January 10, 2022 • 10 min read

New macOS vulnerability, “powerdir,” could lead to unauthorized user data access

Microsoft 365 Defender Research Team

Share

Following our discovery of the “Shrootless” vulnerability, Microsoft uncovered a new macOS vulnerability, “powerdir,” that could allow an attacker to bypass the operating system’s Transparency, Consent, and Control (TCC) technology, thereby gaining unauthorized access to a user’s protected data. We shared our findings with Apple through Coordinated Vulnerability Disclosure (CVD) via Microsoft Security Vulnerability Research (MSVR). Apple released a fix for this vulnerability, now identified as CVE-2021-30970, as part of security updates released on December 13, 2021. We encourage macOS users to apply these security updates as soon as possible.

Introduced by Apple in 2012 on macOS Mountain Lion, TCC is essentially designed to help users configure the privacy settings of their apps, such as access to the device’s camera, microphone, or location, as well as access to the user’s calendar or iCloud account, among others. To protect TCC, Apple introduced a feature that prevents unauthorized code execution and enforced a policy that restricts access to TCC to only apps with full disk access. We discovered that it is possible to programmatically change a target user’s home directory and plant a fake TCC database, which stores the consent history of app requests. If exploited on unpatched systems, this vulnerability could allow a malicious actor to potentially orchestrate an attack based on the user’s protected personal data. For example, the attacker could hijack an app installed on the device—or install their own malicious app—and access the microphone to record private conversations or capture screenshots of sensitive information displayed on the user’s screen.

It should be noted that other TCC vulnerabilities were previously reported and subsequently patched before our discovery. It was also through our examination of one of the latest fixes that we came across this bug. In fact, during this research, we had to update our proof-of-concept (POC) exploit because the initial version no longer worked on the latest macOS version, Monterey. This shows that even as macOS or other operating systems and applications become more hardened with each release, software vendors like Apple, security researchers, and the larger security community, need to continuously work together to identify and fix vulnerabilities before attackers can take advantage of them.

Microsoft security researchers continue to monitor the threat landscape to discover new vulnerabilities and attacker techniques that could affect macOS and other non-Windows devices. The discoveries and insights from our research enrich our protection technologies and solutions, such as Microsoft Defender for Endpoint, which allows organizations to gain visibility to their networks that are increasingly becoming heterogeneous. For example, this research informed the generic detection of behavior associated with this vulnerability, enabling Defender for Endpoint to immediately provide visibility and protection against exploits even before the patch is applied. Such visibility also enables organizations to detect, manage, respond to, and remediate vulnerabilities and cross-platform threats faster.

In this blog post, we will share some information about TCC, discuss previously reported vulnerabilities, and present our own unique findings.

TCC overview

As mentioned earlier, TCC is a technology that prevents apps from accessing users' personal information without their prior consent and knowledge. The user commonly manages it under System Preferences in macOS (System Preferences > Security & Privacy > Privacy):

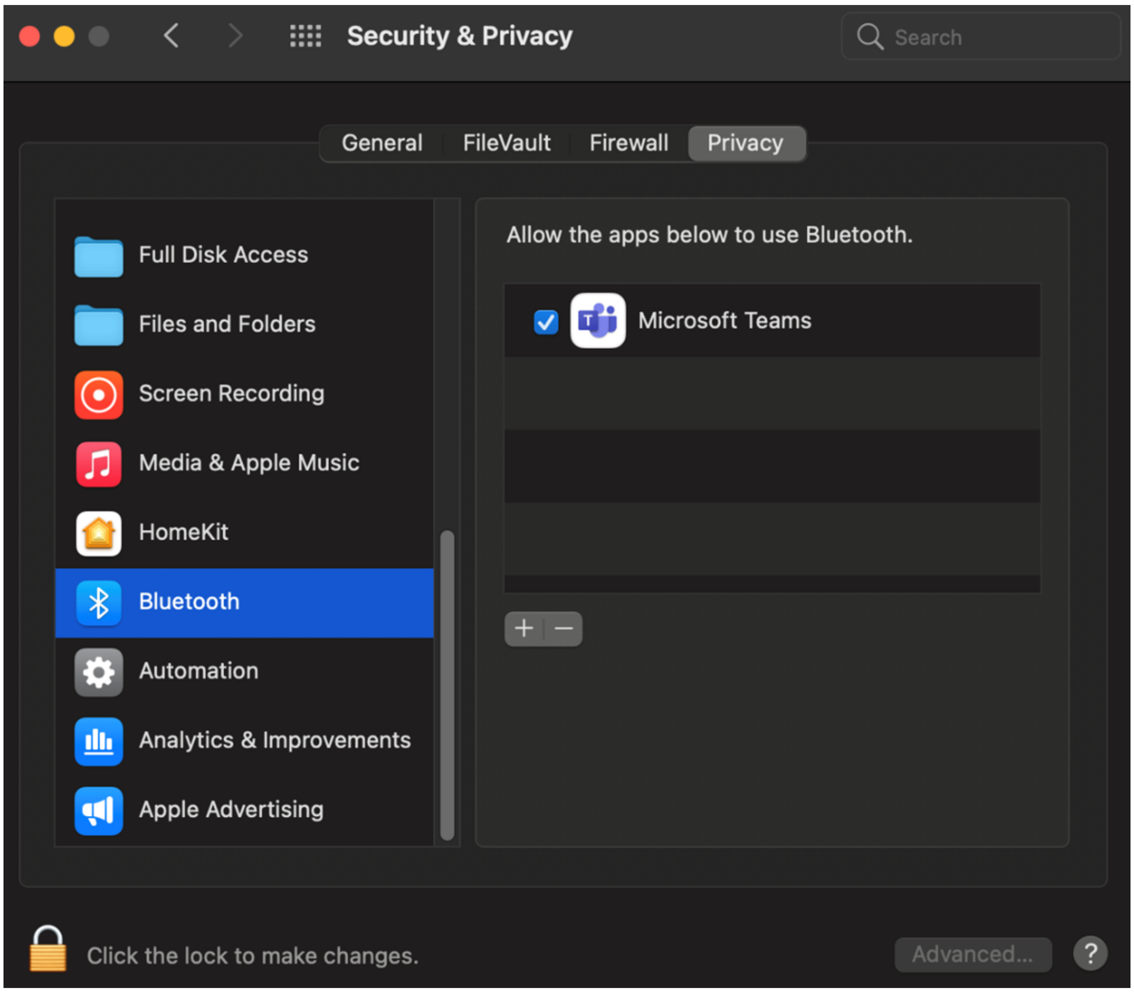


Figure 1. The macOS Security & Privacy pane that serves as the front end of TCC.

TCC maintains databases that contain consent history for app requests. Generally, when an app requests access to protected user data, one of two things can happen:

1. If the app and the type of request have a record in the TCC databases, then a flag in the database

entry dictates whether to allow or deny the request without automatically and without any user interaction.

2. If the app and the type of request do not have a record in the TCC databases, then a prompt is presented to the user, who decides whether to grant or deny access. The said decision is backed into the databases so that succeeding similar requests will now fall under the first scenario.

Under the hood, there are two kinds of TCC databases. Each kind maintains only a subset of the request types:

- **User-specific database:** contains stored permission types that only apply to the specific user profile; it is saved under *~/Library/Application Support/com.apple.TCC/TCC.db* and can be accessed by the user who owns the said profile
- **System-wide database:** contains stored permission types that apply on a system level; it is saved under */Library/Application Support/com.apple.TCC/TCC.db* and can be accessed by users with root or full disk access

macOS implements the TCC logic by using a special daemon called *tccd*. Indeed, there are at least two instances of tccd: one run by the user and the other by root.

Screenshot of two tccd instances

Figure 2. Two tccd instances: per-user and system-wide.

Each type of request starts with a *kTCCService* prefix. While not an exhaustive list, below are some examples:

Request type	Description	Handled by
kTCCServiceLiverpool	Location services access	User-specific TCC database
kTCCServiceUbiquity	iCloud access	User-specific TCC database
kTCCServiceSystemPolicyDesktopFolder	Desktop folder access	User-specific TCC database
kTCCServiceCalendar	Calendar access	User-specific TCC database
kTCCServiceReminders	Access to reminders	User-specific TCC database
kTCCServiceMicrophone	Microphone access	User-specific TCC database
kTCCServiceCamera	Camera access	User-specific TCC database
kTCCServiceSystemPolicyAllFiles	Full disk access capabilities	System-wide TCC database
kTCCServiceScreenCapture	Screen capture capabilities	System-wide TCC database

Table 1. Types of TCC requests.

It should also be noted that the *TCC.db* file is a SQLITE database, so if a full disk access is granted to a user, they can view the database and even edit it:

Screenshot of TCC.db access table dump

Figure 3. Dumping the TCC.db access table, given a full disk access.

The database columns are self-explanatory, save for the *csreq* column. The *csreq* values contain a hexadecimal blob that encodes the code signing requirements for the app. These values can be calculated easily with the *codesign* and csreq utilities, as seen in Figure 4 below:

Screenshot of building the csreq blob

Figure 4. Building the csreq blob manually for an arbitrary app.

Given these, should a malicious actor gain full disk access to the TCC databases, they could edit it to grant arbitrary permissions to any app they choose, including their own malicious app. The affected user would also not be prompted to allow or deny the said permissions, thus allowing the app to run with configurations they may not have known or consented to.

Securing (and bypassing) TCC: Techniques and previously reported vulnerabilities

Previously, apps could access the TCC databases directly to view and even modify their contents. Given the risk of bypass mentioned earlier, Apple made two changes. First, Apple protected the system-wide *TCC.db* via System Integrity Protection (SIP), a macOS feature that prevents unauthorized code execution. Secondly, Apple enforced a TCC policy that only apps with full disk access can access the *TCC.db* files. Note, though, that this policy was also subsequently abused as some apps required such access to function properly (for example, the SSH daemon, sshd).

Interestingly, attackers can still find out whether a user’s Terminal has full disk access by simply trying to list the files under */Library/Application Support/com.apple.TCC*. A successful attempt means that the Terminal has full disk access capabilities, and an attacker can, therefore, freely modify the user’s *TCC.db*.

In addition, there have been several previously reported vulnerabilities related to TCC bypass. These include the following:

- **Time Machine mounts** (CVE-2020-9771): macOS offers a built-in backup and restore solution called Time Machine. It was discovered that Time Machine backups could be mounted (using the *apfs_mount* utility) with the "*noowners*" flag. Since these backups contain the *TCC.db* files, an attacker could mount those backups and determine the device's TCC policy without having full disk access.
- **Environment variable poisoning** (CVE-2020-9934): It was discovered that the user's *tccd* could build the path to the *TCC.db* file by expanding *$HOME/Library/Application Support/com.apple.TCC/TCC.db*. Since the user could manipulate the *$HOME* environment variable (as introduced to *tccd* by *launchd*), an attacker could plant a chosen *TCC.db* file in an arbitrary path, poison the *$HOME* environment variable, and make *TCC.db* consume that file instead.
- **Bundle conclusion issue** (CVE-2021-30713): First disclosed by Jamf in a blog post about the XCSSET malware family, this bug abused how macOS was deducing app bundle information. For example, suppose an attacker knows of a specific app that commonly has microphone access. In that case, they could plant their application code in the target app's bundle and "inherit" its TCC capabilities.

Apple has since patched these vulnerabilities. However, based on our research, the potential bypass to *TCC.db* can still occur. The following section discusses the vulnerability we discovered and some details about the POC exploits we developed to prove the said vulnerability.

Modifying the home directory: The 'powerdir' vulnerability

In assessing the previous TCC vulnerabilities, we evaluated how Apple fixed each issue. One fix that caught our attention was for CVE-2020-9934 (the *$HOME* environment variable poisoning vulnerability). The fix can be seen in the *_db_open* function in *tccd*:

Screenshot of the tccd fix for CVE-2020-9934

Figure 5. The tccd fix for CVE-2020-9934.

We noted that instead of expanding the *$HOME* environment variable, Apple decided to invoke getpwuid() on the current user (retrieved with getuid()). First, the *getpwuid* function retrieves a structure in memory (*struct password**) that contains information about the given user. Then, *tccd* extracts the *pwdir* member from it. This *pwdir* member includes the user's home directory, and its value persists even after the *$HOME* environment variable is modified.

While the solution indeed prevents an attack by environment variable poisoning, it does not protect against the core issue. Thus, we set out to investigate: can an app programmatically change the user's home directory and plant a fake *TCC.db* file?

The first POC exploit

Our first attempt to answer the above question was simple: plant a fake *TCC.db* file and change the home directory using the Directory Services command-line utility (dscl):

While requiring root access, we discovered that this works only if the app is granted with the TCC policy *kTCCServiceSystemPolicySysAdminFiles*, which the local or user-specific *TCC.db* maintains. That is weaker than having full disk access, but we managed to bypass that restriction with the dsexport and dsimport utilities.

Next, simply by exporting the Directory Services entry of a user, manipulating the output file, and importing the file again, we managed to bypass the *dscl* TCC policy restriction.

Our first POC exploit, therefore, does the following:

1. Get a *csreq* blob for the target app.
2. Plant a fake *TCC.db* file with required access and the *csreq* blob.
3. Export the user's Directory Services entry with *dsexport*.
4. Modify the Directory Services entry to change the user's home directory.
5. Import the modified Directory Services entry with *dsimport*.
6. Stop the user's *tccd* and reboot the process.

Using this exploit, an attacker could change settings on any application. In the screenshot below, we show how the exploit could allow attackers to enable microphone and camera access on any app, for example, Teams.

Screenshot of the working exploit

Figure 6. Our first working POC exploit working without a popup notification from TCC.

We reported our initial findings to the Apple product security team on July 15, 2021, before becoming aware of a similar bypass presented by Wojciech Reguła and Csaba Fitzl at BlackHat USA 2021 in August. However, our exploit still worked even after Apple fixed the said similar finding (now assigned as CVE-2020-27937). Therefore, we still considered our research to be a new vulnerability.

Monterey release and the second POC exploit

We shared our findings to Apple through Coordinated Vulnerability Disclosure (CVD) via Microsoft Security Vulnerability Research (MSVR) before the release of macOS Monterey in October. However, upon the release of the said version, we noticed that our initial POC exploit no longer worked because of the changes made in how the *dsimport* tool works. Thus, we looked for another way of changing the home directory silently.

While examining macOS Monterey, we came across */usr/libexec/configd*, an Apple binary shipped with the said latest macOS release that is a System Configuration daemon responsible for many configuration aspects of the local system. There are three aspects of *configd* that we took note and made use of:

1. It is an Apple-signed binary entitled with "*com.apple.private.tcc.allow*" with the value *kTCCServiceSystemPolicySysAdminFiles*. This means it can change the home directory silently.
2. It has extensibility in configuration agents, which are macOS Bundles under the hood. This hints that it might load a custom Bundle, meaning we could inject code for our purposes.
3. It does not have the hardened runtime flag to load custom configuration agents. While this aspect is most likely by design, it also means we could load completely unsigned code into it.

By running *configd* with the *-t* option, an attacker could specify a custom Bundle to load. Therefore, our new POC exploit replaces the *dsexport* and *dsimport* method of changing the user's home directory with a *configd* code injection. This results in the same outcome as our first POC exploit, which allows the modification of settings to grant, for example, any app like Teams, to access the camera, among other services.

As before, we shared our latest findings with Apple. Again, we want to thank their product security team for their cooperation.

Detecting the powerdir vulnerability with Microsoft Defender for Endpoint

Our research on the powerdir vulnerability is yet another example of the tight race between software vendors and malicious actors: that despite the continued efforts of the former to secure their applications through regular updates, other vulnerabilities will inevitably be uncovered, which the latter could exploit for their own gain. And as system vulnerabilities are possible entry points for attackers to infiltrate an organization's network, comprehensive protection is needed to allow security teams to manage vulnerabilities and threats across all platforms.

Microsoft Defender for Endpoint is an industry-leading, cloud-powered endpoint security solution that lets organizations manage their heterogeneous computing environments through a unified security console. Its threat and vulnerability management capabilities empower defenders to quickly discover, prioritize, and remediate misconfigurations and vulnerabilities, such as the powerdir vulnerability. In addition, Defender for Endpoint's unparalleled threat optics are built on the industry's deepest threat intelligence and backed by world-class security experts who continuously monitor the threat landscape.

One of the key strengths of Defender for Endpoint is its ability to generically detect and recognize malicious behavior. For example, as seen in the previous section, our POC exploits conduct many suspicious activities, including:

- Dropping a new *TCC.db* file with an appropriate directory structure
- Killing an existing *tccd* instance
- Suspicious Directory Services invocations such as *dsimport* and *dsexport*

By generically detecting behavior associated with CVE-2020-9934 (that is, dropping a new *TCC.db* file fires an alert), Defender for Endpoint immediately provided protection against these exploits before the powerdir vulnerability was patched. This is a testament of Defender for Endpoint’s capabilities: with strong, intelligent generalization, it will detect similar bypass vulnerabilities discovered in the future.

Screenshot of Microsoft Defender for Endpoint alert for potential TCC bypass

Figure 7. Microsoft Defender for Endpoint detecting potential TCC bypass.

Learn how Microsoft Defender for Endpoint delivers a complete endpoint security solution across all platforms.

Jonathan Bar Or

Microsoft 365 Defender Research Team

Filed under:

Cybersecurity

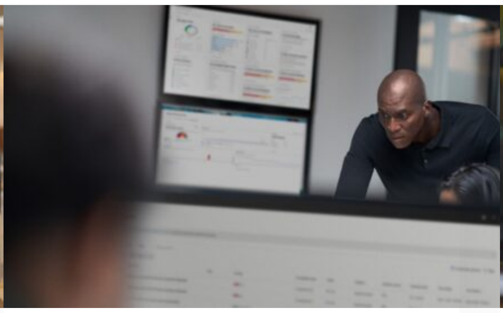
You may also like these articles

January 17, 2023 • 8 min read

Secure your business like you secure your home: 5 steps to protect against cybercrime

Learn five simple actions small and medium-sized businesses can take to protect against evolving cyberattacks, as well as where to access tools and resources for securing your company.

Read more ›

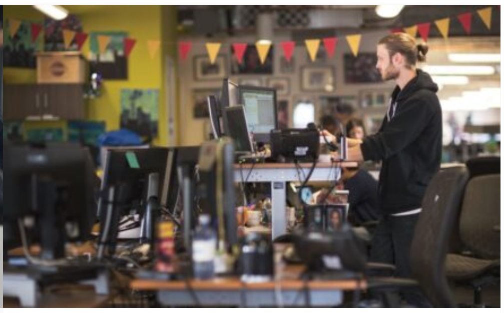


January 9, 2023 • 7 min read

Microsoft Entra: 5 identity priorities for 2023

Organizations are looking for opportunities to do more with less. By adopting the latest identity innovations, you can better protect both your digital estate and your budget.

Read more ›



December 21, 2022 • 12 min read

Microsoft research uncovers new Zerobot capabilities

The Microsoft Defender for IoT research team details information on the recent distribution of a Go-based botnet, known as Zerobot, that spreads primarily through IoT and web-application vulnerabilities.

Read more ›

Get started with Microsoft Security

Microsoft is a leader in cybersecurity, and we embrace our responsibility to make the world a safer place.

LEARN MORE ›

Get all the news, updates, and more at @MSFTSecurity

What's new	Microsoft Store	Education	Business	Developer & IT	Company
Surface Pro 9	Account profile	Microsoft in education	Microsoft Cloud	Azure	Careers
Surface Laptop 5	Download Center	Devices for education	Microsoft Security	Developer Center	About Microsoft
Surface Studio 2+	Microsoft Store support	Microsoft Teams for	Dynamics 365	Documentation	Company news

Surface Laptop Go 2

Surface Laptop Studio

Surface Duo 2

Microsoft 365

Windows 11 apps

Returns

Order tracking

Virtual workshops and training

Microsoft Store Promise

Flexible Payments

Education

Microsoft 365 Education

Education consultation appointment

Educator training and development

Deals for students and parents

Azure for students

Microsoft 365

Microsoft Power Platform

Microsoft Teams

Microsoft Industry

Small Business

Microsoft Learn

Microsoft Tech Community

Azure Marketplace

AppSource

Visual Studio

Privacy at Microsoft

Investors

Diversity and inclusion

Accessibility

Sustainability

English (United States)

Sitemap Contact Microsoft Privacy Manage cookies Terms of use Trademarks Safety & eco Recycling About our ads © Microsoft 2023