenFeatures 127

MetadataPosition000000000000913c

Listing 17: TeamViewer Session File content example.

### **5.3.6. Temporary files**

c:\users\<username>\AppData\Local\Temp was found to contain a variety of TeamViewer artefacts, including:

Bitmap images of profile photos of accounts connected to. The naming format appeared to be <Display Name>.bmp.

A TeamViewer folder with a file named TV15Install.log containing installation logs.

### **5.3.7. Remaining artefacts**

Following TeamViewer uninstallation, some artefacts were found to persist. These were as follows:

The incoming connection log at <ID>\Connections\_incoming.txt.

All files under <ADL> and <ADR> and the temporary files.

The registry key HKLM\SOFTWARE\WOW6432Node\TeamViewer, though without any values.

All other artefacts mentioned could no longer be found.

## **5.4. Autopsy plugin**

To validate and demonstrate the value of our findings, we created a data ingest plugin for Autopsy and tested it against all of our VM snapshots. Our testing showed that the plugin could reliably extract all relevant information.

## **5.5. Summary of results**

Using our forensic methodology and analysis, we identified filesystem and Registry artefacts that can provide evidence and details of TeamViewer:

- Installation.
- User settings.
- Incoming and outgoing connections.
- Public IP addresses of connected machines.
- Failed authentication attempts.
- File transfers.
- Meetings.
- Recorded Sessions.
- Remote rebooting.

In the context of the defined scenarios of interest, only S2 and S3 did not leave any evidence. This is because the specific actions (application and file usage) took place via TeamViewer rather than by direct use of it.

## 6. Discussion

### 6.1. Limitations

Although we were able to identify a large number of TeamViewer activities, there are some notable absences. We could find no evidence of:

- Chat messages sent between parties.
- Application usage by the Controller, such as in scenario S2.

This means in practice that once a session has been established, TeamViewer artefacts themselves cannot provide evidence of non-TeamViewer operations. An Investigator may determine that a TeamViewer session took place during a period of time but cannot prove whether the remote or local user performed, for example, web activity and file creation (but not transfer). With further testing, we may be able to determine when the database files are populated and with what data. 'Premium' features, such as Remote Printing, are only available to organisational or paying users and therefore could not be tested.

Many of the artefacts discovered appear in temporary directories and, therefore, may not be persistent over an extended period or between reboots. Other artefacts depend on human-readable log files that have a maximum size limit before they are rotated, and any previous data is lost [29].

One of our disappointments is that we could not understand the formats of the cryptographic-related Registry keys we found, such as PermanentPassword and ClientKey. We do have some informal ideas of when these values are created and changed through the black-box style experimentation we conducted. However, we cannot, for example, extract the cleartext value of a password stored with TeamViewer. Some reverse engineering of the application itself is likely to be required to do this.

We expect that most artefacts will remain constant for any short-term updates of TeamViewer, but newer versions will likely progressively modify, remove and add artefacts. Of particular concern are the artefacts found through parsing of the Program Log.

Our testing occurred on 64-bit versions of Windows. There will be slight variations in artefact locations on a 32-bit build. For example, Registry locations including WOW6432Node will be underneath the parent Software key instead, and the installation directory will be "Program Files", rather than "Program Files (x86).

We found no IPv6 addresses during our experimentation, though we expect that they would also be logged when used. Matching and extraction of these will be more involved than IPv4 addresses, as they can have many more formats and can be contracted in several different ways.

## **6.2. Comparison to Literature**

Since formal forensic work concerning any RDS is limited, comparison and evaluation with respect to the existing literature is difficult. Our findings show artefacts of similar forensic value, if not more so than some of those found when analysing other applications. Zoom filesystem artefacts found by Mahr et al. [22] are comparatively alike, such as records of meetings and persona information (usernames and IDs).

Our exploration of the Registry and in-depth analysis of the Program Log sets apart our research from many others in the literature. Mahr et al. [22] did not consider the Registry as a resource at all. Kerai [12] did examine Registry changes for RDS products, but at a level much too high to be helpful for any Investigator or developer to create an automated plugin. The authors only noted parent Registry and filesystem locations and broad details on information stored there.

## **7. Conclusions**

We believe we have proven the value of forensic research into RDS beyond doubt. Without understanding such artefacts in a case involving TeamViewer, an Investigator would not be able to say with any level of certainty who was responsible for any suspected actions.

Our work has highlighted a potential privacy concern in the sense of remaining artefacts following an uninstallation. When they uninstall TeamViewer, a user may expect that their associated connection history is also removed, but we have proven this does not occur. Additionally, there were many files in temporary locations which did not appear to be removed between application startups as we would expect.

### **7.1. Suggestions for Future Work**

Because of the prevalence of RDS but the lack of any formal forensic research into them, there are a wide range of options for future work.

Focusing specifically on TeamViewer, research into other features and versions may yield even more artefacts of interest. One of the variables which we could experiment with is the connection type. We looked at only standard and LAN connections. TeamViewer is able to facilitate connections via its own servers and has other approaches for direct connections, such as via port 80.

TeamViewer software for non-Windows OS may be significantly different enough to warrant its own work. Initial, unverified testing of our plugin against an example Debian Linux installation found that many of the primary artefacts are similar, such as the logs found under `/opt/teamviewer/logfiles`. We expect the key difference to be amongst those artefacts stored in the Windows Registry, as Linux does not have a direct equivalent.

Additionally, we could expand our work to cover volatile (in-memory) artefacts and network traffic analysis. Each of these, assuming a formal enough methodology, may justify entire projects themselves.

Another interesting avenue to explore would be TeamViewer Anti-forensics, with respect to the artefacts discovered during our research. Understanding how usage of TeamViewer could be concealed, planted or corrupted, and the detection of such activity would allow an Investigator to have further confidence in their evidence. TeamViewer activity involves multiple parties, but an Investigator may only have access to a single session endpoint and therefore would not necessarily be able to corroborate evidence separately. Even if there is access to both the Controller and Slave, conflicting facts may arise, and anti-forensic techniques could resolve them.

Although TeamViewer is one of the more popular RDS solutions, several others with significant market share exist. From a consumer perspective, a comparison of the forensic privacy implications of TeamViewer with its main competitors could be of significant value. Results could highlight relative weaknesses and help customers and organisations make informed decisions regarding RDS use. For example, some software may store passwords or other personal information in plaintext on a machine. Therefore a user may need to adapt which passwords they use for it or, more likely, avoid it altogether.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Notes on contributor

*Jonathan Manson* is a seasoned Cybersecurity Researcher and Software Engineer, with a BSc (Hons) in Computer Science and an MSc in Advanced Security and Digital Forensics.

## ORCID

Jonathan Manson  <http://orcid.org/0000-0002-0615-3003>



## Glossary

**Controller** TeamViewer session participant that is controlling another. In some sources this may be considered the Client. 5

**NAT** Network Address Translation. 15

**Network Address Translation** A process by which multiple devices can share public IP space. 15

**RAT** Remote Access Trojan. 2

**RCV** RegistryChangesView. 7

**RDS** Remote Desktop Software. 2

**Remote Access Trojan** RDS used for malicious purposes. The software does not have to be written with this purpose in mind. 2

**Remote Desktop Software** Software used to provide a virtual desktop environment of one computer to one remotely. 2

**Slave** TeamViewer session participant that is being controlling by another (the Controller). In some sources this may be considered the Server. 5

**TeamViewer Master Server** Server operated by TeamViewer that provides authentication and routing of traffic. 4

**UDP** User Datagram Protocol. 15

**VHD** Virtual Hard Disk. 9

**Virtual Hard Disk** File format representing a virtual hard drive, often used in virtualisation. 9

**Virtual Machine** An emulation of a computer system using virtualisation software. 7

**VM** Virtual Machine. 7

## References

- [1] Muir M, Leimich P, Buchanan WJ. A forensic audit of the tor browser bundle. *Dig Inv.* 2019;29:118–128.
- [2] Ernest GD, Timothy AA, Kpangkpatri G. The use of remote access tools by system administrators today and their effectiveness: case study of remote desktop, virtual network computing and secure android app. *Int J Comput Appl.* 2016;136:10.
- [3] Office for National Statistics. Coronavirus and homeworking in the uk. <https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/bulletins/coronavirusandhomeworkingintheuk/april2020>, Office for National Statistics., Tech. Rep. July 2020.
- [4] Russon M-A. Us fuel pipeline hackers 'didn't mean to create problems', <https://www.bbc.co.uk/news/business-57050690>, publisher=BBC News, May 2021, cited 2021 May 14.
- [5] Miramirkhani N, Starov O, Nikiforakis N. Dial one for scam: a large-scale analysis of technical support scams. In 22nd Annual Network and Distributed System Security Symposium (NDSS). 2016. (Vol. 16, p. reprint).
- [6] Riley D. Remote assistance cloud software firm teamviewer to raise up to \$2.54B in IPO. *SiliconANGLE*, <https://siliconangle.com/2019/09/11/remoteteassistance-cloud-software-firm-teamviewer-raise-2-54b-ipo>, September 2019, cited 2021 May 14.

- [7] Cision. TeamViewer supports medical technology providers digitalizing the healthcare sector - siemens healthineers already uses teamviewer globally. PR newswire. <https://www.prnewswire.com/news-releases/teamviewersupports-medical-technology-providers-digitalizing-the-healthcare-sector-siemens-healthineers-already-uses-teamviewer-globally-301221426.html> 2021, cited 2021 May 04.
- [8] Enlyft. TeamViewer commands 33.39% market share in remote access. enlyft, <https://enlyft.com/tech/products/teamviewer>, May 2021, cited 2021 May 01.
- [9] Yang TY, Dehghantanha A, Choo KKR, et al, Windows instant messaging app forensics: facebook and skype as case studies. PloS one. 2016;11(3). <https://doi.org/10.1371/journal.pone.0150300>
- [10] Lee B. Teamviewer forensics (tested on v15). ben's IR notes, <https://benleeyr.wordpress.com/2020/05/19/teamviewer-forensics-tested-on-v15> May 2020, cited 2021 May 01.
- [11] Haq A. Digital forensic artifact of teamviewer application. medium, <https://medium.com/mii-cybersec/digital-forensic-artifact-of-teamviewer-application-cfd6290dc0a7> 2011, cited 2021 May 14.
- [12] Kerai P, "Remote access forensics for vnc and rdp on windows platform," Ph.D. dissertation, Edith Cowan University, 2010.
- [13] TeamViewer Support. Change logs teamviewer, <https://community.teamviewer.com/English/categories/change-logs-en>, April 2021, cited 2021 May 14.
- [14] Kerai P, Vekariya VM, "An exploration of artefacts of remote desktop applications on windows," 2016, possibly an unpublished thesis also.
- [15] Altschaffel R, Clausning R, Kraetzer C, et al., "Statistical pattern recognition based content analysis on encrypted network: traffic for the teamviewer application," in *7th International Conference on IT Security Incident Management and IT Forensics*. IEEE, March 2013, pp. 113–121.
- [16] TeamViewer. How to use: all you need to know, <https://www.teamviewer.com/en/documents>, December 2020, cited 2021 May 14.
- [17] TeamViewer Community. What is a teamviewer ID? Teamviewer community, <https://community.teamviewer.com/English/kb/articles/49515-what-is-a-teamviewer-id>, December 2021, cited 2021 May 14.
- [18] MITRE. Cve-2018-16550 [teamviewer brute-force bypass.], MITRE, Tech. Rep., September 2018, Online 2021 May 14: <https://cve.mitre.org>
- [19] WhyNotSecurity. Teamviewer, <https://whynotsecurity.com/blog/teamviewer>, February 2020, cited 2021 May 14.
- [20] MITRE. Cve-2018-14333 [teamviewer password storage.], MITRE, Tech. Rep., July 2018, Online 2021 May 14: <https://cve.mitre.org>
- [21] Dennis M. Demystify TVS file format. jerry's guide, <http://www.jerrysguide.com/tips/demystify-tvs-file-format.html> May 2016, cited 2021 May 01.
- [22] Mahr A, Cichon M, Mateo S, et al, Zooming into the pandemic! A forensic analysis of the zoom application. *F Sci Inter: Digital Investigation*. 2021;36 (30110):7.
- [23] Majeed A, Zia H, Imran R, et al., Forensic analysis of three social media apps in windows 10, in *12th International Conference on High-capacity Optical Networks and Enabling/ Emerging Technologies (HONET)*. IEEE, 2015, pp. 1–5.
- [24] Quick D, Choo KKR. Google drive: forensic analysis of data remnants. *J Network Comput Appl*. 2014;40:179–193.

- [25] Shashidhar NK, Novak D. Digital forensic analysis on prefetch files. *Int J Inf Secur Sci.* **2015**;4(2):39–49.
- [26] Shariati M, Dehghantanha A, Choo -K-KR. Sugarsync forensic analysis. *Aust J Forensic Sci.* **2016**;48(1):95–117.
- [27] Teing YY, Dehghantanha A, Choo KKR. Cloudme forensics: a case of big data forensic investigation. *Concurrency and Computation: Practice and Experience.* **2018**;30(5):e4277
- [28] Halkes G, Pouwelse J. Udp nat and firewall puncturing in the wild, in *International Conference on Research in Networking*. Springer, **2011**, pp. 1–12.
- [29] TeamViewer Community. Log file reading, <https://community.teamviewer.com/French/kb/articles/108789-log-file-readingincoming-connection>, April **2021**, cited 2021 May 28.