Lazarus Group Targeting Windows IIS Web Servers - ASEC

By ATCP

Published: 2023-05-16 · Archived: 2026-04-05 15:46:19 UTC

AhnLab Security Emergency response Center (ASEC) has recently confirmed the Lazarus group, a group known to receive support on a national scale, carrying out attacks against Windows IIS web servers. Ordinarily, when threat actors perform a scan and find a web server with a vulnerable version, they use the vulnerability suitable for the version to install a web shell or execute malicious commands. The AhnLab Smart Defense (ASD) log displayed below in Figure 1 shows that Windows server systems are being targeted for attacks, and malicious behaviors are being carried out through w3wp.exe, an IIS web server process. Therefore, it can be assumed that the threat actor uses poorly managed or vulnerable web servers as their initial breach routes before executing their malicious commands later.

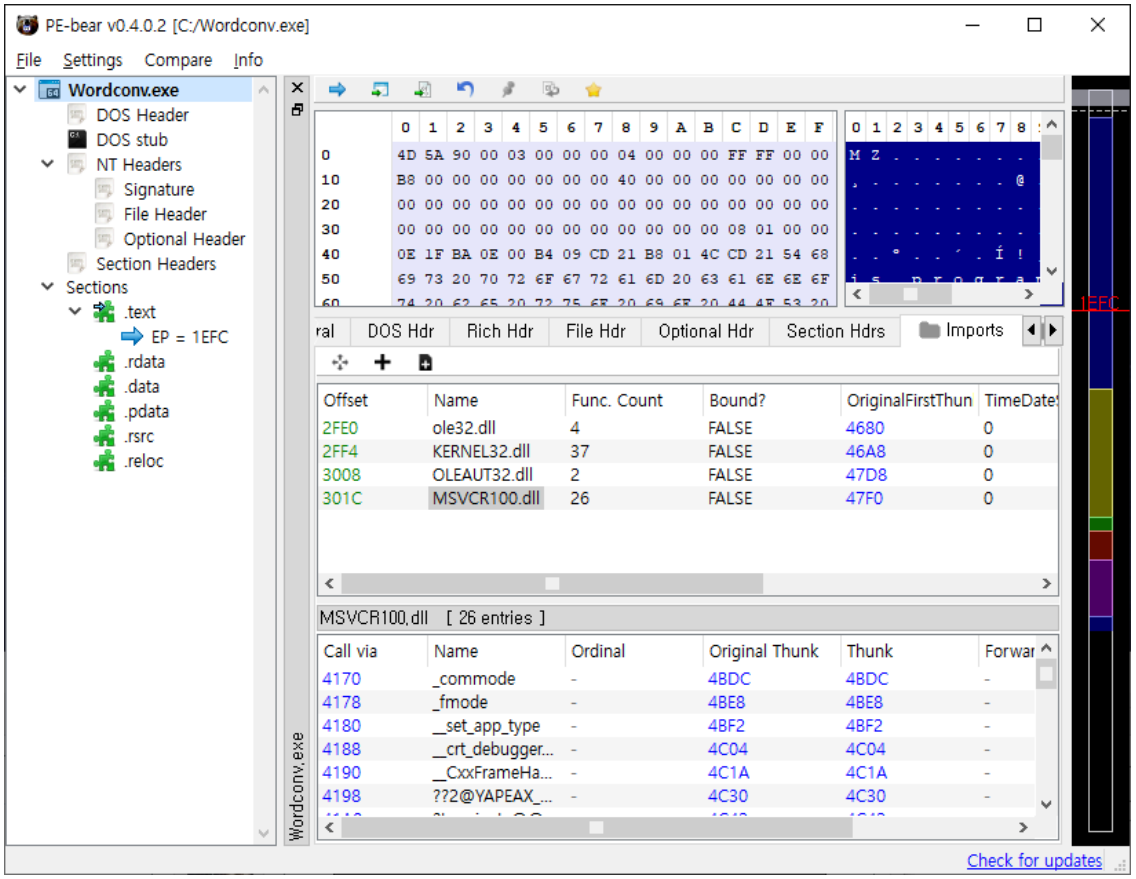
The threat actor places a malicious DLL (msvcr100.dll) in the same folder path as a normal application (Wordconv.exe) via the Windows IIS web server process, w3wp.exe. They then execute the normal application to initiate the execution of the malicious DLL. In MITRE ATT&CK, this method of attack is categorized as the DLL side-loading (**T1574.002**) technique.

Target Type	File Name	File Size	File Path
Current	wordconv.exe	25.61 KB	%ALLUSERSPROFILE%\usoshared\wordconv.exe
Parent	w3wp.exe	23.5 KB	%SystemRoot%\syswow64\inetsrv\w3wp.exe
InjectorOfCurrent	wordconv.exe	25.61 KB	%ALLUSERSPROFILE%\usoshared\wordconv.exe
GeneratedByParent	wordconv.exe	25.61 KB	%ALLUSERSPROFILE%\usoshared\wordconv.exe
GeneratedByParent	msvcr100.dll	81 KB	%ALLUSERSPROFILE%\usoshared\msvcr100.dll

The Lazarus group's use of the DLL side-loading technique to run malware has been **confirmed** many times already. The threat actor has been continuously changing the name of the normal process used in the DLL side-loading technique. This post will cover the DLL side-loading technique used by the threat actor during their initial infiltration process as well as their follow-up behaviors.

1. Initial Infiltration: DLL Side-Loading Using Windows IIS Web Servers (Wordconv.exe, msvcr100.dll)

The threat actor creates Wordconv.exe, msvcr100.dll, and msvcr100.dat through the Windows IIS web server process (w3wp.exe) before executing Wordconv.exe. As shown in the below figure, msvcr100.dll is contained within the import DLL list of Wordconv.exe, so the first DLL file that is loaded when Wordconv.exe is executed is determined by the DLL search priority of the operating system. As a result, the malicious msvcr100.dll is run in the memory of the Wordconv.exe process.



As can be seen in the below Figure 3, the functionality of msvcr100.dll involves decrypting an encoded PE file (msvcr100.dat) and the key (df2bsr2rob5s1f8788yk6ddi4x0wz1jq) that is transmitted as a command-line argument during the execution of Wordconv.exe by utilizing the Salsa20 algorithm. The decrypted PE file is then executed in the memory. It then performs the function of clearing the malicious DLL module that was loaded through the FreeLibraryAndExitThread WinAPI call before deleting itself (msvcr100.dll).

```
"targetProcess": {
  "imageInfo": {
    "commandLine": "%ALLUSERSPROFILE%\\usoshared\\wordconv.exe df2bsr2rob5s1f8788yk6ddi4x0wz1jq c:\\programdata\\usoshared\\msvcr100.dat"
```

Also, msvcr100.dll is very similar in both appearance and features to the cylvc.dll malware covered in the ASEC Blog post **"A Case of Malware Infection by the Lazarus Attack Group Disabling Anti-Malware Programs With the BYOVD Technique"**, which was released back in 2022. Thus, it is speculated that msvcr100.dll is a variant malware of cylvc.dll.

msvcr100.dll

```
v91 = -2i64;
v0 = sub_180002E74(v92, "1BIVvTOhEGqW3M6/");  // kernel32.dll
v1 = sub_180002F60(v0);
LibraryA = LoadLibraryA(v1);
sub_180002EF4(v92);
if ( !LibraryA )
  return 0i64;
v4 = sub_180002E74(v90, "+BITgySiQBnc3NC2SaM=");// GetProcAddress
v5 = sub_180002F60(v4);
qword_1800154C0 = GetProcAddress(LibraryA, v5);
sub_180002EF4(v90);
v6 = sub_180002E74(v90, "/BsIoDOFQjbc1Mc=");  // CloseHandle
v7 = sub_180002F60(v6);
qword_180015430 = qword_1800154C0(LibraryA, v7);
```

cylvc.dll

```
v91 = -2i64;
v0 = sub_180002AC0(v92, "1BIVvTOhEGqW3M6/");  // kernel32.dll
v1 = sub_180002BAC(v0);
LibraryA = LoadLibraryA(v1);
sub_180002B40(v92);
if ( !LibraryA )
  return 0i64;
v4 = sub_180002AC0(v90, "+BITgySiQBnc3NC2SaM=");// GetProcAddress
v5 = sub_180002BAC(v4);
qword_1800154E8 = GetProcAddress(LibraryA, v5);
sub_180002B40(v90);
v6 = sub_180002AC0(v90, "/BsIoDOFQjbc1Mc=");  // CloseHandle
v7 = sub_180002BAC(v6);
```

Similarly to msvcr100.dll, cylvc.dll decrypts the data files with the .dat extension using the Salsa20 algorithm before executing the PE file within the memory space. The PE that was executed within the memory space back in 2022 was a backdoor that communicated with the threat actor’s C&C server.

msvcr100.dll

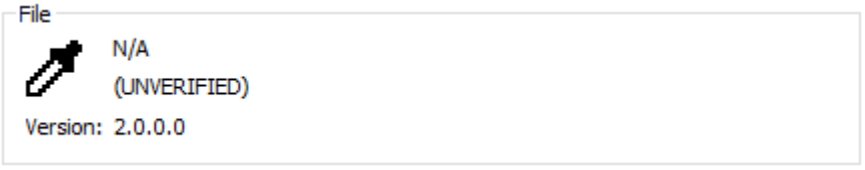
```
v2 = 0;
LODWORD(Size) = 0;
if ( !sub_1800014C8(a1, 0, &Size, 4u) )
  return 0i64;
v6 = operator new(Size);
if ( !sub_1800014C8(a1, 4u, v6, Size) )
{
  if ( v6 )
    j_free(v6);
  return 0i64;
}
v7 = Size;
memset(v16, 0, 260);
memset(&Buf1[1], 0, 0x20ui64);
memset(Buf2, 0, 33);
wsprintfA(v16, "%S", a2);
v8 = (v7 - 32);
memmove_0(Buf1, &v6[v8], 0x20ui64);          // argument : Key
v9 = operator new(v8);
memset(v9, 0, v8);
sub_180001938(v16, v6, v8, v9);              // Salsa20
memset(Buf2, 0, 0x20ui64);
v10 = 0;
if ( v8 > 0 )
{
  v11 = v9;
  do
  {
    v12 = v10 % 32;
    ++v10;
    Buf2[v12] += *v11++;
  }
  while ( v10 < v8 );
}
if ( !memcmp(Buf1, Buf2, 0x20ui64) )
{
  sub_180002A8C(v9);
  return 1;
}
else if ( v9 )
{
  j_free(v9);
}
return v2;
```

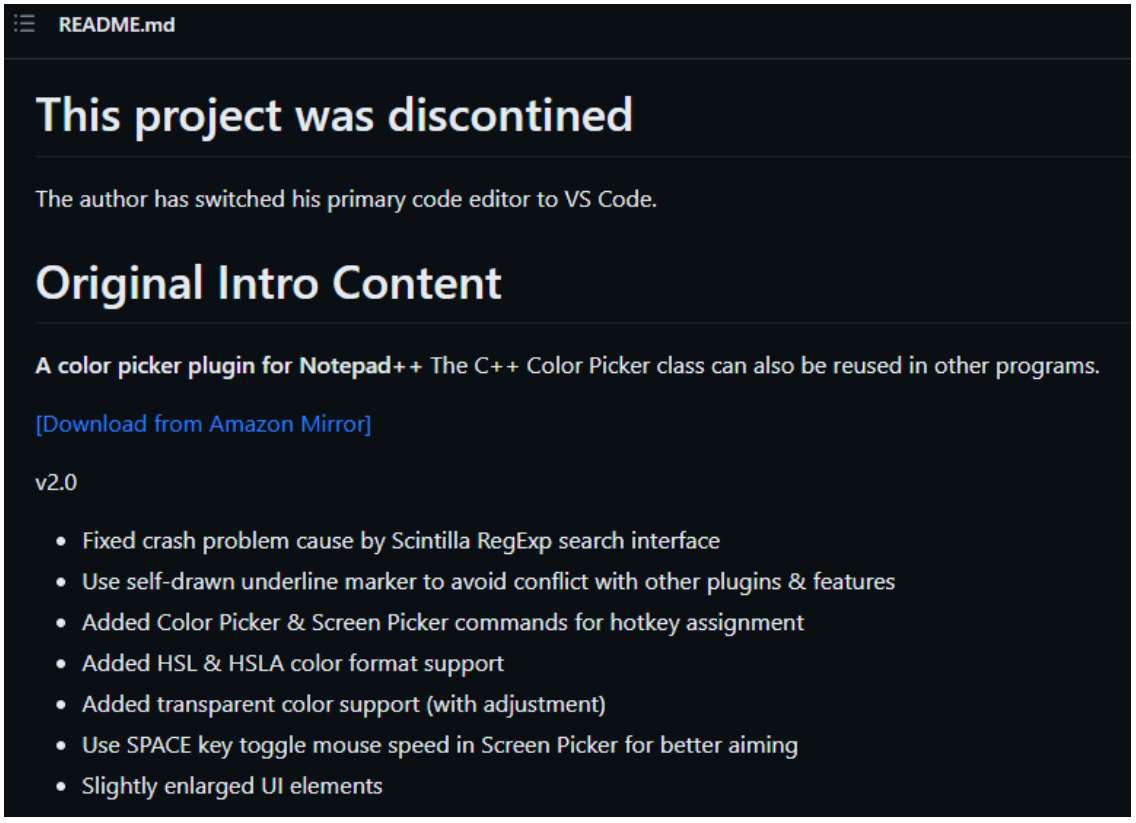
cylvc.dll

```
v2 = 0;
LODWORD(Size) = 0;
if ( !sub_180001048(a1, 0i64, &Size, 4i64) )
  return 0i64;
v6 = operator new(Size);
if ( !sub_180001048(a1, 4i64, v6, Size) )
{
  if ( v6 )
    j_free(v6);
  return 0i64;
}
v7 = Size;
memset(v16, 0, 260);
memset(&Buf1[1], 0, 0x20ui64);
memset(Buf2, 0, 33);
wsprintfA(v16, "%S", a2);                    // argument : Key
v8 = (v7 - 32);
memmove(Buf1, &v6[v8], 0x20ui64);
v9 = operator new(v8);
memset(v9, 0, v8);
sub_1800014B8(v16, v6, v8, v9);              // Salsa20
memset(Buf2, 0, 0x20ui64);
v10 = 0;
if ( v8 > 0 )
{
  v11 = v9;
  do
  {
    v12 = v10 % 32;
    ++v10;
    Buf2[v12] += *v11++;
  }
  while ( v10 < v8 );
}
if ( !memcmp(Buf1, Buf2, 0x20ui64) )
{
  sub_1800026D8(v9);
  return 1;
}
else if ( v9 )
{
  j_free(v9);
}
return v2;
```

2. Establishing Foothold and Stealing Certificates

After the initial infiltration, the threat actor established a foothold before creating additional malware (diagn.dll) by exploiting the **open-source "color picker plugin"**, which is a plugin for Notepad++.




README.md

This project was discontined

The author has switched his primary code editor to VS Code.

Original Intro Content

A color picker plugin for Notepad++ The C++ Color Picker class can also be reused in other programs.

[Download from Amazon Mirror]

v2.0

- Fixed crash problem cause by Scintilla RegExp search interface
- Use self-drawn underline marker to avoid conflict with other plugins & features
- Added Color Picker & Screen Picker commands for hotkey assignment
- Added HSL & HSLA color format support
- Added transparent color support (with adjustment)
- Use SPACE key toggle mouse speed in Screen Picker for better aiming
- Slightly enlarged UI elements

diagn.dll is responsible for receiving the PE file encoded with the RC6 algorithm as an execution argument value before using an internally hard-coded key to decrypt the data file and execute the PE file in the memory.

- RC6 key: 5A 27 A3 E8 91 45 BE 63 34 23 11 4A 77 91 53 31 5F 47 14 E2 FF 75 5F D2 3F 58 55 6C A8 BF 07 A1

The malicious behavior of the PE file executed in the memory is unknown since the PE data file that was encoded during the attack could not be collected, but a log was confirmed through the AhnLab Smart Defense (ASD) infrastructure of the threat actor accessing the memory space of the lsass.exe process through this module. Thus, it is suspected that the threat actor had executed a credential theft tool such as Mimikatz.

Process	Module	Behavior	Rule Desc	Data
rundll32.exe	diagn.dll ASD.Prevention(100)	Collected data	Environment variable lookup	
rundll32.exe	diagn.dll ASD.Prevention(100)	Collected data	Computer information collected	
rundll32.exe	diagn.dll ASD.Prevention(100)	Opens process	Accessed to process(lsass.exe) to steal account information	Target Process lsass.exe

3. Lateral Movement

After acquiring the system credentials, the threat actor performed internal reconnaissance before utilizing remote access (port 3389) to perform lateral movement into the internal network. No further malicious activities by the threat actor have been uncovered since then.

Process	Module	Target	Behavior	Rule DESC	Data
File Less Submit diag.dat	N/A	N/A	Connects to network	Connects to network.	10.119.0.9:3389

4. Conclusion and Response

The Lazarus group used a variety of attack vectors to perform their initial breach, including **Log4Shell**, **public certificate vulnerability**, **3CX supply chain attack**, etc. This group is one of the highly dangerous groups that are actively launching attacks worldwide. Therefore, corporate security managers should utilize attack surface management to identify the assets that could be exposed to threat actors and practice caution by applying the latest security patches whenever possible.

In particular, since the threat group primarily utilizes the DLL side-loading technique during their initial infiltrations, companies should proactively monitor abnormal process execution relationships and take preemptive measures to prevent the threat group from carrying out activities such as information exfiltration and lateral movement.

AhnLab's products detect and block the malware identified in the attack case covered in this post using the following aliases.

[File Detection]
– Trojan/Win.LazarLoader.C5427612 (2023.05.15.02)
– Trojan/Win.LazarLoader.C5427613 (2023.05.15.03)

MD5

228732b45ed1ca3cda2b2721f5f5667c

47d380dd587db977bf6458ec767fee3d

4d91cd34a9aae8f2d88e0f77e812cef7

e501bb6762c14baafadbde8b0c04bbd6

Additional IOCs are available on AhnLab TIP.

URL

https[:]//www[.]samdb[.]or[.]kr/info/pinfo[.]aspx

Additional IOCs are available on AhnLab TIP.
