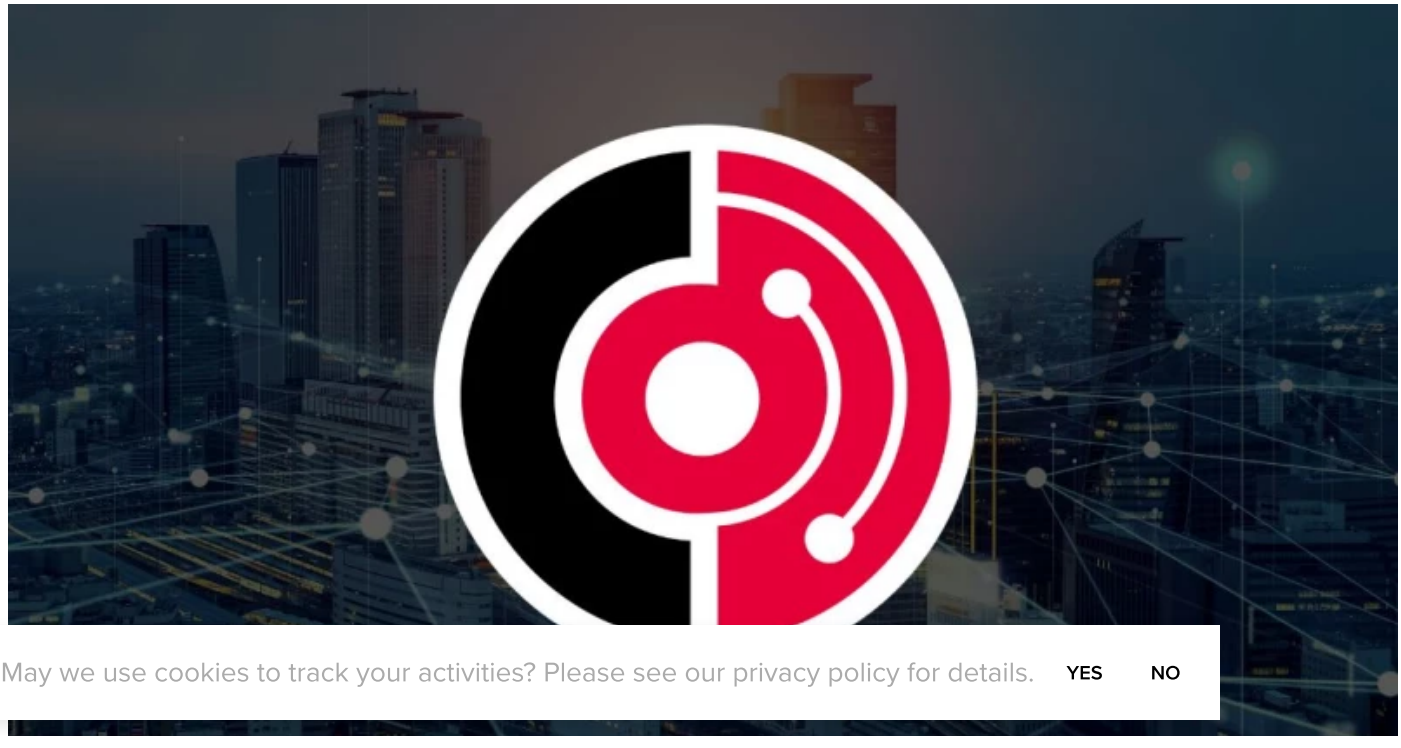
Team82 Research

With Management Comes Risk: Finding Flaws in FileWave MDM

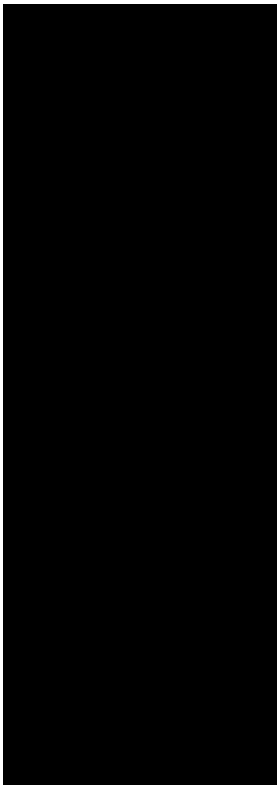
Noam Moshe / July 25th, 2022



Executive Summary

- Team82 has uncovered and disclosed two critical vulnerabilities, CVE-2022-34907 and CVE-2022-34906, in FileWave's mobile device management (MDM) system.
- The vulnerabilities are remotely exploitable and enable an attacker to bypass authentication mechanisms and gain full control over the MDM platform and its managed devices.
- **CVE-2022-34907**, an authentication bypass flaw exists in FileWave MDM before version 14.6.3 and 14.7.x, prior to 14.7.2. This vulnerability is similar in nature to the vulnerability that was recently identified in **F5 BIG-IP WAF**.
- **CVE-2022-34906**, a hard-coded cryptographic key, exists in FileWave MDM prior to version 14.6.3 and 14.7.x, prior to 14.7.2.
- During our research, we found thousands of vulnerable internet-facing FileWave servers in numerous industries, including government agencies, education, and large enterprises.
- **FileWave has addressed these vulnerabilities in a recent update**, and users are urged to apply the update. FileWave has also written a blog about its **resolution of these vulnerabilities**.
- Team82 wishes to acknowledge and thank FileWave for its cooperation and coordination throughout this disclosure. These vulnerabilities were addressed in a prompt, complete fashion, users were notified, and the vast majority were verified as up to date.

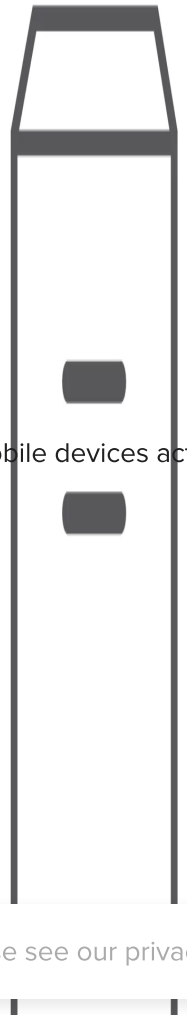
Below is a demonstration of Team82's proof-of-concept exploit in action—described in the last section of this report. Our attack compromises the Filewave MDM, then infects each managed device with a phony ransomware sample:



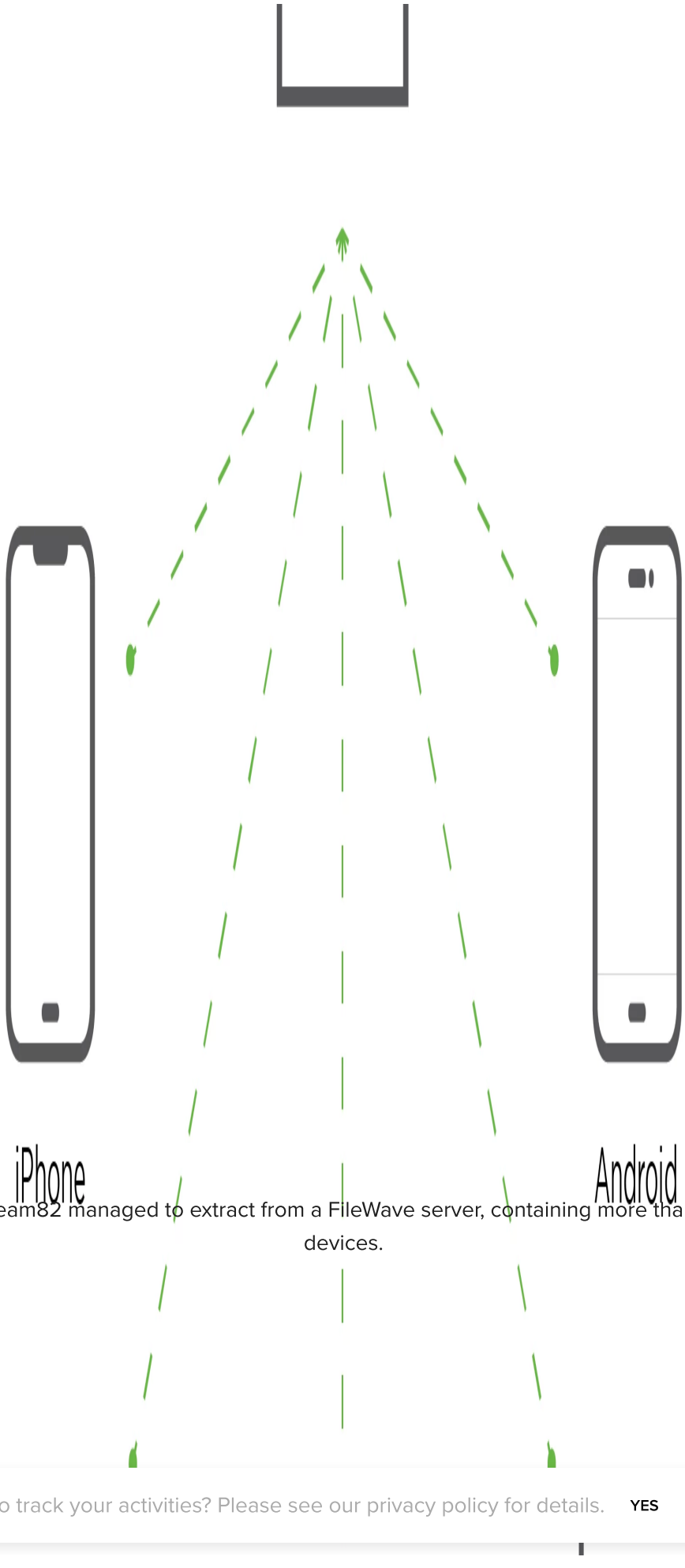
0:00 / 0:47

Introduction

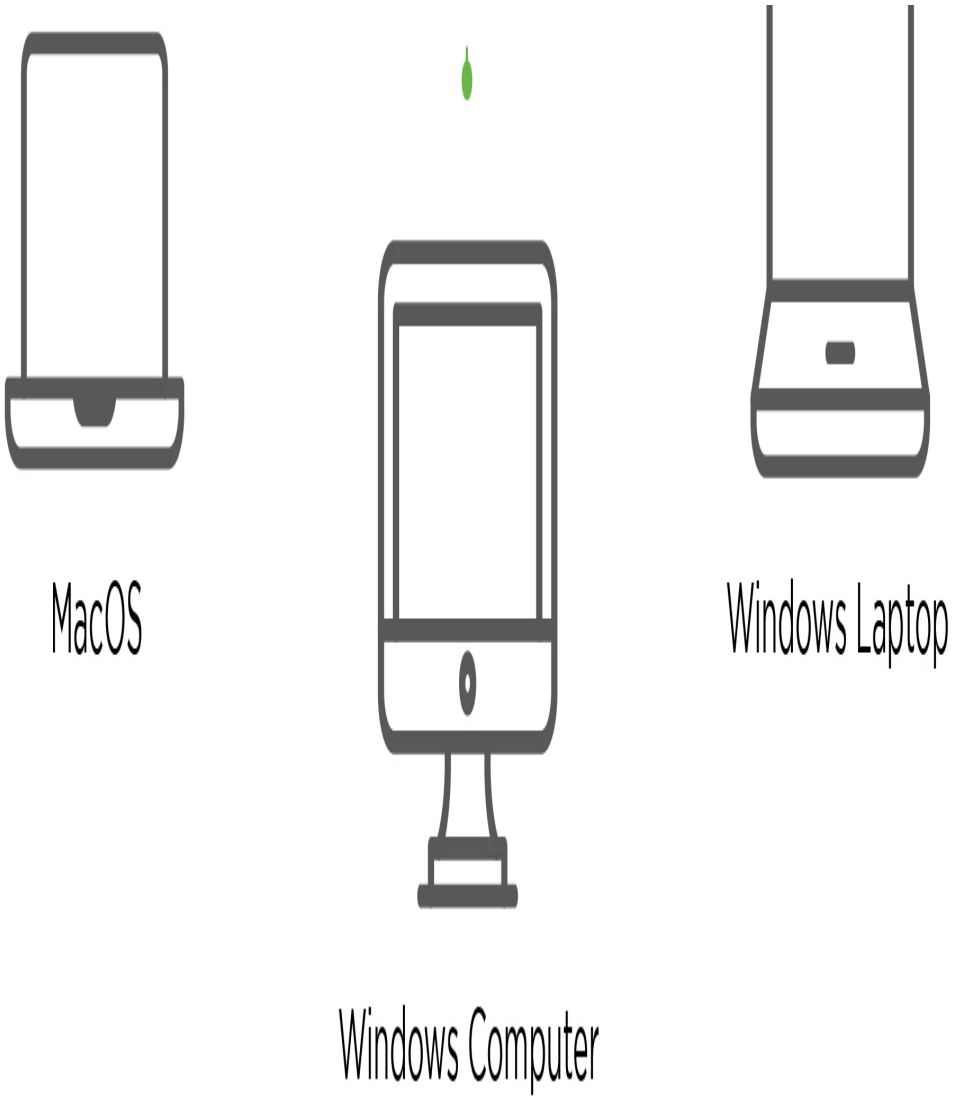
FileWave MDM is a multi-platform mobile device management solution that allows IT administrators to manage, monitor, and view all of an organization’s devices. Currently, FileWave MDM supports a wide range of devices, from iOS and Android smartphones, MacOS and Windows tablets, laptops and workstations, and smart devices such as televisions.



FileWave MDM manages mobile devices actively used across industries.



Example of data Team82 managed to extract from a FileWave server, containing more than 10,000 managed devices.



An attacker capable of compromising a FileWave MDM server can exploit all managed devices.

To fully understand the range and relevance of this vulnerability, we had to first know how many organizations actually use this product. In order to do so, we've used a few different methods which netted us with more than 1,100 different instances of FileWave MDM, each vulnerable to the vulnerabilities described below. Between those exposed services, we've discovered organizations

from many different fields, including corporations, schools and educational institutions, government agencies and small-to-medium businesses.

Technical Details

Looking into CVE-2022-34907: Authentication Bypass

An important FileWave MDM component is the MDM web server, written in Python using the Django framework. It exposes TCP port 20443 and 20445

The web server handles not only client devices but also the admin's GUI application. It retrieves device information from clients, handles device enrollment, and supplies commands to devices. For the admin, the web server returns data about client devices using different query methods, and allows the administrator to control all managed devices, including the ability to change a device configuration remotely, install packages on the device, and remotely control it.

Since this service should be accessible to mobile devices at all times, it is usually exposed to the internet, and handles both clients' and admins' requests. Its connectivity makes it a primary target in our research on this platform.

We discovered that this server handles different client types, each authenticating differently. Firstly, there are the mobile devices. By default devices can start the enrollment process and interact with the server without requiring any form of user-based authentication (although the option to require credentials in the enrollment process does exist in FileWave MDM). All enrolled devices are first placed in the "quarantine" group, a logical group of devices that are not part of the organization. This means that all routes that involve device enrollment do not require authentication, however since these routes do not allow us to exfiltrate sensitive data, or to take control over devices, this vector was less explored by us

FileWave MDM enrollment routes. Through this web server, mobile devices enroll to this specific instance and allow the IT administrators to manage their device.

Then, there is system administrator authentication. This method takes a username/password combination, and returns a valid token. Using this token, the IT admin could retrieve data about devices, change device configurations, and much more. In most cases, this token is checked correctly, and allows only valid users to access routes that require authentication.

The FileWave MDM admin interface. Through this interface, IT administrators can authenticate to the MDM platform.

Lastly, we researched the backend services running on the MDM server, which performs another type of authentication within this service. Except for the MDM web server, there are a few extra services that exist in the FileWave MDM ecosystem. One service in particular, the scheduler service, also written in Python, caught our attention. Its purpose is to schedule and execute specific tasks that the MDM platform needs to perform, and to call the corresponding callback whenever the tasks are completed.

As part of the business logic of the system, the web server uses the scheduler in many cases, and in return, the scheduler informs the web server whenever a task is finished. In order to do so, the scheduler calls some specific web routes in the web server, routes that are accessible only to authenticated and authorized users in the system. However, the scheduler does not know the administrator's account details, and instead uses a hardcoded shared secret in order to authenticate to the web server. This shared secret does not change between each installation of FileWave MDM, nor between different versions of the system.

The SCHEDULER_SECRET variable is used by both the scheduler service and web server in order to verify and authenticate the scheduler. This is configured inside this file: /usr/local/filewave/django/filewave/settings_common.py.

In FileWave MDM platform, each route that requires valid authentication must inherit the FWAuthMixin class defined in /usr/local/filewave/django/fwauth/utillity.py (or any class that inherits this class). This check is performed inside the test_func function, where if this function returns True the request will be fulfilled, and if this function returns False, a 401 Unauthorized will be returned. When we looked into this function we found the code, below, (some code was redacted because it was irrelevant):

The code of the test_function function that decides whether a user is authenticated.

As we can see, this function takes the Authorization header from the HTTP request, and compares it to the scheduler secret (after being base64 decoded). If they match, the request is given the permissions of the superuser account, which is the system's administrator account. This means that if [illegible] valid user's [illegible] to the system using the superuser's permissions (which are the highest permissions available).

using the superuser's permissions (which are the highest permissions available).

A request to a vulnerable FileWave MDM server, passing the SCHEDULER_SECRET in its HTTP Authorization header, and therefore the server treats the request as if it was made by the scheduler service.

This vulnerability exists in FileWave up to version 13.1.3. However, when we tested this vulnerability against newer versions of the system, we learned it did not work. Something changed making the vulnerability above irrelevant in newer versions, so we decided to look into what that was in newer versions and find a bypass to this new security mechanism.

As it turns out, FileWave changed this logic inside the FWAuthMixin class, instead only accepting valid users' tokens.

The code to the test_func function in versions newer than 13.1.3. We can see that the server no longer compares the authorization header to the scheduler secret.

This change had us stumped for a second, thinking our exploit is no longer viable. However, when we kept searching for new code, we discovered that FileWave added a new middleware (code which runs before requests are handled). This middleware, aptly named AppTokenMiddleware (and defined in /usr/local/filewave/django/fwauth/middleware.py) performs a similar action to the previous one used inside the older version of test_func function.

The new middleware function code.

uced,

comparing request.gethost() to localhost. If we pass this check, we will once again be granted the permissions of the superuser. When we searched for what get_host returns, we reached this **documentation from Django**:

As we can see, this value also comes from headers users supply, namely from the Host HTTP header. This means that because we supply this value, we can simply supply localhost as our value, thus passing this new check and gaining the super_user permissions.

We can see that in this request in newer versions of FileWave MDM, the Host header is changed to bypass the new check and be localhost, thus passing the check and gaining the privileges of super_user.

Using this vulnerability, in either variety described above, we were able to gain highest privileges in all versions of the FileWave MDM. This allows us to gain the ability to attack and control every instance exposed to the internet. This enables us to control all of the servers' managed devices, exfiltrate all sensitive data being held by the devices, including usernames, email addresses, IP addresses, geo-location etc, and install malicious software on managed devices.

In order to showcase this vulnerability and the severity and potential harm it can cause, we created a standard FileWave setup, and enrolled 6 devices of our own. Then, using this vulnerability, we exploited the MDM web server, which allowed us to leak data about all of the devices managed by this MDM server.

Lastly, using regular MDM functionality which allows IT administrators to install packages and software on managed devices, we installed malicious packages on each controlled device, popping a fake ransomware virus on each of those managed devices. Doing so, we demonstrated how a potential attacker can leverage Filewave's capabilities in order to take control over different managed devices.

We started our testbed with multiple devices connected to the Filewave server.

Then, we ran our exploit in order to bypass authentication on the MDM server and gain administrative access to it. Using this access, we exfiltrated information about the managed devices, including their operation system, ecosystem, settings, and much more.

Finally, We pushed a malicious package to all of the managed devices, which resulted in us being able to execute remote code on all devices. We used it to install fake ransomware on each device.

Acknowledgement

Team82 would like to thank Filewave for its coordination with us in working through this disclosure, and for its swift response in confirming our findings and swiftly patching these vulnerabilities. Filewave has addressed these issues in a **recent update, v14.7.2**, and worked with their customers to patch or update affected systems.

Share:

Recent Vulnerability Disclosures

CVE-2022-3086

CVE-2022-38465

CVE-2022-41666

CVE-2022-41667

CVE-2022-41669

SOLUTIONS

Industrial Cybersecurity

Healthcare Cybersecurity

Commercial Cybersecurity

THREAT RESEARCH

Team82 Home

Vulnerability Disclosure Dashboard

Research

PGP Key

PARTNERS

Partners

Technology Alliance Partners

Channel Partners

Become a Partner

Find a Partner

Partner Login

RESOURCES

Resource Library

Blog

White Papers

Reports

Case Studies

Datasheets

Integration Briefs

Podcasts

Videos