Gh0stCringe RAT Being Distributed to Vulnerable Database Servers

By ATCP

Published: 2022-03-07 · Archived: 2026-04-06 00:45:36 UTC

The ASEC analysis team is constantly monitoring malware distributed to vulnerable database servers (MS-SQL, MySQL servers). This blog will explain the RAT malware named Gh0stCringe[1].

Gh0stCringe, also known as CirenegRAT, is one of the malware variants based on the code of Gh0st RAT. It was first discovered in December 2018, and it is known to have been distributed via SMB vulnerability (using the SMB vulnerability tool of ZombieBoy).[2] Since then, no direct relationship has been found, but it was mentioned in the KingMiner CoinMiner analysis report[3] published in June 2020.

Gh0stCringe RAT that is recently being discovered is being distributed to vulnerable database servers. Gh0stCringe-related logs in AhnLab's ASD show that logs were not only created by the sqlservr.exe process (MS-SQL server) but also by the MySQL server process for Windows environment (see figure below).

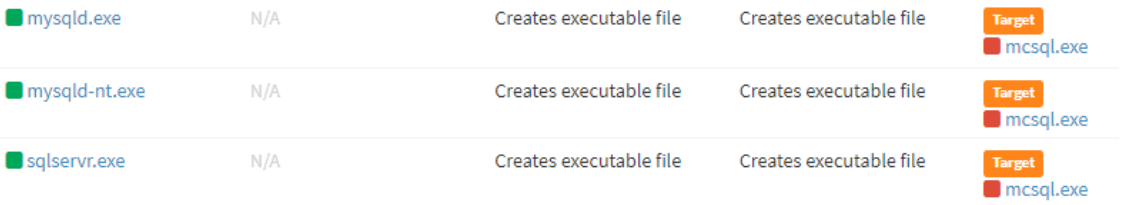

Figure 1. Gh0stCringe RAT creation logs

Considering the fact that MySQL servers are targets of attack in addition to MS-SQL servers, it can be assumed that Gh0stCringe targets poorly-managed DB servers with vulnerable account credentials.

Since database servers with vulnerable account credentials usually become targets of various attackers and malware, many different malware infection logs were found. In fact, the typical attack path of KingMiner malware that was mentioned above was an attack against SQL servers with vulnerable account credentials. Furthermore, the infection log of systems installed with Gh0stCringe shows a history of infection from different malware other than KingMiner such as Vollgar CoinMiner[4] that are distributed through brute force attacks.

Gh0stCringe was created based on the source code of publicly released Gh0st RAT. The following is a comparison between the CFileManager::OpenFile() function of Gh0st RAT that was publicly released and the function of Gh0stCringe. It shows that a part of the source code was used without modifications. Note that unlike normal variants where the majority of their codes is similar to the original code based on the Gh0st RAT source code, the majority of Gh0stCringe codes is unique just like how Gh0stCringe has its own name.

```
if (RegOpenKeyEx(HKEY_CLASSES_ROOT, lpExt, 0L, KEY_ALL_ACCESS, &hKey) != ERROR_SUCCESS)
        return false;
RegQueryValue(hKey, NULL, strTemp, &nSize);
RegCloseKey(hKey);
memset(lpSubKey, 0, sizeof(lpSubKey));
wsprintf(lpSubKey, "%s\\shell\\open\\command", strTemp);

if (RegOpenKeyEx(HKEY_CLASSES_ROOT, lpSubKey, 0L, KEY_ALL_ACCESS, &hKey) != ERROR_SUCCESS)
        return false;
memset(strTemp, 0, sizeof(strTemp));
nSize = sizeof(strTemp);
RegQueryValue(hKey, NULL, strTemp, &nSize);
RegCloseKey(hKey);

lpstrCat = strstr(strTemp, "\"%1");
if (lpstrCat == NULL)
        lpstrCat = strstr(strTemp, "%1");

if (lpstrCat == NULL)
{
        lstrcat(strTemp, " ");
        lstrcat(strTemp, lpFile);
}
else
        lstrcpy(lpstrCat, lpFile);

STARTUPINFO si = {0};
PROCESS_INFORMATION pi;
si.cb = sizeof si;
if (nShowCmd != SW_HIDE)
        si.lpDesktop = "WinSta0\\Default";

CreateProcess(NULL, strTemp, NULL, NULL, false, 0, NULL, NULL, &si, &pi);
```

```
if ( RegOpenKeyExA(HKEY_CLASSES_ROOT, v2, 0, 0xF003Fu, &phkResult) )
  return 0;
RegQueryValueA(phkResult, 0, Data, &cbData);
RegCloseKey(phkResult);
memset(SubKey, 0, sizeof(SubKey));
wsprintfA(SubKey, "%s\\shell\\open\\command", Data);
if ( RegOpenKeyExA(HKEY_CLASSES_ROOT, SubKey, 0, 0xF003Fu, &phkResult) )
  return 0;
memset(Data, 0, sizeof(Data));
cbData = 260;
RegQueryValueA(phkResult, 0, Data, &cbData);
RegCloseKey(phkResult);
v4 = strstr(Data, a1_0);                    // ""%1"
if ( v4 || (v4 = strstr(Data, a1)) != 0 )   // "%1"
{
  lstrcpyA(v4, Str);
}
else
{
  lstrcatA(Data, SubStr);                   // " "
  lstrcatA(Data, Str);
}
memset(&StartupInfo.lpReserved, 0, 0x40u);
StartupInfo.cb = 68;
if ( a2 )
  StartupInfo.lpDesktop = aWinsta0Default;  // "WinSta0\Default"
return CreateProcessA(0, Data, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
```

Figure 2. Gh0stCringe that utilizes Gh0st RAT source code

As its name suggests, Gh0stCringe RAT is a RAT malware that connects to the C&C server and performs various malicious behaviors by receiving commands from the attacker. The attacker can designate various settings to Gh0stCringe just like other RAT malware. The explanations in this blog are based on the analyzed sample.

The following is settings data of various features that can be performed when executed in addition to the C&C commands. There are 7 settings, and the malware performs different behaviors depending on the configured data.

```
.data:10011FB2                  db    0
.data:10011FB3                  db    0
.data:10011FB4 conf_selfCopy    dd 1                    ; DATA XREF: fn_installService+428↑r
.data:10011FB4                                          ; DllMain(x,x,x)+416↑r
.data:10011FB8 conf_execMode    db 2                    ; DATA XREF: FUCKYOU:loc_1000724B↑r
.data:10011FB8                                          ; FUCKYOU:loc_100072D3↑r ...
.data:10011FB9                  align 2
.data:10011FBA conf_fileAppend  dw 0                    ; DATA XREF: gh0st_SaveToFile↑r
.data:10011FBC conf_AntiAnalysis dd 1                   ; DATA XREF: FUCKYOU+3F↑r
.data:10011FBC                                          ; DllMain(x,x,x)+32↑r
.data:10011FC0 conf_offlineKeylogger dd 0               ; DATA XREF: thread_main+9C↑r
.data:10011FC4                  align 8
.data:10011FC8 conf_killRundll32 dd 0                   ; DATA XREF: FUCKYOU:loc_1000723E↑r
.data:10011FC8                                          ; DllMain(x,x,x):loc_10007BEA↑r
.data:10011FCC conf_fileAttr    dd 7                    ; DATA XREF: fn_installService+1A9↑r
.data:10011FCC                                          ; FUCKYOU+91↑o ...
.data:10011FD0                  db    0
```

Figure 3. Gh0stCringe settings data

- **Self-copy** [On/Off]: If turned on, copies itself to a certain path depending on the mode.
- **Mode of execution** [Mode]: Can have values of 0, 1, and 2. See below for explanations on the modes.

- **File size change** [Size]: In Mode #2, the malware copies itself to the path '%ProgramFiles%\Cccogae.exe', and if there is a set value, it adds junk data of the designated size to the back of the file.
- **Analysis disruption technique** [On/Off]: Obtains the PID of its parent process and the explorer.exe process. If it results in a value of 0, terminates itself.
- **Keylogger** [On/Off]: If turned on, keylogging thread operates.
- **Rundll32 process termination** [On/Off] If turned on, executes 'taskkill /f /im rundll32.exe' command to terminate the rundll32 process that is running.
- **Self-copy file property** [Attr]: Sets property to read-only, hidden, and system (FILE_ATTRIBUTE_READONLY|FILE_ATTRIBUTE_HIDDEN|FILE_ATTRIBUTE_SYSTEM).

The keylogging feature can operate by receiving a command from the C&C server, and it can also be activated depending on the settings data. Unlike Gh0st RAT which uses the Windows Hooking method (use of SetWindowsHookEx() API), Gh0stCringe uses the keylogging technique of Windows Polling method (using GetAsyncKeyState() API).

```
Sleep(0xAu);
if ( lstrlenA(String) )
{
  if ( CKeyboardManager::SaveInfo() )
  {
    CKeyboardManager::SaveToFile(asc_100118AC);
    CKeyboardManager::SaveToFile(String);
  }
  else
  {
    CKeyboardManager::SaveToFile(String);
  }
  memset(String, 0, sizeof(String));
}
v2 = 0;
while ( 1 )
{
  KeyState = GetKeyState(16);
  v4 = dword_100113DC[v2];
  v5 = KeyState;
  if ( ((GetAsyncKeyState(v4) >> 8) & 0x80u) == 0 )
  {
    v6 = v8[v4];
    if ( v6 )
    {
      v8[v4] = 0;
      if ( v4 == 8 )
      {
        lstrcatA(String, String2);
        CKeyboardManager::SaveToFile(String);
```

Figure 4. Keylogging that uses GetAsyncKeyState() API

The malware saves logged user key inputs to the path '%SystemDirectory%\Default.key' by single-byte XOR encoding similarly to Gh0st RAT. In this case, 0x62 was used as the key.

```
FileA = CreateFileA(Buffer, 0x40000000u, 2u, 0, 4u, 0x80u, 0);// "C:\Windows\System32\Default.key"
NumberOfBytesWritten = 0;
if ( GetFileSize(FileA, 0) < 0x3200000 )
  SetFilePointer(FileA, 0, 0, 2u);
size_keylog = lstrlenA(data_keylog);
lpEncodeBuffer = operator new(size_keylog);
lpEncodeBuffer_1 = lpEncodeBuffer;
if ( size_keylog > 0 )
{
  v1 = (data_keylog - lpEncodeBuffer);
  do
  {
    *lpEncodeBuffer = lpEncodeBuffer[v1] ^ 0x62;
    ++lpEncodeBuffer;
    --size_keylog;
  }
  while ( size_keylog );
}
nLength = lstrlenA(data_keylog);
WriteFile(FileA, lpEncodeBuffer_1, nLength, &NumberOfBytesWritten, 0);
```

Figure 5. XOR encoding with 0x62

Gh0stCringe supports four different modes. The three modes are the values of the settings data mentioned above, which were 0, 1, and 2. In addition to them, the malware is executed in an exclusive mode in Windows 10 version. All modes ultimately communicate with the C&C server and perform commands, but there are differences in their features related to maintaining persistence.

- **Mode #0**: If the ‘Rsuyke mkgcgkuc’ service does not exist, the malware creates the service but does not register it in a proper way. If the self-copy setting is turned on, the malware copies itself to the path %ProgramFiles% with a random name. As the created service is not registered in a proper way and the malware does not use additional techniques such as Run Key registration, persistence is not maintained.
- **Mode #1**: Just like in Mode #0, the malware creates an abnormal ‘Rsuyke mkgcgkuc’ service. It registers the service to HKLM Run Key, enabling it to maintain persistence.
- **Mode #2**: The malware copies itself to the path %ProgramFiles%\Cccogae.exe’ and registers to ‘Rsuyke mkgcgkuc’ service. When it executes the service, it gives ‘Win7’ as the argument and executes Gh0stCringe. As the service is registered in a proper way, persistence is maintained. Additionally, if the self-copy settings is turned on, it copies itself to the path ‘%SystemDirectory%[Random].bak’.
- **Mode Windows 10**: It registers to HKCU Run Key, enabling it to maintain persistence.

이름	종류	데이터
(기본값)	REG_SZ	(값 설정 안 됨)
ConnectGroup	REG_SZ	Default
Description	REG_SZ	Ekcewq wqqemkmcsukuwyqyqc
DisplayName	REG_SZ	Yckumm yakucega
ErrorControl	REG_DWORD	0x00000001 (1)
FailureActions	REG_BINARY	80 51 01 00 00 00 00 00 00 00 00 00
ImagePath	REG_EXPAND_SZ	C:₩Program Files₩Cccogae.exe
MarkTime	REG_SZ	2022-03-07
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000110 (272)

Figure 6. Rsuyke mkgcgkuc service registered in a proper way

After the initial routine, it connects to the C&C server and communicates periodically, waiting for the attacker's command. Apart from the command perform routine, it collects the following information of the infected system on initial connection and sends it to the C&C server.

Offset	Size	Description
+0x0000	0x01	0xC8
+0x0001	0x07	"Default"
+0x0024	0x04	IP address of the infected system
+0x0028	0x04	Host name of the infected system
+0x005C	0x9C	Windows ver.
+0x00F8	0x04	Number of CPUs
+0x010C	0x04	CPU speed (GHz)
+0x0114	0x04	Network performance
+0x0118	0x04	Number of webcams
+0x011C	0x04	Wow64 availability
+0x0120	0x04	Memory capacity (MB)
+0x0124	0x04	Local disk capacity (MB)
+0x0128	0x04	Date of malware installation
+0x0159	N/A	List of installed security products
+0x01BE	N/A	Certain data area (not used)
+0x0204	0x04	Network interface speed (Mbps)
+0x0208	0x04	"V9.0"
+0x0228	0x04	Whether there were key inputs in the last 3 minutes

Table 1. Structure of data collected from the infected system

The most noticeable collected data is the list of installed security products. The malware scans the names of the currently running processes compares them to the following list, and if they match, it records the information and sends it to the C&C server.

A 000000012204	000010012204	0	BaiduSdSvc.exe
A 00000001221C	00001001221C	0	ServUDaemon.exe
A 000000012238	000010012238	0	DUB.exe
A 00000001224C	00001001224C	0	1433.exe
A 000000012260	000010012260	0	S.exe
A 000000012274	000010012274	0	pfw.exe
A 00000001228C	00001001228C	0	MPMon.exe
A 0000000122A4	0000100122A4	0	FYFireWall.exe
A 0000000122C0	0000100122C0	0	kpfwtray.exe
A 0000000122DC	0000100122DC	0	rfwmain.exe
A 0000000122E8	0000100122E8	0	Outpost Firewall
A 0000000122FC	0000100122FC	0	outpost.exe
A 000000012308	000010012308	0	Comodo
A 000000012310	000010012310	0	cpf.exe
A 000000012318	000010012318	0	Kaspersky
A 000000012324	000010012324	0	avp.exee
A 000000012330	000010012330	0	ZoneAlarm
A 00000001233C	00001001233C	0	vsmon.exe
A 000000012348	000010012348	0	F-Prot AntiVirus
A 00000001235C	00001001235C	0	F-PROT.exe
A 000000012368	000010012368	0	Avira Antivir
A 000000012378	000010012378	0	avgaurd.exe
A 000000012384	000010012384	0	Mcafee
A 00000001238C	00001001238C	0	Dr.web
A 000000012394	000010012394	0	spidernt.exe
A 0000000123A4	0000100123A4	0	AVG Anti-Virus
A 0000000123B4	0000100123B4	0	avg.exe
A 0000000123BC	0000100123BC	0	Symantec Norton
A 0000000123CC	0000100123CC	0	ccapp.exe

Figure 7. A part of the list of scanned security products

The original Gh0st RAT uses a signature string called **“Gh0st”** just like its name to communicate with the C&C server. The following is the routine that decides on the “Gh0st” string via PacketFlag in the original Gh0st RAT.

```
CClientSocket::CClientSocket()
{
        WSADATA wsaData;
        WSAStartup(MAKEWORD(2, 2), &wsaData);
        m_hEvent = CreateEvent(NULL, true, false, NULL);
        m_bIsRunning = false;
        m_Socket = INVALID_SOCKET;
        // Packet Flag;
        BYTE bPacketFlag[] = {'G', 'h', '0', 's', 't'};
        memcpy(m_bPacketFlag, bPacketFlag, sizeof(bPacketFlag));
}
```

Figure 8. Gh0st signature string

The following is the packet structure of Gh0stCringe before its encryption. The information of the infected system listed above was a 0x22C size, and additional data of a 0xF size was added in the front. The **“xy “** string (first 3 bytes 0x787920) is the signature string of Gh0stCringe.

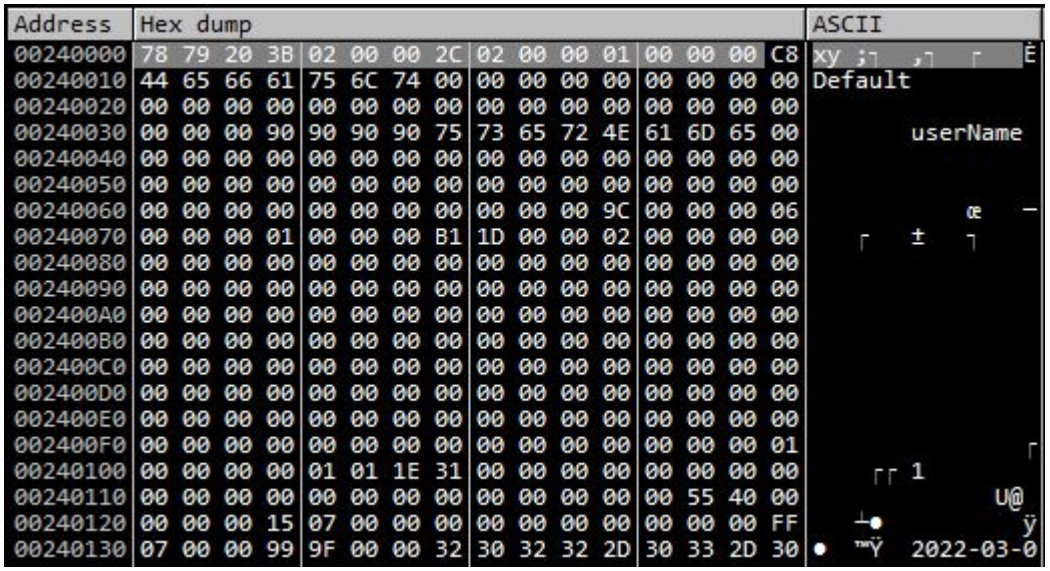

Figure 9. Packet structure of Gh0stCringe before encryption

The following table provides the details on the structure of the packet before its encryption:

Offset	Size	Data	Description
+0x00	0x03	“xy “	Signature string
+0x03	0x04	0x0000023B	Total packet size
+0x07	0x04	0x0000022C	Size of data to be sent
+0x0B	0x04	0x00000001	Hard-coded data
+0x0F	Variable	Variable	Data collected from the infected system

Table 2. Packet structure

The data that has been created goes through a self-encoding method and is ultimately encrypted. The following is the packet encryption routine and the packet created through this process to be sent to the C&C server:

```
packet_encoded = arg_buf;
for ( i = 0; i < arg_size; ++i )
  *(i + arg_buf) = (*(i + arg_buf) - 0x46) ^ 0x19;
return packet_encoded;
```

Figure 10. Packet encryption routine

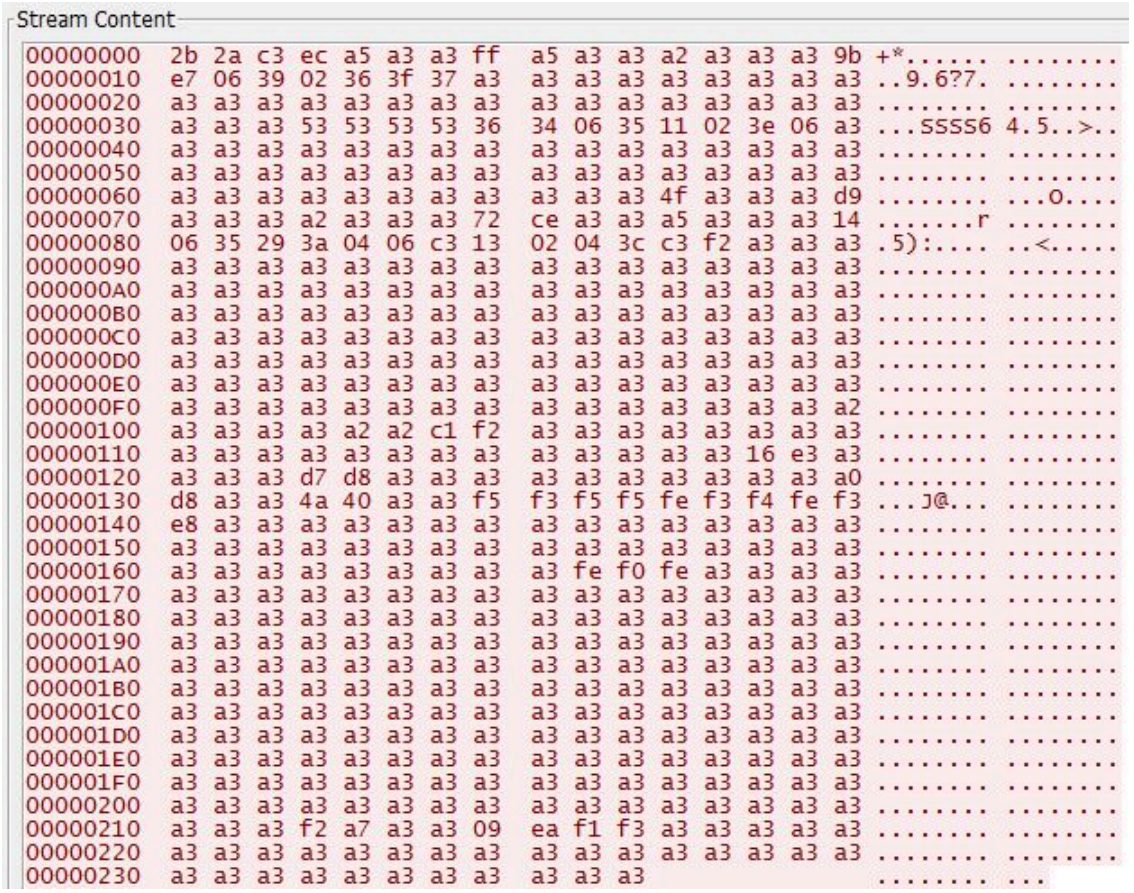

Figure 11. Encoded information of the infected system to be sent to C&C server

The following is a list of various commands that can be received from the C&C server and performed. Some of the commands of Gh0stCringe exist in the original Gh0st RAT, but there are many unique commands as well.

Performing Commands

– **Downloader**: Downloads additional payload from the C&C server and executes it. Can send specific arguments.

– **Connecting to a specific URL**: Connects to a specific website via Internet Explorer There is an option to hide the window so that the user does not notice it.

– **Destroying MBR**

Stealing Information

– **Keylogging**: Regardless of whether there is settings data or not, keylogging can be performed by the C&C server's command.

– **Stealing clipboard**: Steals the saved data to the current clipboard.

– **Collecting Tencent-related file information**: Collects the list of files in the path '%APPDATA%\Tencent\Users\', and it is assumed that it would normally include user data related to QQ Messenger.

Self-Control

– **Update**: Updates the malware

– **Uninstall**: Deletes the registered service and self-deletes using a batch file. Also deletes the keylogging data Default.key file.

– **Service control**: Changes the ‘Host’ or ‘ConnectGroup’ item for the malware service (‘Rsuyke mkgcgkuc’).
– **Event Cleanup**
– **Registering Run Key**: Registers Run Key for the path ‘C:\Program Files\Common Files\scvh0st.exe’.

System Control
– **Terminating system**
– **Rebooting NIC**

Additional Module Control
Downloads an additional module from the C&C server to memory and loads it to call the following export function. Judging by the export function, the first module appears to be a proxy-related module, and the second module appears to be a Plugin module with additional features.
– **Module #1**: Export function OpenProxy(), CloseProxy()
– **Module #2**: Export function PluginMe()

Others
– **Scanning whether a certain process is running**
– **Scanning for the existence of certain Windows**
– **Writing for a certain registry**: ‘HKLM\SYSTEM\Clore / Clore’
– **Message pop-up**

Typical attacks that target database servers (MS-SQL, MySQL servers) include brute force attacks and dictionary attacks to systems where account credentials are poorly being managed. Although it seems like these methods make up the majority of the attacks, there can be vulnerability attacks against systems where their vulnerability has not been patched.

In the case of MS-SQL servers, servers that have been installed by ERP and work-purpose solutions are being targeted by attackers in addition to the servers that are established normally. Because of this, administrators should use passwords that are difficult to guess for their accounts and change them periodically to protect the database server from brute force attacks and dictionary attacks, and maintain the latest patch to prevent vulnerability attacks. Administrators should also use security programs such as firewalls for database servers accessible from outside to restrict access of external attackers.

AhnLab’s anti-malware software, V3, detects and blocks the malware above using the aliases below.

[File Detection]
– Backdoor/Win.Gh0stRAT.C4976413 (2022.02.19.00)
– Backdoor/Win.Gh0stRAT.C4976420 (2022.02.19.00)

[References]
[1] https://twitter.com/James_inthe_box/status/1125004664041197568
[2] https://www.binarydefense.com/gh0stcringeformerly-cirenegrat/
[3] https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophos-labs-kingminer-botnet-report.pdf
[4] https://www.guardicore.com/blog/vollgar-ms-sql-servers-under-attack/

MD5

782cbc8660ff9e94e584adfcbc4cb961

9adc9644a1956dee23c63221951dd192

bd8611002e01d4f9911e85624d431eb0

Additional IOCs are available on AhnLab TIP.

URL

http[:]//172[.]86[.]127[.]224[:]8000/

http[:]//tuwu[.]meibu[.]net[:]2220/

Additional IOCs are available on AhnLab TIP.

Gain access to related IOCs and detailed analysis by subscribing to **AhnLab TIP**. For subscription details, click the banner below.

Source: https://asec.ahnlab.com/en/32572/