Security Auditing Report

Teleport RFD 33-37

RDP Desktop Access for Windows

Prepared for: Gravitational, Inc. DBA Teleport
Prepared by: Lorenzo Stella, Norbert Szetei and Ben Caller
Date: 01/23/2023

Table of Contents

Revision History

Version	Date	Description	Author
1	01/14/2022	First release of the final report	Norbert Szetei, Ben Caller, Lorenzo Stella
2	01/14/2022	Peer Review	Lorenzo Stella
3	01/17/2022	Peer Review	Norbert Szetei
4	11/11/2022	Retesting	Sercan Sayitoglu
5	11/15/2022	Improved retesting result notes	Luca Carettoni
6	01/23/2023	Pre-publication review	Anthony Trummer

Contacts

Company	Name	Email
Gravitational, Inc	Russell Jones	rjones@gravitational.com
Gravitational, Inc	Sasha Klizhentas	sasha@gravitational.com
Doyensec, LLC	John Villamil	john@doyensec.com
Doyensec, LLC	Luca Carettoni	luca@doyensec.com

Executive Summary

Overview

Gravitational, Inc (DBA "Teleport") engaged Doyensec to perform a security assessment of the Teleport Desktop Access capabilities. The feature brings identity-based, passwordless RDP access to Windows hosts across all cloud, on-premises, and edge environments.

The audit was split into the following two-phase schedule:

- The first phase (A) commenced on 12/06/2021 and ended on 12/17/2021, involving two (2) security researchers performing fuzz-based testing of the Teleport RDP Client and its custom Desktop protocol. The efforts to develop the appropriate tooling for this task are documented in *Appendix C - "RDP Fuzzing"*. After this first phase (A) and before the second (B) began, Doyensec ran the fuzzing tools for a time span of two weeks, after which their final results were analyzed in (B). This project phase resulted in twelve (12) unique crashes causing panics in the *rdp-rs* library used by the main thread and consequently killing the Teleport daemon.

- The second phase (B) commenced on 01/03/2022 and ended on 01/14/2022 requiring two (2) security researchers. In this phase the Desktop Access feature was audited as a whole, focusing on the `windows_desktop_service` Gateway Modes, the Web Client UI, and RBAC Bindings elements. This project phase resulted in eleven (11) findings, of which seven (7) were rated as *Low*.

Doyensec security engineers retested the entire platform against all outstanding vulnerabilities. This document summarizes the state of the finding as of 11/11/2022.

The project consisted of a manual web application security assessment, and fuzzing .

Testing was conducted remotely from Doyensec's EMEA and US offices.

Scope

Through meetings with Gravitational, the scope of the project was clearly defined.

- Identify misconfigurations and vulnerabilities in the implementation Desktop Access features of Teleport Community and Enterprise:
  - RFD 33 - Desktop Access
  - RFD 34 - Desktop Access - Windows
  - RFD 35 - Desktop Access - Windows Certificate Authentication
  - RFD 37 - Desktop Access - Desktop protocol

- Evaluate the overall security posture and best practices compared to other industry peers

We list the agreed-upon assets below:

- Teleport Community
  - https://github.com/gravitational/teleport

- Teleport Enterprise
  - https://github.com/gravitational/teleport.e

The testing took place in multiple development environments using the latest version of the software features at the time of testing.

In detail, this activity was performed on the following releases:

- Teleport 8.0.7
  - Cloned at commit SHA 383bf998 (*master*) on 01/03/2022

Scoping Restrictions

During the engagement, Doyensec did not encounter any major difficulties testing the functionalities of the application. The Gravitational engineering team was very responsive in debugging any issue to ensure a smooth assessment.

At the time of testing, the following aspects of the features in scope were still not implemented:

- Session recording and video export
- Clipboard integration
- Windows restricted session
- `windows_desktop_service` Agent mode
- Bi-directional File Transfer
- Per-session Desktop MFA
- Support for multiple CA certs per cluster
- HSM integration

While testing included the review of the Teleport internal dependencies, Doyensec did not perform a complete source code review for all packages involved in the scoped features. The Windows RDP server implementation and its GPO policies were also not audited. As previously agreed, only the Windows versions listed in the "Setup Phase" of the "Methodology" section were tested.

It is also important to notice that Teleport is a highly flexible platform in which several configurations can be customized by the end-user. For instance, permissions for roles/users are completely customizable, hence Doyensec focused on vulnerabilities in the core logic instead of enumerating potential misconfigurations in user-defined policies.

Findings Summary

Between the first and second phases, Doyensec researchers discovered and reported *twenty-three (23)* total vulnerabilities in Teleport's Desktop Access platform. Most of the issues were departures from best practices and low-severity flaws.

It is important to reiterate that this report represents a snapshot of the environment's security posture at a point in time.

During the fuzzing phase (A), we identified several vulnerabilities causing the panic of the main thread and teleport daemon. All revealed vulnerabilities stem from the *rdp-rs* implementation and trusting the server-provided data during processing. In most cases, a potentially malicious RDP server can overflow or underflow the integer data type using arithmetic operations without bounds checking.

As the Rust language is designed to be memory safe, instead of memory corruptions, we currently assume an attacker could only exploit the revealed findings for a Denial of Service attack. A second code review was performed to identify potential vulnerable patterns and evaluate the overall risk with more certainty.

In the second phase (B), the findings included multiple vulnerabilities in the design and implementation of some features. A number of Insecure Design practices were highlighted, particularly on the setup of the solution (TEL-Q122-2) and the complementary relationship between Teleport ACL and Windows GPO (TEL-Q122-3). In the same category, a way to force the RDP disconnection of a user was also identified (TEL-Q122-5). A bypass of the Teleport login restrictions (TEL-Q122-4) and multiple smart card abuses (TEL-Q122-7, TEL-Q122-7) particularly afflicted the security of the RDP authentication mechanism. Some Denial Of Service (TEL-Q122-1, TEL-Q122-8) and Information Exposure (TEL-Q122-6) risks were also discovered during the audit.

Overall, the security posture of the Internet-facing APIs was found to be in line with industry best practices.

At the design level, Doyensec found the system to be well architected with the exclusion of the following aspects:

- Lack of enough documentation around the risks and hardening practices that should be applied alongside the solution

Recommendations

The following recommendations are proposed based on studying Teleport Desktop Access' security posture and the vulnerabilities discovered during this engagement.

Short-term improvements

- Work on mitigating the discovered vulnerabilities. You can use **Appendix B** - Remediation Checklist to make sure that you have covered all areas

Long-term improvements

- Migrate to certificate-based authentication for Active Directory (`teleport/lib/srv/desktop/ldap.go`).
- Improve the Windows Desktop locking by not only terminating the websocket live connection, but by working together with LDAP's certificate revocation list mechanism.
- Since Rust-originated panics can terminate the Teleport daemon, run the Rust code in a different process / thread or catch the exceptions to reduce the impact of future Denial of Service (DoS) issues.
- Sanitize the username to avoid domain spoofing in (`teleport/lib/srv/desktop/windows_server.go`).
- Ensure during the setup of the solution that the appropriate security GPOs are set in place. This includes:
    - Restricting the user groups capable of joining new workstations to the Windows domain
    - Creating tailored GPO rules on the OU that will be enrolled in Teleport, reflecting the configuration enforced by the roles
- Check if either Powershell Remoting, Powershell Web Access (PSWA), or Windows Remote Management (WinRM) capabilities are enabled
- The Rust RDP client is prone to panicking when it receives unexpected data from the RDP server. Sanity check data received from the RDP server especially when performing arithmetic operations which could overflow or underflow and when calling `unwrap` on `Option` types.

Retesting Overview

Most of the issues were addressed in a timely manner by Teleport. However, there are some security vulnerabilities that would require mitigations or workarounds. Due to the complexity of those changes, Teleport has decided to mitigate those issues in upcoming releases.

Methodology

Overview

Doyensec treats each engagement as a fluid entity. We use a standard base of tools and techniques from which we built our own unique methodology. Our 30 years of information security experience has taught us that mixing offensive and defensive philosophies is the key to standing against threats. Thus we recommend a *whitebox* approach combining dynamic fault injection with an in-depth study of the source code to maximize the ROI on bug hunting.

During this assessment, we have employed standard testing methodologies (e.g., OWASP Testing guide recommendations), as well as custom checklists, to ensure full coverage of both code and vulnerability classes.

Setup Phase

Gravitational provided access to dedicated Teleport environments, source code repositories, and binaries for all components in scope. The Teleport cluster used during testing was available at https://doyensec-win.gravitational.io, pre-configured with the following Desktops:

- Windows Server 2016 Datacenter v10.0 (14393)
- Windows Server 2019 Datacenter v10.0 (17763)

Doyensec later added a virtualized Windows Server 2012 R2 host.

Tooling

When performing assessments, we combine manual security testing with state-of-the-art tools in order to improve efficiency and efficacy of our effort.

During this engagement, we used the following tools:

- Burp Suite
- Protobuffer Decoder
- Protoc
- Gosec
- golangci-lint
- Semgrep
- CyberChef
- Curl, netcat and other Linux utilities

Web Application and API Techniques

Web assessments are centered around the data sent between clients and servers. In this realm, the principle audit tool is Burp Suite. However, we also use a large set of custom scripts and extensions to perform specific audit tasks. We focus on authorization, authentication, integrity and trust. We study how data is interpreted, parsed, stored, and relayed between producers and consumers.

We subvert the client with malicious data through reflected and DOM based Cross Site Scripting and by breaking assumptions in trust. We test the server endpoints for injection style flaws including, but not limited to, SQL, template, XML, and command injection flaws. We look at each request and response pair for potential Cross Site Request Forgery and race conditions. We study the application for subtle logic issues, whether they are authorization bypasses or insecure object references. Session storage and retrieval is scrutinized and user separation is thoroughly tested.

Web security is not limited to popular bug titles. Doyensec researchers understand the goals and needs of the application to find ways of breaking the assumed control flow.

Desktop and Server Applications

Doyensec has extensive experience finding flaws in thick clients, standalone binaries, and server daemons. We write customized tools to map out control flow and study an application's behavior and internals. Mapping out attack surface, whether local or remote, is paramount to a successful engagement. Doyensec also studies the application's ecosystem, looking for potential pitfalls and common misconceptions.

Fuzzing and instrumentation are important parts of the SDLC which we use to test an application's response to untrusted data. Doyensec has a mature, automated, and flexible fuzzing framework which will be customized to meet the needs of the target and detect its response to malicious input continuously over the testing period.

We deconstruct the application looking for privacy leaks and secrets. We understand the storage, transmission, and protection of user information is critical, along with the server-side handling of user provided data.

Project Findings

The table below lists the findings with their associated ID and severity. The severity ranking and vulnerability classes are defined in **Appendix A** at the end of this document. The vulnerability class column groups the entry into a common category, while the status column refers to whether the finding has been fixed at the time of writing.

This table is organized by time of discovery. The issues at the top were found first, while those at the bottom were found last. Presenting the table in this fashion has a number of benefits. It inherently shows the path our auditing took through the target and may also reveal how easy or difficult it was to discover certain findings. As a security engagement progresses, the researchers will gain a deeper understanding of a target which is also shown in this table.

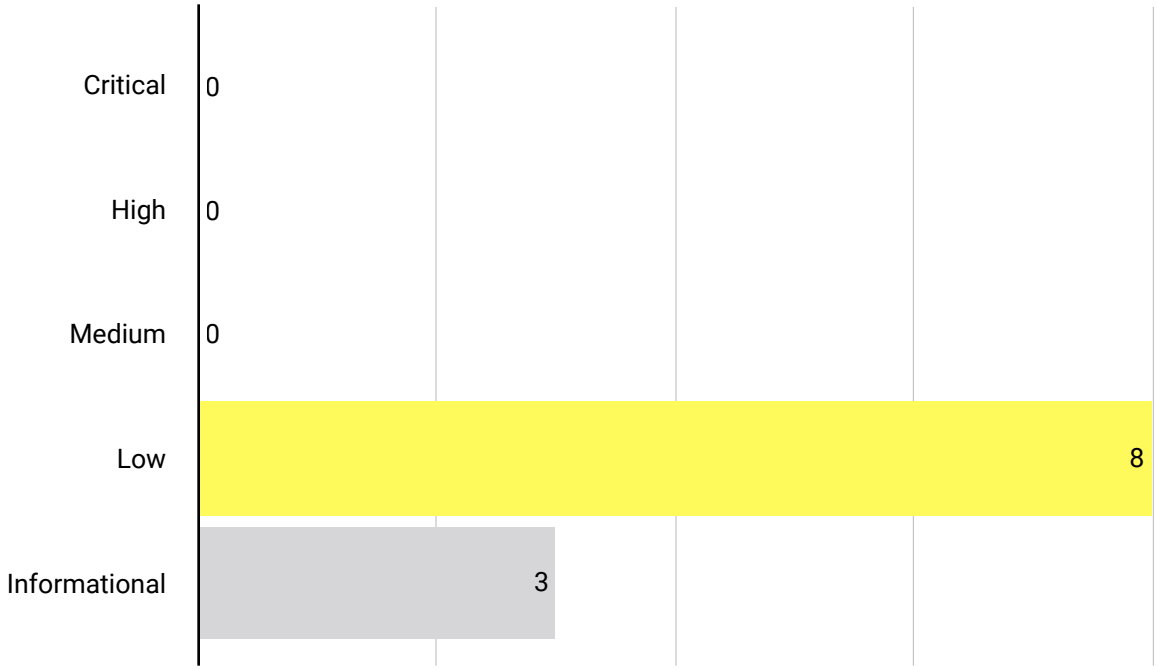
Findings Recap Table

ID	Title	Vulnerability Class	Severity	Status
TEL-Q122-1	Memory Exhaustion via Long Usernames	Denial of Service (DoS)	Low	Closed
TEL-Q122-2	Insecure LDAP Authentication Setup	Insecure Design	Low	Closed
TEL-Q122-3	Weak Access Policy Enforcement Between Teleport and Windows AD	Insecure Design	Informational	Risk Accepted
TEL-Q122-4	Bypass Teleport Login Username Restriction	Insufficient Authentication and Session Management	Low	Closed
TEL-Q122-5	Forced RDP Disconnection Abusing Direct Connection URLs	Insecure Design	Low	Closed
TEL-Q122-6	Missing Password File Permission Check	Information Exposure	Low	Closed
TEL-Q122-7	Teleport Virtual Smart Card Abuses	Insufficient Authentication and Session Management	Low	Risk Accepted
TEL-Q122-8	RDP Connections Remain Open Forever	Denial of Service (DoS)	Low	Closed
TEL-Q122-9	Privilege Escalation Risk Via The SeMachineAccountPrivilege Security Policy Setting	Insufficient Authentication and Session Management	Low	Closed

ID	Title	Vulnerability Class	Severity	Status
TEL-Q122-10	Enable Network Level Authentication (NLA)	Insecure Design	Informational	Risk Accepted
TEL-Q122-11	Fixed PIN Code For Smart Card	Insecure Design	Informational	Closed

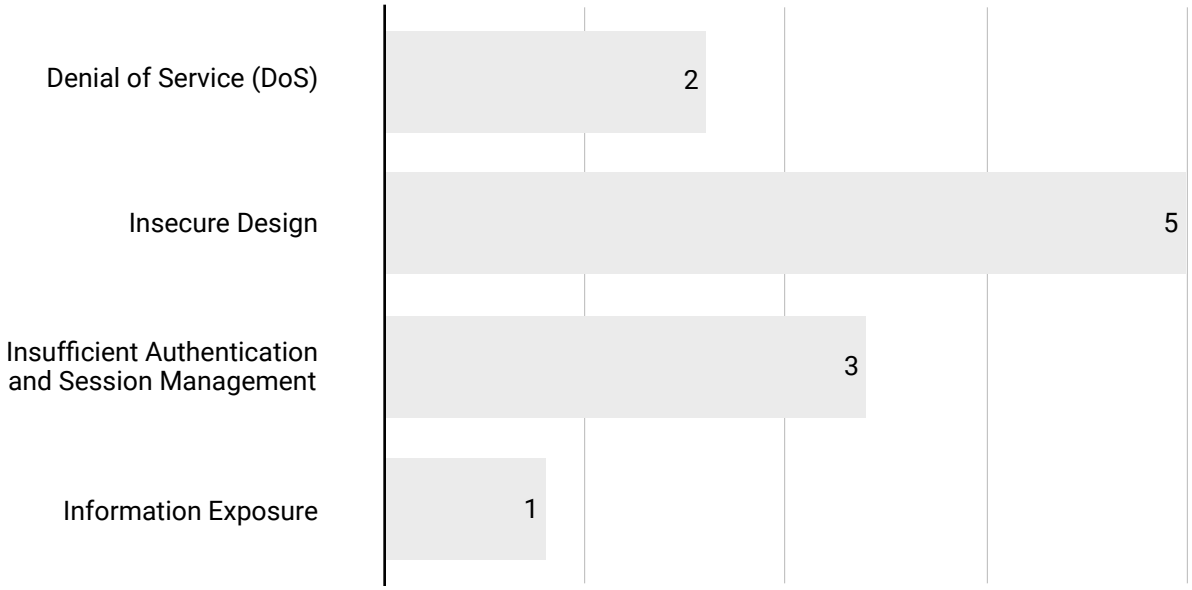
Findings per Severity

The table below provides a summary of the findings per severity.



Findings per Type

The table below provides a summary of the findings per vulnerability class.



TEL-Q122-1. Memory Exhaustion via Long Usernames	
Severity	**Low**
Vulnerability Class	Denial of Service (DoS)
Component	lib/srv/desktop/tdp/proto.go#369
Status	Closed

Description

In RFD 37, the client username message type is defined as:

```
| message type (7) | username_length uint32 | username []byte
```

The `decodeString` function in lib/srv/desktop/tdp/proto.go#369 makes a byte array with a size defined by `username_length`.

```
func decodeString(r io.Reader) (string, error) {
    var length uint32
    if err := binary.Read(r, binary.BigEndian, &length); err != nil {
        return "", trace.Wrap(err)
    }
    s := make([]byte, int(length))
    if _, err := io.ReadFull(r, s); err != nil {
        return "", trace.Wrap(err)
    }
    return string(s), nil
}
```

As this value is a `uint32`, the array can take up to 4 gigabytes of memory when `username_length` is `0xFFFFFFFF`.

Windows can support usernames up to 256 characters in length as defined by `UNLEN` in *lmcons.h*[1].

Reproduction Steps

The following steps can be taken to reproduce this issue:

1. Connect to a websocket with a valid bearer token.
2. Send the following client username websocket message:

```
07 ff ff ff ff 61
```

[1] https://github.com/microsoft/win32metadata/blob/b07213e28bcc48221c155024f9c5e0e1a92c6497/generation/WinSDK/RecompiledIdlHeaders/shared/lmcons.h#L91

3. The RAM usage of the Teleport process increases by approximately 4GB. Note that it is not required to actually send 4GB of username.

Impact

DoS via memory exhaustion. The attacker can make many concurrent websocket connections to increase the impact.

Complexity

An account on the Teleport web proxy is required in order to obtain a valid bearer token and open a websocket connection.

Remediation

Limit the string length allowed for client-provided values. Client username messages with `username_length` > 256 should return an error before the array memory is allocated. This issue may also affect the clipboard messages.

Retesting Results

During retesting, we observed that the issue has been fixed and the vulnerability can no longer be exploited.

Checking of the username length has been implemented and doesn't cause any memory issues anymore.

TEL-Q122-2. Insecure LDAP Authentication Setup	
Severity	**Low**
Vulnerability Class	Insecure Design
Component	https://goteleport.com/docs/desktop-access/getting-started/
Status	Closed

Description

Teleport requires a dedicated service account to connect to the customer's Active Directory domains. The documentation recommends to create this account by opening a PowerShell prompt and pasting in these copied commands:

```
$Name="Teleport Service Account"
$SamAccountName="svc-teleport"
$OutputFile="teleport-svc-pass.txt"

# Generate a random password that meets the "Password must meet complexity
requirements" security policy setting.
Add-Type -AssemblyName 'System.Web'
do {
   $Password=[System.Web.Security.Membership]::GeneratePassword(15,1)
} until ($Password -match '\d')
$SecureStringPassword=ConvertTo-SecureString $Password -AsPlainText -Force

# Save the plaintext password to a file for later use in your teleport.yaml.
$Password | Out-File $OutputFile

New-ADUser `
  -Name $Name `
  -SamAccountName $SamAccountName `
  -AccountPassword $SecureStringPassword `
  -Enabled $true
```

This script will call `generatepassword`[2] for the user and save it to a local text file. A warning about the need to delete this file from the Windows host is present in the documentation. This introduces a risk factor that could be lowered by using the following approaches:

- Only temporarily show on console or save to the clipboard the password instead of saving it locally
- Printing the warning on the same Powershell console to remind the user. An interactive prompt could let users quickly delete the file after the copying operations are completed (*"Press any key after the LDAP setup is completed to delete the file securely"*).
- Using certificated-based authentication instead of username and password authentication

[2] https://docs.microsoft.com/it-it/dotnet/api/system.web.security.membership.generatepassword?view=netframework-4.8

After generating the password file, either an encoding conversion from the Windows UTF-16LE to UTF-8 Unix or copying the characters manually, is needed. These issues may also indicate that displaying the password on the Powershell console and/or copying it to the clipboard would be the preferable option.

Reproduction Steps

N/A, this finding is a design improvement.

Impact

The password for the service account could be indefinitely stored in plaintext in a text file with a common name. Storing a password in this way may result in a system compromise.

Complexity

An attacker would still need access to the host containing the Teleport Service account password.

Remediation

- **Only show the password in the console or save it to the clipboard to lower the window of risk, or**
- **Print a warning on the same Powershell console reminding the user to delete the file**

Resources

- OWASP, "Password Plaintext Storage"
  https://owasp.org/www-community/vulnerabilities/Password_Plaintext_Storage

Retesting Results

During retesting, we observed that the issue has been fixed and the vulnerability can no longer be exploited.

The documentation has been updated. The getting-started guide no longer includes the insecure practice of saving sensitive information as plain text.

TEL-Q122-3. Weak Access Policy Enforcement Between Teleport and Windows AD	
Severity	**Informational**
Vulnerability Class	Insecure Design
Component	N/A
Status	Risk Accepted

Description

By design, Teleport Desktop Access allows users to filter the Windows hosts available using a list of DNS entries for any static hosts or an LDAP search filter (through the `teleport.yaml` configuration using `windows_desktop_service.ldap.discovery`).
In addition to this, in order to gain access to a remote desktop, a Teleport user needs to have the appropriate permissions for that desktop defined in a role they own, passing both its label (`windows_desktop_labels`) and the logins constraints (`windows_desktop_logins`), e.g.:

```
kind: role
version: v4
metadata:
  name: windows-desktop-admins
spec:
  allow:
    windows_desktop_labels:
      "*": "*"
    windows_desktop_logins: ["Administrator"]
```

Since Teleport is generating a generic smart card certificate for an LDAP user on the domain (lasting 6 minutes), as long as a user can access a `windows_desktop_logins` on any machine, it's possible for them to access any other machine on the domain, regardless of the `windows_desktop_labels` restriction specified in the role, or even the LDAP search filters specified in the parent `teleport.yaml` configuration. These logins would not be recorded in the event logs.

Reproduction Steps

Define a Teleport role restricting access only to a Windows label. Use the reproduction steps outlined in TEL-Q122-7 to bypass the rule.

Impact

An attacker could bypass the policy defined in their assigned roles or `teleport.yaml` configuration file and connect to other excluded Windows hosts in the same domain.

Complexity

Access to at least one user on an enrolled Windows machine is still needed to carry out an attack.

Remediation

The issue can be mitigated by clearly differentiating the role of Teleport vs the role of the sysadmin in protecting the Windows Desktop resources:

- **A warning should be provided to users setting up the solution in the Teleport online documentation that Teleport is not a replacement for a tailored GPO collection.**
- **Custom GPO security filtering should be applied to the appropriate OU, reflecting the configuration enforced by Teleport**

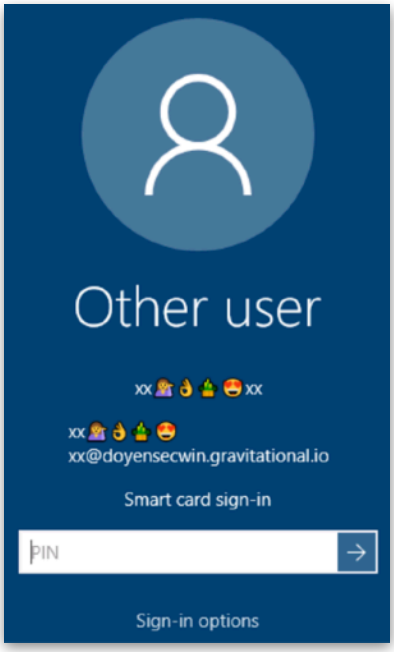
Retesting Results

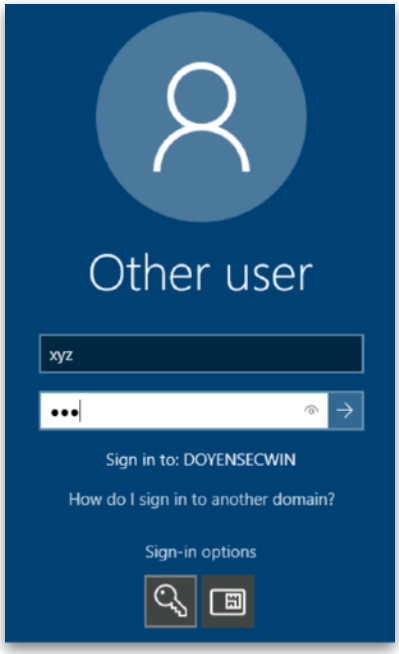
This issue can be only effectively remediated by Teleport's users. No documentation updates have been made.

TEL-Q122-4. Bypass Teleport Login Username Restriction	
Severity	**Low**
Vulnerability Class	Insufficient Authentication and Session Management
Component	RDP Authentication
Status	Closed

Description

Teleport controls which Windows desktop usernames a user can authenticate as. If there is a misconfiguration and a user is assigned Windows desktop usernames which do not exist on the target machine or domain, it is possible for the user to log in to a different Windows account once the first authentication fails. This will create a discrepancy in the Teleport event log where the original login username is recorded instead of the actual username.





Reproduction Steps

To login to another machine:

1. Add a non-existent Windows desktop login username to the Teleport configuration.
2. Restart Teleport
3. Start an RDP session as the non-existent user.

4. Login will fail, but the interactive login screen can be used over RDP. Select "Sign-in options" and then click on the key to enable username & password authentication.

Impact

The Teleport event log will show the original login username (the non-existent username) instead of the actual username.

Complexity

A valid password is required for the alternative user account.

Remediation

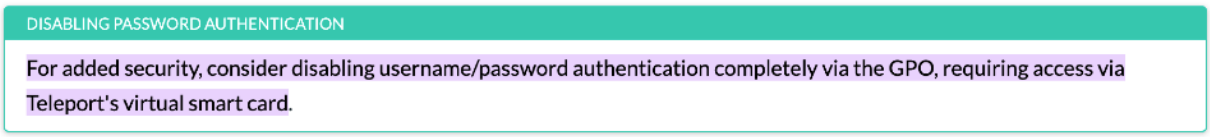
- **Recommend that customers set the "Interactive login: require smart card" Group Policy Option to Enabled.** This will disable username & password authentication.
- **Enable NLA (see TEL-Q122-10)**. The login screen will not be shown.

References

- Interactive logon: Require smart card - security policy setting https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/interactive-logon-require-smart-card

Retesting Results

During retesting, it has been observed that the issue is still exploitable in environments that do not enforce the use of smart card. Teleport has updated the documentation to recommend disabling username/password authentication via the GPO.



A Teleport user can pretend to login to the target machine with an arbitrary account by using the default Windows Login Screen as long as a correct credential is used. Teleport will log the Teleport login username associated with that session instead of the actual user that performed the login to the Windows OS. This is no longer possible once the smart card is required by the AD policy.

TEL-Q122-5. Forced RDP Disconnection Abusing Direct Connection URLs	
Severity	**Low**
Vulnerability Class	Insecure Design
Component	teleport/common/teleport.go
Status	Closed

Description

When initiating a web session from the Teleport UI, the user is redirected to a connection URL including the targeted cluster name, Windows host, and username, that will later be converted into a final session URL:

> https://doyensec-win.gravitational.io:3080/web/cluster/ip-172-31-6-85/desktops/EC2AMAZ-LRFTUVK-doyensecwin-gravitational-io/Administrator

After the URL is opened, a connection is automatically initiated to the targeted desktop. An attacker can abuse this to disconnect an authenticated user session for a particular Desktop by tricking a victim into initiating an attacker-controlled connection action.

This is because contrary to SSH sessions (where multiple consoles using the same user is allowed), on Windows when a new RDP connection occurs, the previous user is disconnected unless the *"Restrict each user to a single session"*[3] GPO is set to *Disable*.

Additionally, such an attack can't be logged, since it's undistinguishable from a normally opened session initiated from the Desktop list.

Reproduction Steps

The following steps can be taken to reproduce this issue:

1. Open a new connection to host A
2. Visit an external page with a link pointing to the same connection URL. Since *Samesite* cookies are not specifying the "`Strict`" setting, they will be sent to the Teleport web server.
3. If the *"Restrict each user to a single session"* GPO is not set to *Disable*, by default the session in the original tab will be disconnected.

[3] Run `gpedit.msc` and visit the path `Computer Configuration\Administrative Templates\Windows Components\Terminal Services\Terminal Server\Connections\`

Impact

An attacker having a controlled tab on the victim's browser could use this to prevent or hinder the ability of an Admin to apply remediation actions. This technique could be used as a secondary step during a larger attack.

Complexity

An attacker first needs a way to control a page opened by an authenticated victim.

Remediation

After a new connection URL is opened, first check that the user doesn't already have an opened session in another tab. If so, prompt a dialog to warn the user of the existence of the other session, letting them decide if they wish to continue.

Retesting Results

During retesting, we observed that the issue has been fixed and the vulnerability can no longer be exploited.

During the connection to the target (with a live session), the user is getting a warning about the potential risk and can decide whether they wish to continue.

TEL-Q122-6. Missing Password File Permission Check	
Severity	**Low**
Vulnerability Class	Information Exposure
Component	lib/config/configuration.go
Status	Closed

Description

In Teleport, the plaintext password for the LDAP authentication is fetched by reading a file (`password_file`) specified in the `teleport.yaml` configuration. In the event in which this file is stored with insecure permissions, no warning or panic is returned by the Teleport daemon (`lib/config/configuration.go:1255`):

```
// applyWindowsDesktopConfig applies file configuration for the
"windows_desktop_service" section.
func applyWindowsDesktopConfig(fc *FileConfig, cfg *service.Config) error {
 ...
 ldapPassword, err := os.ReadFile(fc.WindowsDesktop.LDAP.PasswordFile)
 if err != nil {
        return trace.WrapWithMessage(err, "loading the LDAP password from file
%v",
                fc.WindowsDesktop.LDAP.PasswordFile)
 }
```

As an additional defense, this check should occur in case the LDAP password file is world readable (i.e., is set to anything above `X44` instead of `400` or `600`). The following table summarizes the suggested panic or warning conditions:

Permission	Owner	Group	Public
Read	PASS	FAIL	FAIL
Write	PASS	FAIL	FAIL
Execute	PASS	FAIL	FAIL

Reproduction Steps

Set the `password_file` permissions to `777`. Observe that the Teleport daemon does not raise any warnings in its logs.

Impact

Medium, if permissions are not checked during the setup, an attacker could more easily read the configuration password.

Complexity

Medium, an attacker still needs access to the local system to read a password file having insecure permissions.

Remediation

Check the permissions of the configuration file to only allow reading from the file by the superuser. The correct permission would be `600` or `400` in this case. This would allow the owner to read/write and gives no permissions to other users or groups.

Retesting Results

The Teleport team migrated from the password_file design in favor of client certificate authentication. Because of this, the finding is considered Closed.

TEL-Q122-7. Teleport Virtual Smart Card Abuses	
Severity	**Low**
Vulnerability Class	Insufficient Authentication and Session Management
Component	teleport/lib/srv/desktop/rdp/rdpclient/src/scard.rs
Status	Risk Accepted

Description

Teleport integrates a virtual smart card using an RDPDR (device redirection) static channel. This smart card is available to the system indefinitely after login.

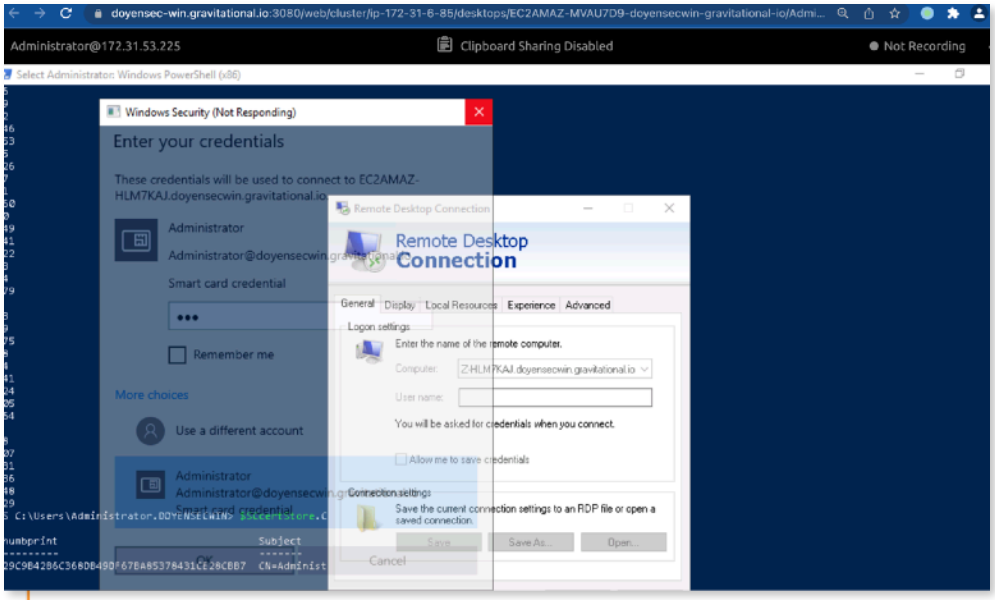
Teleport limits which Windows desktops a Teleport user can connect to. Since the smart card is connected to the Windows machine via RDP, the smart card credentials can be relayed to make further connections to restricted hosts. The smart card can also be used to encrypt, decrypt and sign arbitrary data, although this should not have any security impact.

The smart card certificate identifies a specific domain user, but is not restricted to a specific machine.

Reproduction Steps

To login to another machine:

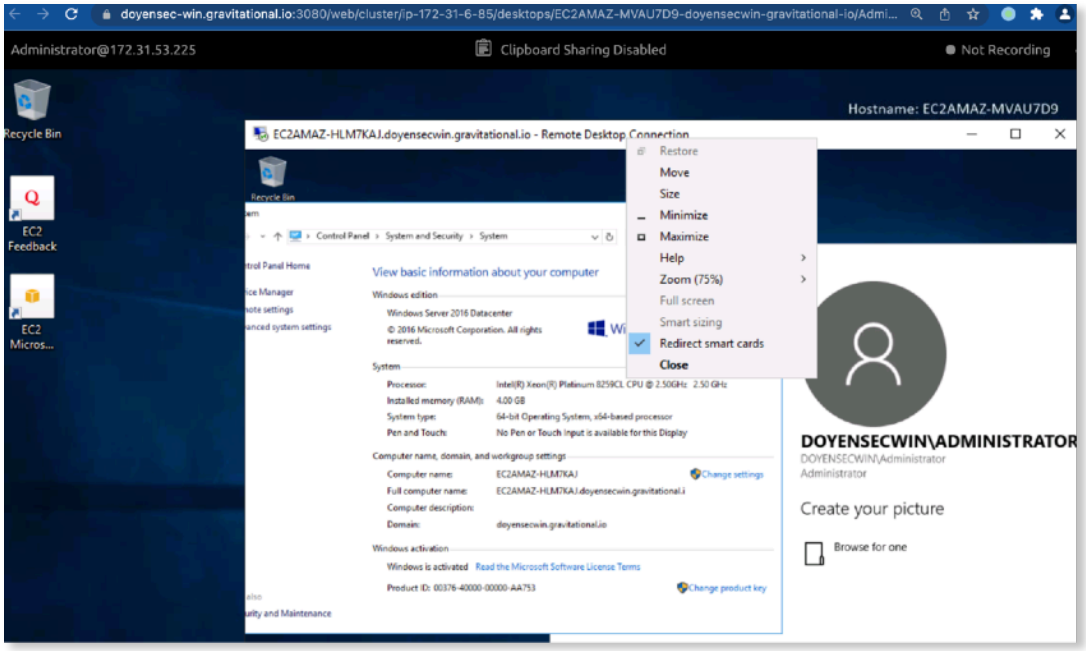
1. Use Teleport to login to a Windows desktop.
2. Open Remote Desktop Connection and attempt to connect to a different machine with your current user name.
3. Use the smart card credential to authenticate. The credentials dialog box may freeze for some (3-4) minutes.



4. You will be connected to the target machine.To inspect the smart card via Powershell, run `certutil -scinfo`.

Impact

Users connected to an allowed Windows host, can make further connections to restricted hosts in violation of the Teleport configuration. The login will not be tracked by Teleport. The smart card can only be used to authenticate as the same domain user.



Complexity

Smart card certificates have a validity period of 6 minutes.

Remediation

Possible solutions include:

- **Instruct customers to restrict inbound RDP connections to those from their Teleport host.**
- **Reduce the validity duration of the smart card certificate to seconds, or the limit the amount of times the smart card emulator can reply to a challenge.**
- **Set a random smart card PIN for every RDP session**

Resources

- Sysadmins LV, “How to enumerate all certificates on a smart card (PowerShell)” https://www.sysadmins.lv/retired-msft-blogs/alejacma/how-to-enumerate-all-certificates-on-a-smart-card-powershell.aspx

Retesting Results

During the retest, the issue was not fully addressed. While the PIN now randomly changes on a per-session basis and the documentation better explains the feature, some underlying risks still exist:

- Even if the PIN is dynamic, the virtual smart card does not respond by employing an anti-hammering logic, which rejects further attempts for a period of time or by blocking the card. This is also known as lockout
- The PIN code is not compared with a time-safe comparison operator, potentially leaking its value
- As a design decision, any smart card hardware commands will still be forwarded to the client (`windows_desktop_service` in this case). If a user opens another RDP client within their remote desktop, they should just be able to log into any host in the same domain without a password, even if the Teleport RDP configuration restricts this

TEL-Q122-8. RDP Connections Remain Open Forever	
Severity	**Low**
Vulnerability Class	Denial of Service (DoS)
Component	rdp-rs
Status	Closed

Description

When Teleport connects to a well-behaved RDP server, closing the websocket connection causes the RDP connection to be closed. However, it is possible for a misbehaving RDP server to keep the TCP connection between Teleport and the RDP server open indefinitely.

In *rdp-rs*[4], there are several places where execution blocks waiting for a fixed number of bytes from the RDP server, e.g. tpkt.rs#121:

```
let mut buffer = Cursor::new(self.transport.read(2)?);
```

There is no timeout or cancellation on the underlying `Read.read_exact` function. Therefore, if the RDP server does not send 2 bytes, the connection will remain open and execution will be paused inside the `read` function forever.

Reproduction Steps

The following steps can be taken to reproduce this issue:

1. Add a Linux machine as a Windows desktop in the Teleport configuration.
2. On the Linux machine, start a *netcat* or *socat* listener on port `3389`, e.g.,

   ```
   socat -v TCP4-LISTEN:3389,fork,reuseaddr -
   ```

3. From Teleport web, open multiple Teleport RDP sessions to the Linux machine.
4. Close the RDP connection browser tabs to close the websocket connections.
5. Wait several minutes and then use *netstat* on the Teleport server to confirm that there are still open TCP connections between Teleport and port `3389` on the Linux machine.
6. Close the connections from the Linux machine by killing the *socat* listener.

[4] https://github.com/gravitational/rdp-rs

7. The Teleport log will show the Rust error:

```
Rdp(Io(Error { kind: UnexpectedEof, message: "failed to fill whole buffer" }))
```

Impact

Multiple TCP connections can be held open forever, potentially causing a Denial of Service.

Complexity

The misbehaving RDP server must be accessible from the Teleport configuration.

Remediation

Kill hanging RDP connections. When the websocket is closed, the RDP connection should also be closed. This could, for instance, be done with timeouts on socket reads.

Resources

- Rust Team, “TcpStream.set_read_timeout” https://doc.rust-lang.org/std/net/struct.TcpStream.html#method.set_read_timeout

Retesting Results

The TCP socket is now closed from the Go side. A pull request responsible for this was merged on July 28th (https://github.com/gravitational/teleport/pull/14703).

TEL-Q122-9. Privilege Escalation Risk Via The SeMachineAccountPrivilege Security Policy Setting	
Severity	**Low**
Vulnerability Class	Insufficient Authentication and Session Management
Component	https://goteleport.com/docs/desktop-access/getting-started/
Status	Closed

Description

By default, the Group Policy Object (GPO) configuration for the Default Domain Controller Policy grants the "Add workstations to domain user" right to all authenticated users[5]. Users with this right could add a device to the domain that will be listed in the Teleport list of Desktops.
By combining findings TEL-Q122-3, TEL-Q122-5 and TEL-Q122-7, it is possible for an attacker to:

1. Enroll a new controlled Windows Desktop to the domain
   i. Start a Teleport Desktop Access session to an existing machine.
   ii. RDP into the attacker-controlled machine.
   iii. Register the machine on the domain using the relayed smart card for domain authentication.
2. Wait for the Desktop to be registered by the Teleport synchronization
3. Use the direct connection URL to trick an authenticated user into opening a new connection to their controlled machine. The attacker can then:
   - Relay the smart card login
   - Use the Teleport-emulated smart card to access other resources

Since Teleport automatically lists the new computers added to the customer's Active Directory domain, Teleport should suggest to not universally grant this level of access.

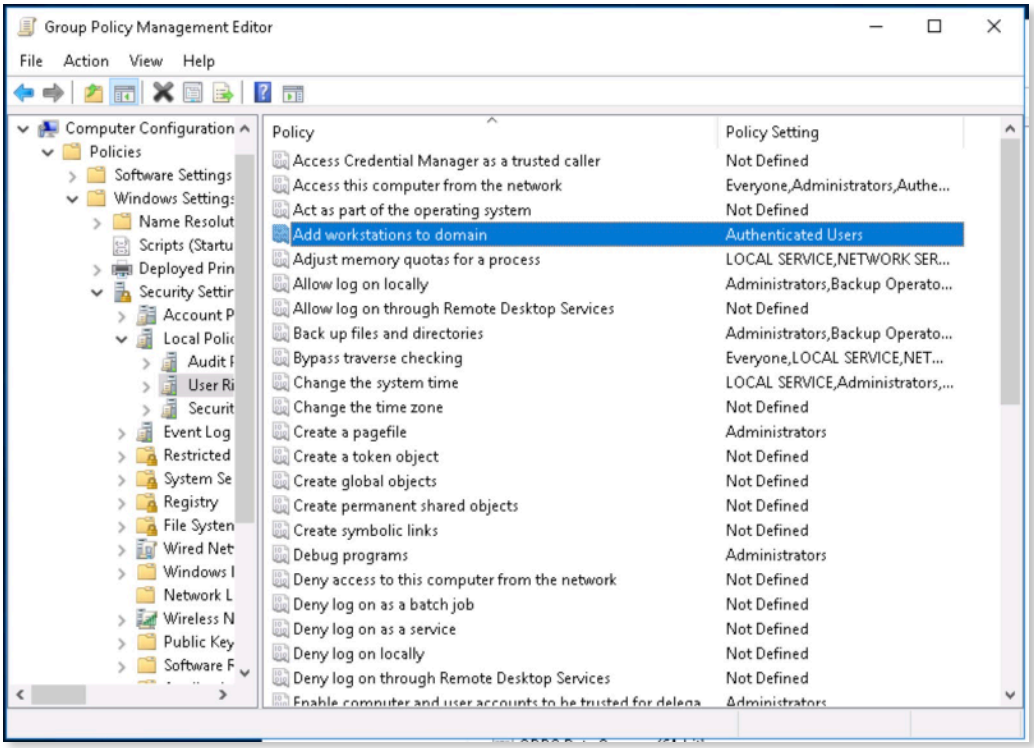
Reproduction Steps

To check the default configuration of the Default Domain Controller Policy GPO:

1. Using an RDP client, log in to a machine that has administrative access to your Active Directory domain.
2. Open the Group Policy Management Console (GPMC).
3. Go to Forest > Domains > domain-name > Group Policy Objects, where domain-name is the name of your Active Directory domain.
4. Right-click Default Domain Controller Policy and click Edit.
5. In the Group Policy Management Editor console, go to Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > User Rights Assignment.
6. Double-click Add workstations to domain.

[5] https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/add-workstations-to-domain

7. In Properties, observe that Authenticated Users (S-1-5-11) is in the list. If any group SIDs other than S-1-5-32-544 (Administrators) is present, the domain is at risk.



Impact

The "Add workstations to domain user" right presents a moderate risk for Teleport, since users with this right could add a malicious Desktop device to the domain that can be programmed to exploit the Teleport-provided virtual smart card and move to other machines with other users' privileges.

Complexity

High, an attacker would still need to authenticate to the domain in order to mount this attack. A malicious Windows or RDP Server version would also be required to relay the smart card from the malicious host without the user approval.

Remediation

The Teleport documentation should advise to configure this setting so that only authorized members of the IT team are allowed to add computers to the domain.

Retesting Results

During retesting, we observed that the issue has been fixed and the vulnerability can no longer be exploited.

It has been discovered that the documentation has been updated and now it is advising to configure this setting so that only authorized members of the IT team are allowed to add computers to the domain.

TEL-Q122-10. Enable Network Level Authentication (NLA)	
Severity	**Informational**
Vulnerability Class	Insecure Design
Component	https://github.com/gravitational/rdp-rs/blob/cb61119d2803f647b60e6c9b2ef05ab587cc1966/src/nla/cssp.rs#L135
Status	Risk Accepted

Description

Teleport currently requires that Network Level Authentication (NLA)[6] is disabled on the target Windows hosts. Enabling NLA can help Teleport customers to avoid Denial of Service (DoS) attacks (continuously loading the login screen from the server for the user, shutting down the machine), as well as remote code execution attacks exploiting the expanded attack surface (e.g., *BlueKeep*). Moreover, the IP addresses of the clients trying to log in will not be stored in the security audit logs, making it harder to block brute force or dictionary attacks by means of a firewall.

The *rdp-rs* library in use by Teleport is currently using a Credential Security Support Provider (CredSSP) Protocol implementation to only create a `TSPasswordCreds` structure[7], while currently missing the `TSSmartCardCreds`[8,9] and its `TSCSPDataDetail`[10].

The `TSSmartCardCreds` structure contains the user's smart card credentials that are delegated to the server:

```
TSSmartCardCreds ::= SEQUENCE {
        pin         [0] OCTET STRING,
        cspData     [1] TSCspDataDetail,
        userHint    [2] OCTET STRING OPTIONAL,
        domainHint  [3] OCTET STRING OPTIONAL
}
```

The `TSCSPDataDetail` structure contains information about the cryptographic service provider (CSP) used during smart card logon:

```
TSCspDataDetail ::= SEQUENCE {
        keySpec       [0] INTEGER,
```

[6] https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-allow-access#why-allow-connections-only-with-network-level-authentication

[7] https://github.com/gravitational/rdp-rs/blob/master/src/nla/cssp.rs#L136

[8] https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cssp/4251d165-cf01-4513-a5d8-39ee4a98b7a4?redirectedfrom=MSDN

[9] https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-CSSP/%5bMS-CSSP%5d.pdf

[10] https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cssp/34ee27b3-5791-43bb-9201-076054b58123

```
        cardName      [1] OCTET STRING OPTIONAL,
        readerName    [2] OCTET STRING OPTIONAL,
        containerName [3] OCTET STRING OPTIONAL,
        cspName       [4] OCTET STRING OPTIONAL
}
```

More specifically, the lack of a dedicated smart card initialization in the `TSCredentials`[11] structure of *rdp-rs* can be identified. This structure is used to set both the user's credentials that are delegated to the server and their type:

```
TSCredentials ::= SEQUENCE {
        credType    [0] INTEGER,
        credentials [1] OCTET STRING
}
```

The `create_ts_credentials` (:135) function in `gravitational/rdp-rs/blob/master/src/nla/cssp.rs` is responsible for creating this structure:

```
fn create_ts_credentials(domain: Vec<u8>, user: Vec<u8>, password: Vec<u8>) ->
Vec<u8> {
    let ts_password_creds = sequence![
        "domainName" => ExplicitTag::new(Tag::context(0), domain as OctetString),
        "userName" => ExplicitTag::new(Tag::context(1), user as OctetString),
        "password" => ExplicitTag::new(Tag::context(2), password as OctetString)
    ];

    let ts_password_cred_encoded = yasna::construct_der(|writer| {
        ts_password_creds.write_asn1(writer).unwrap();
    });

    let ts_credentials = sequence![
        "credType" => ExplicitTag::new(Tag::context(0), 1 as Integer),
        "credentials" => ExplicitTag::new(Tag::context(1),
ts_password_cred_encoded as OctetString)
    ];

    to_der(&ts_credentials)
}
```

Here the `credType` is hardcoded to `1`, meaning that the credentials will always contain a `TSPasswordCreds` structure (section 2.2.1.2.1) that defines the user's password credentials.

As a reference, example implementations (in Twisted Python and C) on how to implement CredSSP smart card auth are available in *RDPy*[12] or *rDesktop*[13] . For comparison, the `cssp_encode_tscredentials` (:448) of `rdesktop/rdesktop/blob/master/cssp.c` is correctly setting the `credType` and encoding the `TSSmartCardCreds` structure.

[11] https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cssp/94a1ab00-5500-42fd-8d3d-7a84e6c2cf03

[12] https://github.com/citronneur/rdpy/blob/master/rdpy/protocol/rdp/nla/cssp.py#L95

[13] https://github.com/rdesktop/rdesktop/commit/d1e8fdc90a905576462815c3a5aa5f1263b8baea

Reproduction Steps

N/A, the finding is a hardening suggestion.

Impact

Medium, since several risks are associated with this functionality. An attacker could:

- Accurately fingerprint the version of Windows
- Potentially identify user accounts on the system
- Leverage the RDP service to consume excessive system resource
- Exploit wormable vulnerabilities in Windows Terminal Services

Complexity

Medium, since an attacker still needs to bypass the Teleport smart card-based authentication. Using NLA is only the first step towards achieving a hardened Remote Desktop Server.

Remediation

Enable NLA with smart card authentication. This provides an added layer of security by ensuring that RDP sessions are only initiated from the Teleport Windows desktop service.

Retesting Results

During retesting, we observed that the issue is still present. Nonetheless, the Teleport team accepted this risk for the time being. Teleport still requires relying on disabling Network Level Authentication (NLA) in order to work.

TEL-Q122-11. Fixed PIN Code For Smart Card	
Severity	**Informational**
Vulnerability Class	Insecure Design
Component	teleport/lib/srv/desktop/rdp/rdpclient/src/piv.rs
Status	Closed

Description

Teleport integrates a virtual smart card using an RDPDR (device redirection) static channel. This smart card is available to the system indefinitely after login.

When performing the auto-login, Teleport sends a hardcoded PIN to Windows as seen in `lib.rs#190`:

```
    // Password must be non-empty for autologin
(sec::InfoFlag::InfoPasswordIsScPin) to trigger on
    // a smartcard.
    let password = "123".to_string();
```

Windows sends the PIN back to the smart card interface in a `Verify` command:

```
Command { class: Class { cla: 0, range: Interindustry(First) }, instruction:
Verify, p1: 0, p2: 128, data: [49, 50, 51, 255, 255, 255, 255, 255], le: 0,
extended: false }
```

where 49, 50, 51 is the decimal ASCII for "`123`".

In `piv.rs#138`, PIN verification is not actually performed in response to the Verify command.

```
fn handle_verify(&mut self, _cmd: Command<S>) -> RdpResult<Response> {
    // No PIN verification needed.
    Ok(Response::new(Status::Success))
}
```

In addition, the cryptographic functions of the smart card are not locked awaiting PIN verification.

The smart card PIN is important to prevent use of stolen physical smart cards which is not an issue for Teleport. However, applications running on the Windows host may access the smart card to authenticate to other machines. Requiring a randomly-generated PIN which is only known to Teleport would prevent use of the smart card for anything besides the initial Windows login.

Reproduction Steps

This is a source code finding. Examples of reuse of the smart card credential are provided in TEL-Q122-7.

Impact

Malicious applications running on the Desktop could use the smart card cryptographic material without user interaction.

Complexity

A good understanding of Windows APIs and smart card devices on Windows is required to write applications which silently use the unlocked smart card. Using the smart card to make RDP connections to additional hosts as described in TEL-Q122-7 is trivial.

Remediation

Require a PIN to access the smart card.

Prevent use of the cryptographic functions for a Handle until the Handle is unlocked. On initial connections, the RDP client should generate a cryptographically secure random 8-digit PIN code which is sent during the initial connection for use by auto-login. The PIN verification function should check the PIN before unlocking access to the smart card.

Retesting Results

During retesting, we observed that the issue has been fixed and the vulnerability can no longer be exploited.

The affected source code has been updated and it doesn't rely on an hardcoded value.

Appendix A - Vulnerability Classification

Vulnerability Severity	**Critical**
	High
	Medium
	Low
	Informational
Vulnerability Class	Components With Known Vulnerabilities
	Covert Channel (Timing Attacks, etc.)
	Cross Site Request Forgery (CSRF)
	Cross Site Scripting (XSS)
	Denial of Service (DoS)
	Information Exposure
	Injection Flaws (SQL, XML, Command, Path, etc)
	Insecure Design
	Insecure Direct Object References (IDOR)
	Insufficient Authentication and Session Management
	Insufficient Authorization
	Insufficient Cryptography
	Memory Corruption (Buffer and Integer Overflows, Format String, etc)
	Race Condition
	Security Misconfiguration
	Server-Side Request Forgery (SSRF)
	Unrestricted File Uploads
	Unvalidated Redirects and Forwards
	User Privacy
	Time-of-Check to Time-of-Use (TOCTOU)
	Insecure Deserialization

Appendix B - Remediation Checklist

The table below can be used to keep track of your remediation efforts inside this report. Mark the boxes when a fix has been implemented for the vulnerability.

☒	**Limit the string length allowed for client-provided values**
☒	• **Only show the password in the console or save it to the clipboard to lower the window of risk, or** • **Print a warning on the same Powershell console reminding the user to delete the file**
☐	**The issue can be mitigated by clearly differentiating the role of Teleport vs the role of the sysadmin in protecting the Windows Desktop resources:** • **A warning should be provided to users setting up the solution in the Teleport online documentation that Teleport is not a replacement for a tailored GPO collection.** • **Custom GPO security filtering should be applied to the appropriate OU, reflecting the configuration enforced by Teleport**
☒	• **Recommend that customers set the "Interactive login: require smart card" Group Policy Option to Enabled.** This will disable username & password authentication. • **Enable NLA (see TEL-Q122-10)**. The login screen will not be shown.
☒	**After a new connection URL is opened, first check that the user doesn't already have an opened session in another tab. If so, prompt a dialog to warn the user of the existence of the other session, letting them decide if they wish to continue**
☒	**Check the permissions of the configuration file to only allow reading from the file by the superuser**
☐	**Possible solutions include:** • **Instruct customers to restrict inbound RDP connections to those from their Teleport host** • **Reduce the validity duration of the smart card certificate to seconds, or the limit the amount of times the smart card emulator can reply to a challenge**
☐	**Kill hanging RDP connections**
☒	**The Teleport documentation should advise to configure this setting so that only authorized members of the IT team are allowed to add computers to the domain**
☐	**Enable NLA with smart card authentication. This provides an added layer of security by ensuring that RDP sessions are only initiated from the Teleport windows desktop service**
☒	**Require a PIN to access the smart card**

When done patching the listed vulnerabilities, many clients find it worthwhile to perform a retest. During a retest, Doyensec researchers will attempt to bypass and subvert all implemented fixes. Retests usually take one or two days. Please reach out if you'd like more information on our retesting process.

Appendix C - RDP Fuzzing

This appendix explains in detail the fuzzing process we performed during the first phase (A) of the engagement. Due to Go-specific components' limited attack surface, we entirely focused our fuzzing on the Remote Desktop Protocol implemented in Rust. In detail, we performed the fuzzing on the following repository parts:

- https://github.com/gravitational/teleport/tree/master/lib/srv/desktop/rdp/rdpclient/src (rev `b95fb530bb7d3ad356d7dce3ec115045db5c6fa8`, tag: 8.0.1)
- https://github.com/gravitational/rdp-rs (rev `755e950dcff0fc6965aa518c4596b995ede3417d`)

<OMITTED FOR PUBLICATION>