UNIVERSITY OF AMSTERDAM

Tunneling data over a Citrix virtual channel

February 9, 2020

Students:
W. Bakker
12246336

N. den Otter
12911283

Assessor:
Cees de Laat

Course:
RP1

Supervisors:
Cedric van Bockhaven
Marat Nigmatullin

Abstract

Citrix is a popular brand that offers the Virtual Apps and Desktops suite. This suite allows an organization to set up a Virtual Desktop Infrastructure (VDI). This VDI gives users remote access to an organization's applications. In the past Citrix servers have proven to be a stepping stone used by attackers to enter an organization's network[1]. However, the toolset that an attacker could use within the VDI was limited due to hardening of the Citrix environment.

Citrix uses virtual channels to extend a Virtual Apps and Desktop session's capabilities. This research investigates the potential of tunneling through an established Citrix session using virtual channels, in order to bypass e.g. firewall limitations.

Tunneling through an established Citrix session is very useful for moving laterally through a network. By moving laterally through a network it is possible to connect to systems that are not directly connected to the internet. This means it is possible to access the company's internal network through an externally facing system. Because the tunneling would happen within an established connection, security measures like firewalls would not block these connections.

During our research, we created a custom virtual channel which was capable of executing commands and receiving the output of these commands. By testing this virtual channel within a Proof of Concept environment, we demonstrated that it is possible to set up bidirectional communication through virtual channels. We also demonstrated that it is possible to move laterally through an organization's internal network.

1 Introduction

Within a business environment, Citrix Virtual Apps and Desktops (also known as XenApp and XenDesktop) can be used to offer employees remote access to applications or virtual desktops. The server does the necessary computations to process user input and data, and display the application/desktop on the client's device.

Citrix Virtual Apps and Desktops uses ICA as protocol stack for the delivery of remote applications and desktops. ICA is a proprietary protocol developed by Citrix.

Citrix Virtual Apps and Desktops uses virtual channels. Virtual channels are communication paths within a session and serve a single purpose like:

- Allowing a server to communicate with a client's locally connected device through USB or COM ports,
- Streaming audio or high quality graphics from the server to the client,
- Having bidirectional clipboard support (which makes copying and pasting text/data from/to either side possible).

Virtual channels are still used to this day. Citrix has added the ability to use custom channels within their software. Anyone can create a custom virtual channel and implement it for use with a Citrix session.

In the past similar research (as described in section 2.2) has proven that it is possible to tunnel TCP connections through protocols similar to ICA. This has shown that it is possible to turn existing sessions from these protocols into pivot points. These pivot points can be used to move laterally through a server's network. This allows clients from external networks to reach systems which are not reachable from the internet. Since connections are initiated from a server instead of a client on an external network, firewalls would not block these connections.

Because ICA uses virtual channels and previous research has shown that these channels could be used to tunnel through in similar protocols, we investigate the ability to use a virtual channel within a Citrix Virtual Apps and Desktops session to move laterally through the network and bypass firewalls.

Our report is structured as follows: section 2 discusses related work relevant for our research. Section 3 presents our research question. Section 4 gives background information on the ICA protocol and virtual channels. Section 5 outlines the methodology used to answer the research question. Section 6 presents the results. Section 7 contains our conclusion which answers our research question. Section 8 contains a discussion. The paper concludes with section 9 which discusses possible future work.

2 Related work

During our research we have not found previous research which targeted tunneling over the ICA protocol stack. However, the possibility of tunneling through other (similar) protocols has been researched. This chapter will elaborate on some of these possibilities.

2.1 Bypass restrictions

Amirante et. al [2] discussed how HTTP tunneling can circumvent restrictive firewalls, proxies and Network Address Translation. His insight presents the importance of tunneling. Reardon et. al [3] discussed their solution to reduce latency within Tor, in which they propose to multiplex each TCP stream onto a single Datagram Transport Layer Security (DTLS) connection. This allows them to use an existing DTLS connection to tunnel data through. The proposed solution is similar to what Citrix does with it's virtual channels. By re-using an already established connection, security measures like Citrix NetScaler will not interrupt the connection.

2.2 Practical implementation rdp2tcp

A practical approach is the rdp2tcp project. This project has made it possible to tunnel TCP connections through the Remote Desktop Protocol (RDP), offered by Microsoft. RDP allows users to access applications or virtual desktops and makes use of virtual channels, similarly to ICA. Tunneling TCP connections through RDP is done by using the virtual channels of the RDP protocol and using port forwarding over an existing RDP connection[4]. Tunneling over RDP makes it possible for a client outside the network to reach systems that are otherwise not accessible from the public network. This idea of tunneling is used as a basis for our research towards tunneling over the ICA protocol.

Note that while RDP and ICA share similarities, e.g. virtual channels, they are different protocols. While rdp2tcp has shown that it is possible to tunnel through RDP, it does not inherently mean that the same is possible through ICA as ICA works in a different way.

3 Research Question

With the previous research in mind, we want to investigate the possibility to tunnel over a Citrix Virtual Apps and Desktops session using virtual channels. Furthermore, we want to be able to bypass basic security measures like a firewall. We have defined the following research question:

How can virtual channels, used within the Citrix Virtual Apps and Desktops suite, be used to move laterally through a private network from an external connection?

4 Background

4.1 Independent Computing Architecture

Independent Computing Architecture (ICA) is a Citrix proprietary protocol stack, designed for the reliable transport of data related to a Citrix environment. This protocol stack is comprised of two transport protocols which in turn have Virtual Channels (VCs) that use these protocols. The transport protocols are Enlightened Data Transport (EDT) and TCP, while the VCs enable interaction with a virtual desktop environment [5]. ICA is designed to reduce bandwidth requirements while simultaneously ensuring session reliability[6].

4.1.1 ICA transport protocol

Key functionality of TCP is it's reliability. However, this reliability introduces extra latency. To reduce this latency Citrix introduced EDT. EDT is a proprietary transport protocol with the reliability property of TCP and the low latency property of UDP. This is achieved by adding a reliability layer on top of UDP, as illustrated in figure 1. Both transport protocols can be used to send data via VCs. At the application layer the overarching Adaptive Transport mechanism detects if it is possible to use EDT. If an EDT based connection is not possible the connection will fallback to TCP [7].

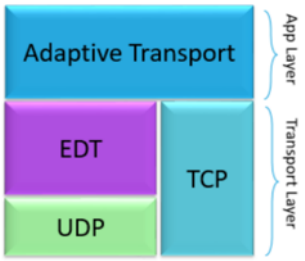

Figure 1: Adaptive transport contains Enlightened Data Transport and TCP [5]

4.2 Virtual Channels

4.2.1 Functionality

VCs are communication paths within an established ICA session. For a session to be usable, Citrix loads a number of default VCs. Other VCs can be added to allow for more functionality. For example, graphics are handled by the Thinwire virtual channel (VC) while clipboard support is handled by the Clipboard VC. To support the usage of VCs the client requires a virtual driver to be loaded and the server an application to be started. The client-side virtual driver is usually a .DLL file which is loaded by the Citrix Receiver when it is launched[8]. Citrix Receiver is the application used to connect to the virtual desktop.

When a client connects to a Virtual Apps and Desktops server it sends information about the VCs it supports. The server then starts the supporting applications for each VC and obtains the handles for each channel[9]. Figure 2 shows that each VC has one purpose. This means that, for example, one VC takes care of visualization while another takes care of user input.

A maximum of 32 VCs can be used within a Citrix session. Seventeen of these channels are reserved by Citrix, which enables organizations to load fifteen custom channels. While Citrix has seventeen VCs available for use, only four channels are mandatory to set up a session. These are the Thinwire, LicenseHandler, ICACTL and TWI channels [9]. A description of all VCs from Citrix can be found in Appendix A on page 14.

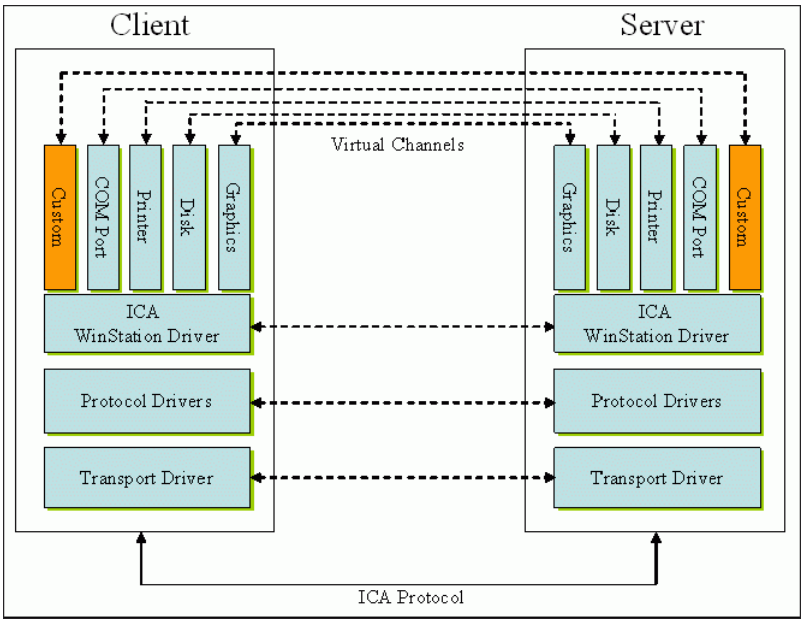

Figure 2: Image describing how virtual channels are managed using WinStation [9]

4.2.2 Disabling channels

It is possible for the VCs included with the Citrix software to be administratively disabled. An administrator can decide for each VC whether a client is allowed to use that channel. It is among other things possible to allow or deny usage of a channel based on the attributes of the user, the location the user connects from, the client used to connect with and the desktop that the user connects to. It is not possible to disable the VCs required for the session to successfully set up.

We cannot confirm nor deny that the usage of custom VCs can be restricted. Citrix does not appear to have any documentation on this and we could not find any related settings within our Citrix environment. It would be possible to limit the usage of custom VCs by restricting the user's ability to load these client-side and disallowing the uploading and/or launching of unauthorized applications on the server. Restricting the user's ability to load custom channels client-side could prove difficult as often an employee's own device can be used (which is usually not managed by the company) to connect to a Citrix environment.

4.3 Virtual Channel SDK

Citrix offers a Virtual Channel SDK (VCSDK). This SDK makes it easier to develop a custom VC. The SDK provides the Citrix Virtual Driver Application Programming Interface (VDAPI) which is used in conjunction with the Citrix Server API SDK (WFAPI SDK) to create new VCs.

Bundled with this SDK are example channels which, after compiling, are ready for use by loading them on both the server and client. These example channels are[10]:

- Ping: Records the round-trip delay time for a test packet sent over a virtual channel,
- Mix: Demonstrates a mechanism to call functions (for example, to get the time of day) on a remote client,
- Over: Simple asynchronous application where the server must receive a response from the client asynchronously and responds with a different packet than the one received.

When compiling a VC, two files are delivered. One file is the client-side driver in the form of a .DLL file. This file is used by the Citrix Receiver and is loaded when the Receiver is launched. The other file is an executable which is used by the server. This executable takes care of opening the VC and transmitting data through the channel.

5 Method

5.1 Virtual channel implementation

To be able to tunnel data through a VC it is necessary to either manipulate the transmitted data from a VC, alter the way an existing VC operates or create a new VC designed for tunneling data. We decided to create a new VC for our research.

We decided to modify the **Over** example channel from the VCSDK. This example channel already has mechanisms in place to have the client and server respond to each other. We would have to expand this functionality to allow us to execute commands we defined ourselves. These commands should interact with network devices (e.g. pinging a device or tracing the route to an IP). The server would also have to send back output from the executed command to the client. The data that would be sent back and forth would be transmitted only over this channel, which would demonstrate the ability to send and receive data through a VC.

For this research we will be focusing on Windows as Operating System (OS) for both the client- and server-side. While it is possible to create and compile VCs for Linux-based clients, this is out of scope for this research.

5.2 Test environment

Our research is solely focused on tunneling through a Virtual Apps and Desktops session. To test this tunneling, we set up our own test environment. Note that for the remainder of this document we refer to Virtual Apps and Desktops as XenDesktop; this is the old name of the Desktop part of the Virtual Apps and Desktops suite. We use this name due to the used versions of software within the test environment (explained in section 5.2.2).

5.2.1 Environment setup

The test environment was set up on a single physical machine. The machine contained an Intel(R) Core(TM) i7-8700 CPU @ 3.20Ghz, 16GB DDR4 memory and a 240GB SSD. The operating system (OS) used is Windows 10, using VMware Workstation 15.5.1 as hypervisor. Figure 3 demonstrates the test environment.

Five virtual machines (VM) are used of which two are based on Windows 2008 R2 and one on XenServer 5.6. One Windows Server contained the Active Directory Domain Controller role while the other contained a Citrix Controller. The XenServer VM hosts the remaining two VMs which are Windows 7 machines. These machines functioned as a XenDesktop server.

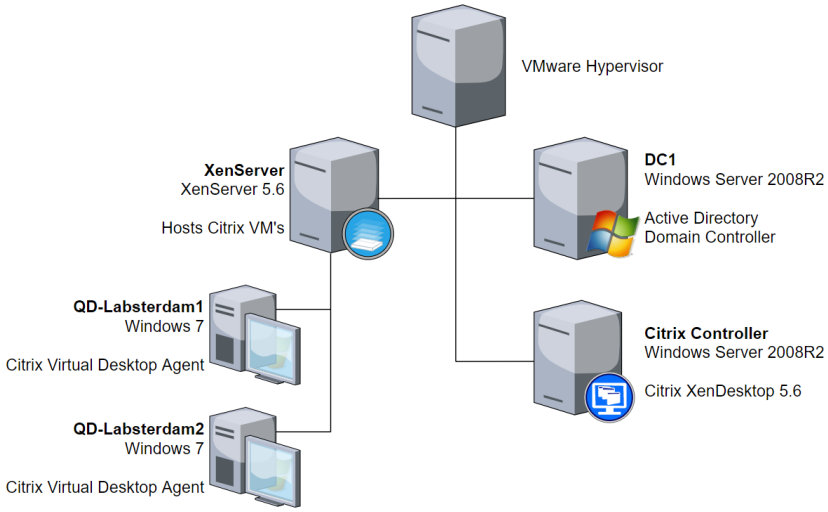

Figure 3: Test environment

The Active Directory Domain Controller is required for the Citrix suite to function. Citrix uses Active Directory for multiple purposes, e.g. centralized authentication, computer management and the application of policies.
The Citrix Controller allows users to log in to the virtual environment, does session management, applies policies and does management of VM's within the Citrix environment. Note that the controller only assists with setting up a session; sessions within our environment take place directly between a VM and Citrix client.
The XenServer VM contains a hypervisor which hosts the two Windows 7 VMs. These VMs contain the Citrix Virtual Desktop Agent (VDA). These VDAs are used by Citrix to offer users access to a desktop. The Windows 7 VMs are referenced to as a XenDesktop server within this document.

The Citrix environment was set up using the Quick Deploy feature within XenDesktop. This feature makes it easy to quickly set up a Proof of Concept (PoC) environment by simplifying the installation and configuration of the Citrix environment.

A Windows 10 client was used to connect to the Citrix environment. We had administrative privileges on this client.

5.2.2 Used software

For this research we used version 5.6 of XenDesktop, which received End of Life (EOL) status as of 17 June 2015. The reason for using this version is due to Citrix having changed their licensing policy since November 2019. This resulted in users not being able to request a trial license for Citrix software by themselves.

Because we could not obtain a license for newer versions of the required software, we were forced to use XenDesktop version 5.6. To legitimately use this version we used the built-in 30-day trial option. Since we were forced to use an old version of XenDesktop, we were also required to use older versions of Windows as newer versions were not supported by XenDesktop. The highest supported Windows versions are Windows Server 2008 R2 as server OS and Windows 7 as client OS.

6 Results

6.1 Modifying provided virtual channel

We created a custom VC by modifying the example **Over** channel from the VCSDK. This channel is included with the VCSDK and uses C as programming language.

The basic function of the Over channel is to have the server receive a response to a packet from the client. When the VC is opened by launching the executable on the server, the executable starts sending sequence numbers to the client. The client inspects these sequence numbers, increases the sequence number by 7 whenever the received sequence number is divisible by 10 and returns the sequence number.

6.2 Expanding the example channel Over

We modified the Over channel in the following ways to expand the existing functionality:

1. A console attached and opened when Citrix Receiver is launched,
2. The console accepts commands and arguments from the user and sends these to the XenDesktop server,
3. The server keeps the connection alive by repeatedly polling the client for a command to execute,
4. The server executes commands through the Windows command processor,
5. The server sends output from executed commands back to the client.

These items are explained in more detail in the following subsections.

6.2.1 Allocating and attaching a console

When launching the Citrix Receiver, our channel allocates and attaches a console. This console is shown together with the display of the XenDesktop server. This console is used to display output from the VC and takes input from users (see subsection 6.2.2).

6.2.2 Sending commands and arguments to the server

The client-side console accepts input from a user. Users can execute a number of predefined commands by inputting an integer. Table 1 displays the available commands. Commands containing brackets indicate that the user also needs to input an argument, usually an IP or hostname.

Integer	Command	Description
3	pause	Waits for the user to press enter to continue
4	exit	Exits the application on the server
11	route print	Print available IPv4 and IPv6 routes on the server
12	netstat -a	Lists all currently opened IPv4 sockets
13	ping [IP]	Ping an IP
14	nslookup [IP]	Look up an IP using DNS
15	tracert [IP]	Trace the route used to reach specified IP
16	ipconfig /registerdns	Register this computer with the DNS server

Table 1: Commands available with the custom virtual channel

6.2.3 Server-client polling

The XenDesktop server polls the client every 500ms for a command to execute. This is done by sending a sequence number of 1 to the client. This is repeated until the client returns a different sequence number indicating a command to execute. The command is executed and the server starts sending a sequence number of 1 again every 500ms. A sequence number that does not match a command is ignored.

6.2.4 Executing commands through the Windows command processor

Commands can be executed through the command processor. The command processor is comparable to the Run dialog within Windows which allows users to directly execute commands without having to open a terminal. The program that is requested by a command is launched directly instead of through a terminal.

6.2.5 Returning command output to the client

After having executed a command, the server will return a sequence number of 1 to the client together with the output of the previously executed command. This output will be shown on the client's console.

6.3 Compiling and launching the channel

After modifying the code we compiled it which resulted in a .DLL and .EXE file. To reiterate: the .DLL file is used client-side by the Citrix Receiver while the .EXE is launched on the server to establish and use the VC.

We copied the .DLL file to the installation folder of Citrix Receiver and added it to the appropriate place within the Windows registry. This placement within the registry indicates that this VC should be loaded by the Citrix Receiver. The registry key we modified is `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Citrix\ICAClient\Engine\Configuration\Advanced\Modules\ICA 3.0\VirtualDriverEx`. We added the value `VDOVER` to this key. This value is the name of the .DLL file placed in the application's directory.

We logged in to the Citrix environment as an unprivileged user and initiated a session. The Citrix Receiver launched together with a console window. The console window displayed text indicating it successfully opened the VC.

We uploaded the .EXE file to the server and launched it through the Windows terminal. The terminal indicated it sent a sequence number of 1 and waited to receive data from the client.

By inputting the number thirteen into the console on the client-side, we were asked for an IP. After inputting an IP address the server started pinging the IP-address and returned output of the command back to the client. Figure 4 shows the VC being opened on both the client- and server-side, the ping command being executed and the output being returned from the server.

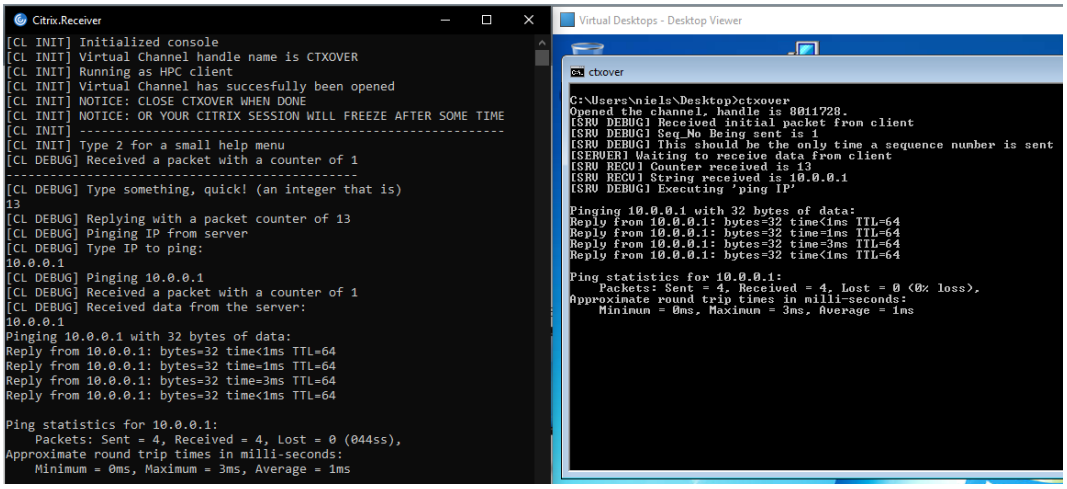

Figure 4: Demonstrating client (left) and server (right) communication

After the server completed the ping command we were again requested to input an integer in the client's console. We entered number sixteen which launched the ipconfig /registerdns command. The server executed this command and received an error with the message 'The requested operation requires elevation'. This indicates our user was not authorised to execute this command.

7 Conclusion

Our research investigated the possibility to tunnel data over a Citrix Virtual Apps and Desktops session using VCs. By researching the ICA protocol stack, we were able to successfully set up and use a custom VC within our test environment.

We were able to successfully launch commands through the custom VC and retrieve output from these commands. By doing this, we have shown that it is possible to reuse an existing Citrix connection to send and receive data through. This in turn shows that it is possible to use VCs within Citrix applications as a pivot point into a company's internal network.

While the VC we compiled was limited in what it could do, it has shown that it is possible to create a custom VC that allows bi-directional communication between a Citrix server, the server's internal network and a client outside this network.

8 Discussion

With our results we have shown that it is possible to use an established XenDesktop version 5.6 session to tunnel data through. By creating a new VC and loading it on both the client- and server-side, we were able to get bidirectional communication working. This in turn allowed us to gather information from the target system and network while bypassing security mechanisms like firewalls. This means the XenDesktop server was being used as a stepping stone to pivot into the server's internal network.

8.1 Modifying Windows registry

We were able to set up the VC on the server as an unprivileged user. No administrative privileges were required to launch the executable that sets up and connects to the VC. To set up the VC on the Windows client we were required to make registry edits in the Local Machine hive. Modification of this hive is restricted to administrators. Depending on the situation, setting up a custom VC might be an issue if the Citrix client can only be used on a computer which has been hardened or when a user account with no administrative privileges is used. In most cases it is possible to use an own device to connect remotely to the Citrix environment; this would negate these concerns.

8.2 Network traffic from a virtual channel

Communication from the custom VC is encapsulated within the session together with all other VCs[11], meaning the execution of commands together with returning output is done through an established TCP connection. This makes it so the network traffic that is generated by the VC looks like legitimate network traffic from the Citrix session. This is an interesting perspective for both a red and blue team.

Because the traffic of the custom VC appears legitimate, it will be hard to detect. If a red team were to infect the Citrix server with Command and Control (C&C) malware, the connection initiated by this malware would usually be blocked since it would require opening a new TCP connection. Opening this connection would be blocked by the firewall. By using a VC, the C&C software can make use of the existing TCP connection from the Citrix session to communicate through.

This means that a blue team would have to use different measures to identify the traffic originating from a Citrix server. While sniffing traffic *to* the Citrix server might no longer be interesting, sniffing traffic *from* the Citrix server could show any suspicious behavior. An example of suspicious behavior is the Citrix server scanning the private network for other (potentially vulnerable) systems. This would certainly raise concerns within a blue team. A blue team would also have to check for newly created VCs. To completely block the usage of custom VCs, a blue team could decide to deny launching non-whitelisted executables on the Citrix server. Since custom VCs are opened by launching an executable, completely blocking the possibility to launch these would prevent the usage of custom VCs. The red team would have to use a different technique to establish the virtual channel.

8.3 Execution of commands

We observed that commands are executed as the currently logged in user, meaning commands requiring administrative privileges gave an access denied error. This means that any compromised Citrix account can be used to execute commands with the custom VC; it does not have to be an administrator.

8.4 Tunneling data

The concept of tunneling data through a Citrix session has been shown working within a PoC. Since Citrix environments can be customized to be suited for a certain organization, limitations and restrictions will vary for each environment. This means that custom VCs (like the one we have demonstrated) might need adjustments to provide the same functionality as shown in this report.

Based on our results, we have successfully demonstrated that it is possible to use VCs as a tunnel to communicate with other network devices.

Please note that we have gathered our results from a PoC environment using old software which is no longer supported. Our tests have been executed using direct connections without any sort of intrusion detection- or intrusion prevention systems. While the used software is old, the concept and implementation of VCs has remained largely the same during the years.

9 Future work

By answering our research question, we have demonstrated a new method of communicating with a target (network) using a VC. While we have tried to make our research as extensive as possible, there are some areas that could be improved:

- Server-side binary injection,
- Expand code for usage of custom applications,
- Reverse engineering the ICA protocol,
- Test performance of the custom VC.

While our VC worked by executing a file which was placed on the hard drive of the Citrix server, further research can determine whether the channel can be used through other methods like binary injection. With binary injection the executable gets loaded directly into memory without needing to be saved to the disk. This reduces the chances of being detected by an anti-virus program while circumventing the restriction of launching executables.

While the VC we developed shows that it's possible to tunnel through VCs, it by no means is finished. While it is possible to execute commands by inputting an integer, it would be more interesting to have the VC act as a passthrough channel or proxy for network traffic. This would make it possible to tunnel traffic to the internal network of the Citrix server from an own device, using the Citrix server as a gateway into the network.

So far, the ICA protocol has not yet been reverse engineered. This makes it tougher to understand how the protocol functions and how data from VCs is treated. Reverse engineering the protocol could give more insight into other methods to achieve the same goal as described in this paper.

We would have liked to examine the performance that a VC can bring but were not able to. By investigating the throughput of VCs, the usability of such channels can be shown from both red and blue teaming perspectives. If it turns out that VCs are quite performant, they could allow the usage of scanning tools like nmap. On the other hand, if it turns out that VCs are only capable of providing limited throughput (e.g. barely enough for text transfer as shown with our custom channel), the usefulness of these channels from a red teaming perspective would be limited. Meanwhile, a blue team would have to take extra measures to make sure VCs cannot be abused to execute malicious tasks like scanning the network or the extraction of data.

References

[1] "Attacks on adc ramp up as citrix releases remaining patches — securityweek.com."

[2] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, "Ntrulo: A tunneling architecture for multimedia conferencing over ip," in *Smart Spaces and Next Generation Wired/Wireless Networking*, pp. 460–472, Springer, 2010.

[3] J. Reardon and I. Goldberg, "Improving tor using a tcp-over-dtls tunnel.," in *USENIX Security Symposium*, pp. 119–134, 2009.

[4] V. V.E.O, "Rdp2tcp," 2019.

[5] F. Klurfan, "Enlightened data transport," 2017.

[6] P. Serwan, "Dive into citrix ica protocol – part1," 2014.

[7] Citrix, "Adaptive transport," 2019.

[8] Citrix, "Citrix ica virtual channels overview." `https://support.citrix.com/article/CTX116890`.

[9] Citrix, "Citrix ica virtual channels." `https://docs.citrix.com/en-us/citrix-virtual-apps-desktops/technical-overview/virtual-channels.html`.

[10] "Using example programs - citrix virtual channel sdk for citrix receiver for windows."

[11] "Architecture - citrix virtual channel sdk for citrix receiver for linux 13.5."

Appendices

A List of Virtual Channels

Table 2 contains a list of all Virtual Channels Citrix makes available within their Virtual Apps and Desktops software, together with a description of their functionality:

Channel name	Description
Thinwire3.0 (Required)	Handles all graphics
ClientDrive	Allows the server to access a local device's disks
ClientPrinterQueue	Allows the server to access a local device's printers
ClientPrinterPort	Tells the server what port a local device's printer uses
Clipboard	Supports copying/pasting of data/text to/from the server
ClientComm	Allows the server to access a local device's COM ports
ClientAudio	Plays audio from the server on the local device's speakers
LicenseHandler (Required)	Manages the licensing of desktops
TWI (Required)	Supports seamless configuration, making Virtual Apps appear as if they are run locally
SmartCard	Adds smartcard authentication functionality
Multimedia	Allows client to render multimedia features (audio/video) requested on the server
ICACTL (Required)	Manages miscellaneous client communication
SSPI	Allows the usage of security providers (for example, Kerberos)
TwainRdr	Redirects TWAIN requests, used for scanners
UserExperience	Records metrics from a Citrix session
Vd3d	Used for desktop composition redirection (deprecated)

Table 2: Virtual Channel names and descriptions

The 'Required' tag indicates that this Virtual Channel is necessary to set up a Citrix session[2].