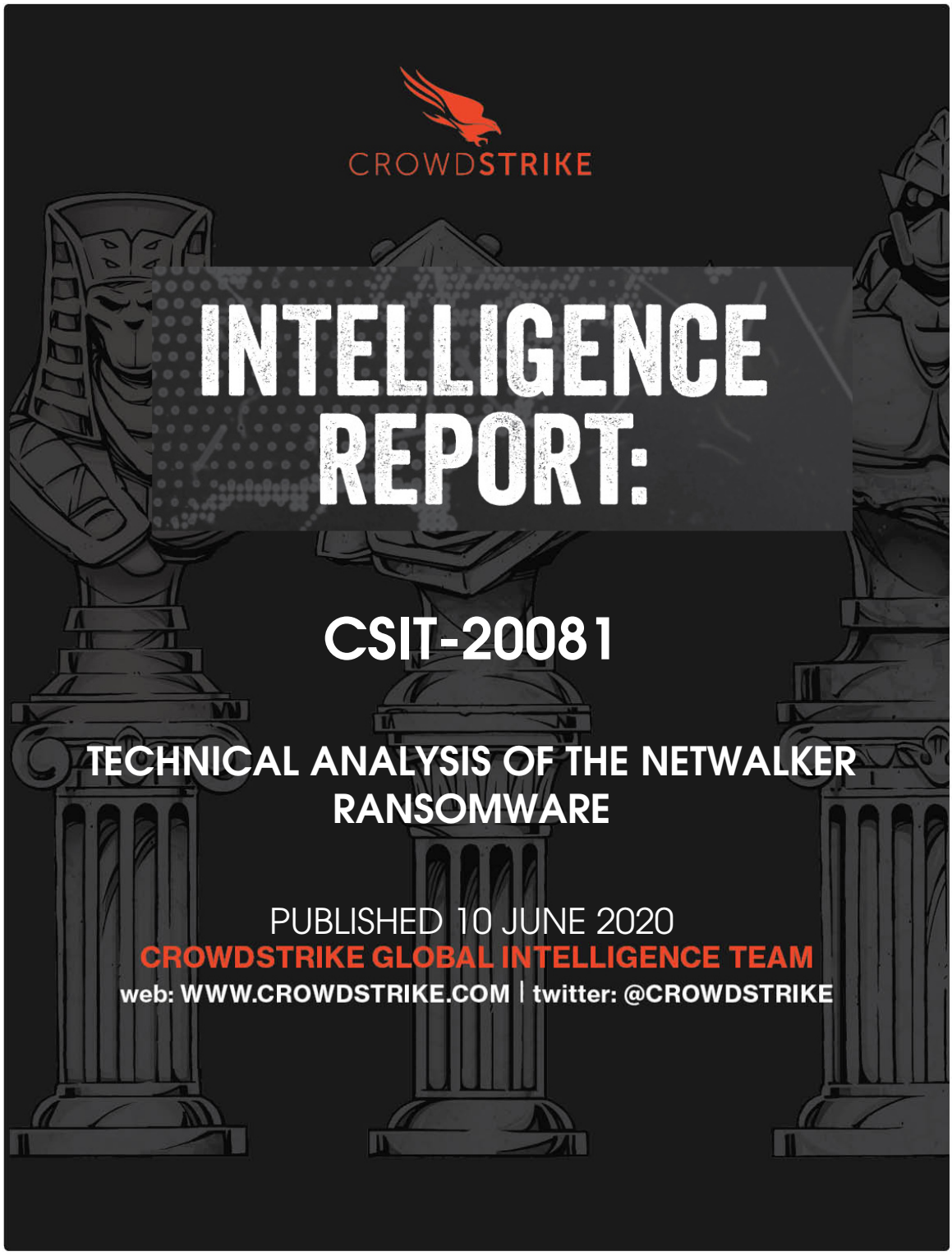

TECHNICAL ANALYSIS OF THE NETWALKER RANSOMWARE

KEY POINTS

- *NetWalker* is ransomware written in C++ and advertised as a Ransomware-as-a-Service (RaaS) on forums by a user known to be part of a group designated as CIRCUS SPIDER.
- NetWalker uses a combination of ChaCha and Elliptic Curve Cryptography (ECC) with Curve25519 to encrypt files.
- The ransomware is controlled by a JSON configuration that is stored encrypted using RC4 in the malware's resource directory named 31337.
- NetWalker encrypts files on the local system, mapped network shares and enumerates the network for additional shares, attempting to access them using the security tokens from all logged-in users on the victim's system.
- Processes that have file handles open to files that are targeted for encryption are terminated by NetWalker to encrypt as many files as possible.
- CIRCUS SPIDER initially provided victims with contact information via email, and around March 2020 the group set up a ransom portal on the TOR network.

The NetWalker ransomware is being developed and maintained by a Russian-speaking actor designated as CIRCUS SPIDER. Initially discovered in September 2019 and having a compilation timestamp dating back to 28 August 2019, NetWalker has been found to be used in Big Game Hunting (BGH)-style operations while also being distributed via spam. CIRCUS SPIDER is advertising NetWalker as being a closed-affiliate program, and verifies applicants before they are being accepted as an affiliate. The requirements range from providing proof of previous revenue in similar affiliates programs, experience in the field and what type of industry the applicant is targeting.

The ransom amount, paid in Bitcoin, is configured by the affiliates and have been observed to range between USD $1,000 in spam campaigns up to USD $3 million in BGH campaigns. Affiliates that are targeting corporations have been observed to base their ransom on the corporation's revenue, such as $100,000 per $10 million in revenue. As of 12 May 2020, CIRCUS SPIDER is threatening to leak data that was exfiltrated during the breach phase similar to TWISTED SPIDER and PINCHY SPIDER.

TECHNICAL ANALYSIS

Initialization and Configuration

NetWalker does not make use of any anti-debugging or anti-sandbox techniques. The only anti-analysis technique is the usage of dynamic imports. NetWalker resolves all the required imports by hash using 32-bit Cycle Redundancy Check (CRC32). If NetWalker is unable to load any of the required libraries, it sleeps for 1 millisecond and exits. However, if the malware fails to resolve a function (which is unlikely to occur unless there was a programming error), it continues execution until the required function is called, which will cause an exception and crash the program.

NetWalker does not run on systems that are configured with the Russian, Ukrainian, Belarusian or Kazakh keyboard layouts. This check is commonly observed in malware, likely to reduce the risk of prosecution by law enforcements in the aforementioned countries. It is worth noting that malware authors usually include detection for more Eastern European countries. Early versions of NetWalker did not implement language detection, making it possible to encrypt files on, for example, systems set up for Russian-language users.

There is no persistence mechanism, which means that if NetWalker process is terminated, it will not restart. Therefore, the victim or operator would have to restart the process.

Configuration

The malware stores its configuration in the resource directory `1337` and with the name `31337`. The configuration is stored encrypted using RC4 where the key length and the key itself is the first bytes in the resource data as shown in Figure 1.

```
00000000  04 00 00 00 2B 7E 43 2D CF E6 0E 9C 5C 7F 62 CA
00000010  B8 36 35 21 CC FD B5 91 81 6B 74 D0 08 02 B0 34
00000020  E0 80 69 0F DF 87 35 D0 84 C9 4D 30 4F 80 D5 E7
00000030  D4 ED C6 1A 2E 74 77 C2 D3 A5 DC 24 B0 A1 D9 A9
00000040  10 7A 5D 56 1F 12 03 37 DF 72 3C D1 6D F1 D3 50
<...truncated...>

RC4 key length
RC4 key
Encrypted configuration
```

Figure 1. Encrypted NetWalker Configuration

NetWalker decrypts and parses the configuration that is in the JavaScript Object Notation (JSON) format. A truncated example is shown in Figure 2. A non-truncated configuration is included in *Appendix A* and a reference table describing all of the known JSON keys is included in *Appendix B*.

```
{
    "lfile": "{id}-Readme.txt",
    "spsz": 15360,
    "lend": "SGkhDQpZb3VyIGZpb<...snip...>gaW5wdXQgZm9ybToNCg0Ke2NvZGV9",
    "namesz": 8,
    "thr": 1500,
    "mpk": <redacted>,
    "pers": false,
    "unlocker": {
       "ignore": {
          "pspath": ["*:\\windows*", <...snip...> "*\\Program
File*\\Fortinet"],
          "use": true,
          "prc": ["psexec.exe", <...snip...> "FCHelper64.exe"]
        },
        "use": true
       },
    "idsz": 6,
```

```
    "mode": 0,
    "net": {
       "ignore": {
          "use": true,
          "disk": true,
          "share": ["ipc$", "admin$"]
       },
       "use": true
    },
    "kill": {
       "use": true,
       "svcwait": 0,
       "svc": ["Lotus*", "veeam*", <...snip...> "acrsch2svc*"],
       "prc": ["nslsvice.exe", "pg*", <...snip...> "agntsvc.exe"],
       "task": ["reboot", "restart", "shutdown", "logoff", "back"]
      },
    "white": {
       "path": ["*system volume information",
<...snip...>"\\\\*\\users\\*\\appdata\\*\\microsoft"],
       "ext": ["msp", "exe", <...snip...> "themepack"],
       "file": ["ntuser.dat*", <...snip...> "bootfont.bin"]
    },
    "onion2":
"rnfdsgm6wb6j6su5txkekw4u4y47kp2eatvu7d6xhyn5cs4lt4pdrqqd[.]onion",
    "onion1":
"pb36hu4spl6cyjdfhing7h3pw6dhpk32ifemawkujj4gp33ejzdq3did[.]onion"
 }
```

Figure 2. NetWalker Configuration

Generate Infection Identifier

Having resolved all the required functions as well as decrypted and parsed the configuration, NetWalker gathers information about the system and generates a unique infection identifier (ID) for the victim's system.

NetWalker uses HMAC-SHA256 to calculate a hash from the gathered system information. The pseudo-code shown in Figure 3 outlines the process of how NetWalker generates an infection ID. The infection ID is used as the file extension for encrypted files, for the ransom note name and a registry key described in the *Key Recovery Blob Generation* section. The length of the infection ID is defined by the `namesz` (for the encrypted file extension) and `idsz` (for the ransom note file name) entries in the NetWalker configuration.

```
from Crypto.Cipher import XOR
import hashlib
import base64

def get_infection_id_from_hmac(cfg):

    cfg_pub_key = base64_decode(cfg.mpk)
    xored_pub_key = XOR.new('\x36').encrypt(cfg_pub_key)

    s = hashlib.sha256(xored_pub_key).digest()
```

```
    s.update(GetComputerNameExW())
    hw_profile = GetCurrentHwProfileW()
    s.update(hw_profile.GUID)

    # Undocked profile / Docked profile
    s.update(hw_profile.docked_profile_string)
    s256_hash = s.final()

    xored_hash = XOR.new('\x5C').encrypt(s256_hash)

    final_hash = hashlib.sha256(xored_hash + s256_hash).hexdigest()

    return final_hash[:cfg.namesz]
```

Figure 3. NetWalker Infection Identifier HMAC Pseudo-Code

UAC Bypass

NetWalker attempts to bypass Windows User Access Control (UAC). This feature is controlled in the NetWalker configuration using the JSON key `pers`. If enabled, NetWalker determines if the victim's system is older than Windows 8.1. If it is not, no bypass attempt is made. If the victim system is running Windows 8.1 or newer, NetWalker sets a registry key `HKCU\Software\Classes\exefile\shell\command\open` with the value being the file path to the NetWalker executable. This path is read when the Windows Activation Client (`slui.exe`) is executed and the file pointed by the registry key value is consequently executed by the Windows Activation Client. As the `slui.exe` is running with high integrity, NetWalker is also executed with high integrity. This technique is patched in the latest version of Windows 10.

However, if the operating system is prior to Windows 7, NetWalker uses the registry key path `HKCU\Software\Classes\mscfile\shell\command\open`. Instead of the Windows Activation Client, NetWalker executes the Windows Event Viewer (`eventvwr.exe`). Similar to the Windows Activation Client, the Windows Event Viewer reads a file path from the registry key and executes it with the same integrity as itself, resulting in elevated privileges. This technique works is exploitable from Windows 7 up to Windows 10 Creators Update build 15007.

NetWalker removes the registry key after attempting the bypass, and if the malware was successful in elevating its privileges, the malware terminates its old instance. Otherwise, the malware continues its execution with the current integrity.

NetWalker Operation

NetWalker attempts to encrypt as many files as possible by killing not only tasks, services, and processes that are blacklisted, but also target processes that have open file handles to targeted files. This section describes the execution of the NetWalker ransomware.

Having initialized, NetWalker acquires the `SeImpersonatePrivilege` and `SeDebugPrivilege` to itself. These privileges are required in order to access other processes and to be able to impersonate the logged in user when accessing remote file shares in order to encrypt files.

If the UAC bypass feature in enabled in the NetWalker configuration, the malware creates a mutex that corresponds to the infection ID to ensure that only one instance of NetWalker is running at once. If the UAC bypass feature is disabled, it allows multiple instances of the malware to be run at the same time on the victim's system.

Before encrypting any files, NetWalker deletes the shadow copies from the system by executing `C:\Windows\system32\vssadmin.exe delete shadows /all /quiet`. The Windows system path is retrieved by calling `GetSystemDirectoryW`.

NetWalker includes functionality to terminate services, tasks and processes. This is configurable in the malware configuration key `kill`. An example of a configuration block is shown in Figure 4.

```
"kill": {
      "use": true,
      "task": ["reboot", "restart", "shutdown", "logoff", "back"],
      "svcwait": 0,
      "svc": ["Lotus*", "veeam*", "stc_endpt_svc", "acrsch2svc*"],
      "prc": ["nslsvice.exe", "pg*", "infopath.exe", "agntsvc.exe"]
}
```

Figure 4. NetWalker Termination of Tasks, Processes and Services

NetWalker creates an individual thread to handle each of these tasks. However, only the thread tasked with terminating services exits after it has performed its tasks. The two remaining threads are run in a continuous loop and sleep 10 milliseconds (process termination thread) and 60 seconds (task termination thread) in order to prevent tasks and processes from interfering with the file encryption process.

In order to ensure that services are given enough time to terminate, NetWalker uses the configuration key `kill.svcwait` to sleep the number of seconds configured before continuing execution.

NetWalker uses multiple threads to simultaneously encrypt files across multiple drives and directories. The maximum number of threads used to encrypt files at any one time is determined by the configuration key `thr` and has been observed to range between 1,000 and 1,500, depending on the operator's choice.

In order to keep track of the running threads, NetWalker spawns a thread which continuously tracks all the running threads in order to allow new threads to execute when a previous thread has finished.

Before starting to encrypt files, NetWalker iterates over all the running process on the victim's system and calculates a CRC32 checksum of the process name in order to compare it to `BE037055` (`explorer.exe`). NetWalker gathers the user security tokens from every running instance of `explorer.exe` in an attempt to authenticate to network shares. This is described in more detail in *Encrypt Remote Files via Token Impersonation*.

NetWalker goes through three encryption phases that target files depending on their storage location and if the files are currently in use.

Encrypt Local and Remote Files on Mapped Network Shares

First, NetWalker retrieves all local drives and mapped network drives by calling `GetLogicalDriveStringsW`. NetWalker creates a new encryption thread for each drive and the threads iterates the contents of the drive and creates a new thread for each file and sub-directory, this is to encrypt as many files as possible, as fast as possible.

NetWalker operates in a blacklist fashion, in that it encrypts every file that does not match any of the entries in the blacklist. The malware's blacklist includes file paths, extensions and filenames that are checked before attempting to encrypt a file. In the event that NetWalker is unable to open a file handle to a given file due to it being locked by another process, NetWalker saves the file path to a list that is used at a later stage after the initial encryption process has been completed. The purpose and usage of this list is described in the *Unlock and Encrypt Locked Files* section.

Encrypt Remote Files via Token Impersonation

In the event that NetWalker is not able to access a mapped network share, the malware attempts to access it using the security tokens previously gathered from the `explorer.exe` processes. This enables NetWalker to encrypt files that the currently logged-in user does not have permission to modify. In order to impersonate a logged-on user using the gathered tokens, NetWalker uses the Windows API `ImpersonateLoggedOnUser`.

NetWalker is able to discover and mount new shares if the `net` feature is enabled in the configuration (`net.use`). NetWalker calls `WNetOpenEnumW` with the `RESOURCE_GLOBALNET` flag set that enumerates resources across the network. In an attempt to gain access to each of the enumerated resources, NetWalker uses the security tokens that it previously gathered. This feature not only puts mapped file shares on a victim's system at risk, but also shares that the victim and logged-in users have access to.

Unlock and Encrypt Locked Files

Terminating blacklisted processes is a common technique used by ransomware families in order to encrypt as many files as possible, such as databases. However, NetWalker takes this technique to the next level. During the encryption process, NetWalker maintains a list of file paths that it was unable to open. After completing the encryption of files on a victim's system and network shares, NetWalker iterates over the list of locked files and looks up the process that has an open handle to each of the files. If the process is not blacklisted in the `unlocker.ignore.prc` or resides in a file path listed in `unlocker.ignore.pspath`, NetWalker attempts to gain access to the process and terminate it, after which NetWalker launches an encryption thread to encrypt the unlocked file.

An example of the blacklisted processes and process paths are listed in Table 1, taken from the analyzed sample's configuration.

STRING	TYPE	DESCRIPTION
`*:\\windows*`	Process path	Windows executables
`*:\\winnt*`	Process path	Windows executables
`*:\\program file*\\vmwar*`	Process path	Processes related to VMware
`*\\Program File*\\Fortinet`	Process path	Processes related to Fortinet endpoint protection

psexec.exe	Process	PsExec Windows utility
system	Process	Windows System process
forti*.exe	Process	Process related to Fortinet endpoint protection
fmon.exe	Process	Fortinet real-time monitor
fcaptmon.exe	Process	Fortinet sandbox agent
FCHelper64.exe	Process	Fortinet Helper

Table 1. NetWalker Process and Path Blacklist

Cryptography

NetWalker uses a combination of symmetric (ChaCha) and asymmetric (Curve25519) algorithms to encrypt files and protect the host specific private key while being stored in the Windows registry. Every file is encrypted with its own ChaCha key and nonce. The ChaCha key material and Curve25519 key-pair generation is further described in this section.

Generating Host-Specific Curve25519 Key Pair

NetWalker generates a Curve25519 private key (`priv0`) by calling `RtlRandomEx` (`RtlRandom` for Windows XP and earlier) 32 times, reading one byte after each call. The malware uses the current system time as a seed, which, when `RtlRandom` is called, can be brute forced to recover the Curve25519 private key given the timestamp of the initial infection.

However, with the introduction of `RtlRandomEx` in Windows Vista that improves the generation of pseudo-random values, it is no longer possible to replicate random numbers given only a static seed. Instead of relying on a single seed, `RtlRandomEx` relies on the input seed and a 256-byte lookup table that is initialized when `ntdll.dll` is loaded, i.e., when the NetWalker process is first launched. This lookup table is populated by calling `RtlRandomEx`, using the process cookie as a seed. The process cookie is according to the reactOS source code[1] a product of XORing multiple values, which by themselves are not easily predictable (such as the processors `KeSystemCalls`, `InteruptTime` and system time), let alone the result. Thus, attacking the cryptographical implementation in NetWalker is likely not feasible in a reasonable amount of time.

NetWalker uses the private key (`priv0`) to derive a public key (`pub0`) that the malware uses to encrypt the shared secrets derived to encrypt files. The shared secret derivation is described in the *Deriving Curve25519 Shared Secrets* section.

Deriving Curve25519 Shared Secrets

NetWalker derives a new shared secret each time that the malware encrypts data, providing an individual key for every encryption call, including file encryption. The input for the derivation is a Curve25519 public key, either the operator supplied key from the NetWalker configuration key `mpk` or `pub0` (described in the previous section).

1 https[:]//github[.]com/mirror/reactos/blob/c6d2b35ffc91e09f50dfb214ea58237509329d6b/reactos/ntoskrnl/ps/query.c#L793

To encrypt a file, NetWalker generates a new Curve25519 key pair (`priv1` and `pub1`) using the same schema as for generating the host-specific key pair, however, the newly generated key pair is only used for one encryption operation (e.g. a single file), equivalent to a session key pair. NetWalker uses the generated private key (`priv1`) together with the host-specific public key (`pub0`) key to derive a 32-byte shared secret, used as the ChaCha encryption key.

NetWalker hashes the shared secret using SHA256 and increments the first byte of the hash by one. The first 8 bytes of the modified hash is used as the ChaCha nonce. The reason for modifying only the first byte of the SHA256 hash may be an attempt to throw off analysts.

Key Recovery Blob Generation

NetWalker uses the infection ID to store information in the Windows registry. The ransomware initially attempts to read the data from two different registry locations; `HKLM\Software\<INFECTION ID>` and `HKCU\Software\<INFECTION ID>`. These registry keys, if previously created by an earlier NetWalker instance, hold the host-specific Curve25519 key pair (public key; `pub0` and private key; `priv0`). The private key `priv0` (which is required to decrypt files) is encrypted with ChaCha using a key derived from the operator supplied public key (`mpk`) and a public key `pubr` derived from a generated key pair (`pubr`, `privr`). The encryption schema is described in the *Encryption and Hashing* section.

If none of the registry keys exists, NetWalker generates a new host-specific Curve25519 key pair as previously described. An example of a key recovery blob stored in the registry is shown in Figure 4. The `pub0` key is written to the registry in order for NetWalker to restart and encrypting files using the same host-specific public key.

```
000000  60 7F 1E 52 8E AC E9 7D 72 0D D6 AB 1D 2D DE A8  `..R.¬é}r.Ö«.-Þ¨
000010  59 55 55 E9 52 EF 86 22 95 EA E5 7C 9D E0 45 1C  YUUéRï."•êå|.àE.
000020  23 4E 73 53 01 AF 4B A6 77 83 67 56 FA D4 F5 E4  #NsS.¯K¦w.gVúÔõä
000030  1C A4 52 36 32 9A 95 60 BF 43 14 A1 09 43 D7 17  .¤R62..`¿C.¡.C×.
000040  D2 54 F3 0A B8 2D AD 56 BD 63 CA 27 C8 FB BB 88  ÒTó.¸-.V½cÊ'Èû».
000050  84 9C 55 1F C6 AD 81 62 3C 5F 9B 29 FD 0E 6C AA  ..U.Æ..b<_.)ý.lª
000060  56 63 6B 9F DA 5A 59 68 23 BB 40 15 F6 61 41 4F  Vck.ÚZYh#»@.öaAO
000070  D0 37 54 AF E7 25 EC 5C 89 70 93 EB 76 4A D5 4A  Ð7T¯ç%ì\.p.ëvJÕJ
00008C  2F 71 A2 2E 6E 08 2D A5 D3 7D 92 8C              /q¢.n.-¥Ó}..

Host-specific public Curve25519 key (pub0)
CRC32 sum of pub0
One-time public Curve25519 key (pubr)
ChaCha nonce
HMAC-SHA256 from ChaCha encryption
ChaCha Encrypted data (priv0)
```

Figure 4. Example NetWalker Key Recovery Blob Stored in the Registry

After the key recovery blob is written to the Windows registry, NetWalker wipes the `priv0` key from memory by overwriting it with null bytes to prevent recovery of the `priv0` from a memory dump of the NetWalker process. The `priv0` key can only be decrypted with a key derived using the actor-controlled private key, corresponding to the `mpk` public key, and the one-time public key stored in the registry `pubr`.

Ransom-Note Generation

The key recovery blob is included in the ransom note where it is part of what is referred to as `User code`. The `User code` is a text blob that the victim needs to submit to the operators in order to authenticate to the ransom portal. The ransom portal is described in the *Ransom portal* section. The `User code` contains, in addition to the data stored in the Windows registry (excluding `pub0`), the length of the infection ID and the infection ID itself, the length and template string for the ransom note filename. The operators use this information to provide the victim with a decrypter, tailored to the victim's environment. An example of the unencrypted `User code` is illustrated in Figure 5.

```
002D8A00  23 4E 73 53 01 AF 4B A6 77 83 67 56 FA D4 F5 E4  #NsS.¯K¦w.gVúÔõä
002D8A10  1C A4 52 36 32 9A 95 60 BF 43 14 A1 09 43 D7 17  .¤R62..`¿C.¡.C×.
002D8A20  D2 54 F3 0A B8 2D AD 56 BD 63 CA 27 C8 FB BB 88  ÒTó.¸-.V½cÊ'Èû».
002D8A30  84 9C 55 1F C6 AD 81 62 3C 5F 9B 29 FD 0E 6C AA  ..U.Æ..b<_.)ý.lª
002D8A40  56 63 6B 9F DA 5A 59 68 23 BB 40 15 F6 61 41 4F  Vck.ÚZYh#»@.öaAO
002D8A50  D0 37 54 AF E7 25 EC 5C 89 70 93 EB 76 4A D5 4A  Ð7T¯ç%ì\.p.ëvJÕJ
002D8A60  2F 71 A2 2E 6E 08 2D A5 D3 7D 92 8C 06 00 00 00  /q¢.n.-¥Ó}......
002D8A70  65 37 36 37 34 38 0F 00 00 00 7B 69 64 7D 2D 52  e76748....{id}-R
002D8A80  65 61 64 6D 65 2E 74 78 74                       eadme.txt

Key recovery blob stored in the Windows registry (excl. pub0)
Length of the infection ID (from the NetWalker configuration field idsz)
Infection ID
Length of ransom note file name template string
Ransom note file name template string
```

Figure 5. NetWalker Unencrypted User Code Inner Layer

The `User code` blob is encrypted using the same encryption scheme as when encrypting files and produces the blob shown in Figure 6.

```
002DBC08  DD A3 E6 0A 15 35 1B 23 77 9E C8 FF CF 89 FA 8B  Ý£æ..5.#w.ÈÿÏ.ú.
002DBC18  06 4F 6B DC 5B B3 45 91 0B 17 C7 5E A9 D5 A5 91  .OkÜ[³E...Ç^©Õ¥.
002DBC28  F7 9E 2D B3 F2 78 02 1A F5 03 4E 63 6D 6F 57 61  ÷.-³òx..õ.NcmoWa
002DBC38  AB C5 4E 2F F7 50 56 F7 40 56 13 05 65 10 12 BA  «ÅN/÷PV÷@V..e..º
002DBC48  2D AB A7 31 4A 86 36 07 XX XX XX XX 23 4E 73 53  -«§1J.6.....#NsS
002DBC58  71 2A 22 0D 70 5B 0D 4B D1 1D FD DD 38 5C 1E 54  q*".p[.KÑ.ýÝ8\.T
002DBC68  DD 57 03 0E 65 81 4F F5 6D 63 06 A4 F2 1F 1E 73  ÝW..e.Oõmc.¤ò..s
002DBC78  01 0F 19 67 91 55 35 D0 82 00 E9 7B 0C F1 FA 08  ...g.U5Ð..é{.ñú.
002DBC88  02 D8 09 59 AC 77 66 78 0A 68 08 C8 CD EC 3F 9D  .Ø.Y¬wfx.h.ÈÍì?.
002DBC98  CA A7 B6 B8 37 B2 5F 76 CF 3D 97 F1 58 48 CF 8C  Ê§¶¸7²_vÏ=.ñXHÏ.
002DBCA8  B9 19 12 93 C9 ED FC 88 39 E5 B7 48 16 58 79 B7  ¹...Éíü.9å·H.Xy·
002DBCB8  F3 24 61 F9 D9 94 32 5C 72 B8 07 3C 08 5C 66 66  ó$aùÙ.2\r¸.<.\ff
002DBCC8  9D 1C 44 6C F8 35 AE 04 FD 56 6A E6 D9 4E EC A8  ..Dlø5®.ýVjæÙNì¨
002DBCD8  4C AE 04 E6 EB 85 46 FC 57                       L®.æë.FüW

ChaCha nonce
HMAC-SHA256 Hash from the ChaCha Encryption
One-time public key (pub1)
CRC32 of the NetWalker configuration public key (campaign-specific censored)
CRC32 of the host-specific public key (pub0)
Encrypted User Code (Inner Layer)
```

Figure 6. NetWalker Encrypted User Code

NetWalker encodes the encrypted `User code` using Base64 and writes it at the end of the ransom note (as highlighted in Figure 7) that it drops in every directory after encrypting a file.

```
Hi!
Your files are encrypted by Netwalker.
All encrypted files for this computer has extension: .e76748

--
If for some reason you read this text before the encryption ended,
this can be understood by the fact that the computer slows down,
and your heart rate has increased due to the ability to turn it off,
then we recommend that you move away from the computer and accept that you
have been compromised.
Rebooting/shutdown will cause you to lose files without the possibility of
recovery.

--
Our  encryption algorithms are very strong and your files are very well
protected,
the only way to get your files back is to cooperate with us and get the
decrypter program.

Do not try to recover your files without a decrypter program, you may damage
them and then they will be impossible to recover.

For us this is just business and to prove to you our seriousness, we will
decrypt you one file for free.
Just open our website, upload the encrypted file and get the decrypted file
for free.

--

Steps to get access on our website:

1.Download and install tor-browser: https://torproject[.]org/

2.Open our website:
pb36hu4spl6cyjdfhing7h3pw6dhpk32ifemawkujj4gp33ejzdq3did[.]onion
If the website is not available, open another one:
rnfdsgm6wb6j6su5txkekw4u4y47kp2eatvu7d6xhyn5cs4lt4pdrqqd[.]onion

3.Put your personal code in the input form:

{VICTIM INFORMATION REDACTED}
```

Figure 7. NetWalker Ransom Note

Encryption and Hashing

In addition to encrypting the data with ChaCha, NetWalker uses HMAC-SHA256 to ensure both integrity and authenticity of the encrypted data. NetWalker does not use a Windows API for the HMAC calculation but has implemented it by hand. A Python implementation of the encryption function is shown in Figure 8.

```
import hashlib
from chacha import ChaCha # See footnote[2]
from Crypto.Cipher import XOR

def encrypt_data(key, iv, in_data):

    cc = ChaCha(key, iv[:8] rounds=8)

    x_key = XOR.new('\x36').encrypt(key)

    digest = hashlib.sha256(key + x_key + in_data).digest()

    enc_data = cc.encrypt(in_data)

    x_key = XOR.new('\x5C').encrypt(key)
    final = hashlib.sha256(digest + enc_data + x_key + digest).hexdigest()

    return final, enc_data
```

Figure 8. NetWalker Encryption Implementation

Encryption of Files

NetWalker supports three different encryption modes that determine how the ransomware encrypts the targeted file. The operator uses the configuration key `mode` that can be either `0`, `1` or `2`. When the modes `1` or `2` are specified in the configuration, they instruct NetWalker to only use one encryption mode while `0` is dynamic. The modes are described in Table 2.

MODE	DESCRIPTION
0	Encrypt three chunks of the file, the chunk size is defined by the configuration key `spsz`.
1	Encrypt one chunk of the file at the start of the file. The chunk size is defined by `spsz`.
2	Encrypt the full file.

Table 2. NetWalker Encryption Modes

Before NetWalker attempts to open a file for encryption, it sets the `FILE_ATTRIBUTE_ARCHIVE` attribute of the file. This attribute, according to the Microsoft documentation is designed to be used to mark a file for backup. After a file has been backed up, the attribute is cleared. However, it is unknown if this flag is effectively used by any backup software.

Having opened a file, NetWalker reads the last four bytes of the file and compares them to the CRC32 checksum of the public key (`mpk`) in the NetWalker configuration. NetWalker uses this marker to indicate that a file is already encrypted.

If the encryption mode is configured as 0, NetWalker determines how to encrypt the file depending on the file size. However, if the file size is less 239 bytes, which is a limit that is hardcoded in the malware, NetWalker ignores the file and continues with the next file. The determination of the encryption mode based on the file size is described in Table 3.

[2] https[:]//www.seanet[.]com/~bugbee/crypto/chacha/chacha.py

FILE SIZE	DESCRIPTION
`File size > spsz * 5`	Encryption mode 0
`spz <= File size <= spsz * 5`	Encryption mode 1
`239 <= File size < spsz`	Encryption mode 2

Table 3. NetWalker Encryption Modes

In encryption mode `0`, NetWalker encrypts the start, middle and end of a file. In order to determine the offset of the middle chunk, NetWalker uses the formula shown in Figure 9.

```
offset = (((file_size / 2) - 1) / spsz) * spsz)
```

Figure 9. NetWalker Chunk Offset Calculation

The malware saves the original file name information in the format shown in Figure 10. NetWalker encrypts the file information with ChaCha using the same key material that was used to encrypt the file contents.

```
00601D70  06 00 00 00 65 37 36 37 34 38 0B 00 00 00 74 00  ....e76748....t.
00601D80  65 00 73 00 74 00 69 00 6E 00 67 00 2E 00 74 00  e.s.t.i.n.g...t.
00601D90  78 00 74 00                                      x.t.

Infection ID Length
Infection ID
Original File Name Length
Original File Name
```

Figure 10. NetWalker Unencrypted Original File Information of Encrypted File (Inner Layer)

In addition to original file information, NetWalker appends additional fields as illustrated in Figure 11. These additional fields include all the information needed by the operators to be able to decrypt the file successfully.

```
0060BDA8  8E 77 1A 44 A2 75 A8 9A 56 E5 A0 79 21 7E 57 CF  .w.D¢u¨.Vå y!~WÏ
0060BDB8  52 BE 67 26 0C 51 4C ED 8C EA F9 BB F7 F7 3F 97  R¾g&.QLí.êù»÷÷?.
0060BDC8  1E EE 10 2A 24 00 00 00 A7 D3 93 56 58 78 81 62  .î.*$...§Ó.VXx.b
0060BDD8  32 23 3D CA D3 58 1D ED 95 B3 2F 02 FB 00 65 65  2#=ÊÓX.í.³/.û.ee
0060BDE8  DD B1 83 63 C5 CF A1 93 83 0B C0 D0 EA D2 10 6C  Ý±.cÅÏ¡...ÀÐêÒ.l
0060BDF8  A3 80 90 61 09 6F BE 23 C5 79 5E 15 3D 15 9C 34  £..a.o¾#Åy^.=..4
0060BE08  0B 9E BB 13 10 66 D5 F2 83 0B C0 D0 EA D2 10 6C  ..»..fÕò..ÀÐêÒ.l
0060BE18  A3 80 90 61 09 6F BE 23 C5 79 5E 15 3D 15 9C 34  £..a.o¾#Åy^.=..4
0060BE28  0B 9E BB 13 10 66 D5 F2 83 0B C0 D0 EA D2 10 6C  ..»..fÕò..ÀÐêÒ.l
0060BE38  A3 80 90 61 09 6F BE 23 C5 79 5E 15 3D 15 9C 34  £..a.o¾#Åy^.=..4
0060BE48  0B 9E BB 13 10 66 D5 F2 C4 C7 A4 D9 4E A0 BE 76  ..»..fÕòÄÇ¤ÙN ¾v
0060BE58  00 00 00 00 00 3C 00 00 23 4E 73 53 01 AF 4B A6  .....<..#NsS.¯K¦
0060BE68  77 83 67 56 FA D4 F5 E4 1C A4 52 36 32 9A 95 60  w.gVúÔõä.¤R62..`
0060BE78  BF 43 14 A1 09 43 D7 17 D2 54 F3 0A B8 2D AD 56  ¿C.¡.C×.ÒTó.¸-.V
0060BE88  BD 63 CA 27 C8 FB BB 88 84 9C 55 1F C6 AD 81 62  ½cÊ'Èû»...U.Æ..b
0060BE98  3C 5F 9B 29 FD 0E 6C AA 56 63 6B 9F DA 5A 59 68  <_.)ý.lªVck.ÚZYh
0060BEA8  23 BB 40 15 F6 61 41 4F D0 37 54 AF E7 25 EC 5C  #»@.öaAOÐ7T¯ç%ì\
0060BEB8  89 70 93 EB 76 4A D5 4A 2F 71 A2 2E 6E 08 2D A5  .p.ëvJÕJ/q¢.n.-¥
```

```
0060BEC8  D3 7D 92 8C 83 2F 4F 14 5F 02 05 93 90 C0 9A 32  Ó}.../O._....À.2
0060BED8  78 43 3D 64 C9 D7 FA 52 EC 87 E4 B1 F4 1E 1B 24  xC=dÉ×úRì.ä±ô..$
0060BEE8  3B 9E AA 16 XX XX XX XX                          ;.ª.....

Original File Information (Inner Layer, Encrypted)
Size of Encrypted File Information
HMAC-SHA256 Hash of Encrypted File Information
HMAC-SHA256 Hashes of the Three Encrypted Chunks
ChaCha Nonce
Encryption Mode
Chunk size (spsz)
Key Recovery Blob
One-time Curve25519 public key (pub1)
CRC32 of the configuration public key (mpk, campaign-specific censored)
```

Figure 11. NetWalker Encrypted File Footer

The `Key Recovery Blob` contains the information needed by the operators to decrypt the host-specific private key in order to decrypt the file contents.

NetWalker appends the file footer to the encrypted file and restores the `modified` timestamp of the encrypted file to what it was before the file was encrypted as well as changing the file extension to the infection ID.

Ransom Portal

CIRCUS SPIDER has set up a ransom portal accessible through the Tor network on `rnfdsgm6wb6j6su5txkekw4u4y47kp2eatvu7d6xhyn5cs4lt4pdrqqd[.]onion` and `pb36hu4spl6cyjdfhing7h3pw6dhpk32ifemawkujj4gp33ejzdq3did[.]onion`. Upon visiting the ransom portal, the victim is asked to submit the `User code` from the ransom note. Alternatively, the victim can supply a `User key` as shown in Figure 12. However, this key is only generated once the victim has activated the portal as illustrated in Figure 13.

Figure 12. NetWalker Ransom Portal Landing Page

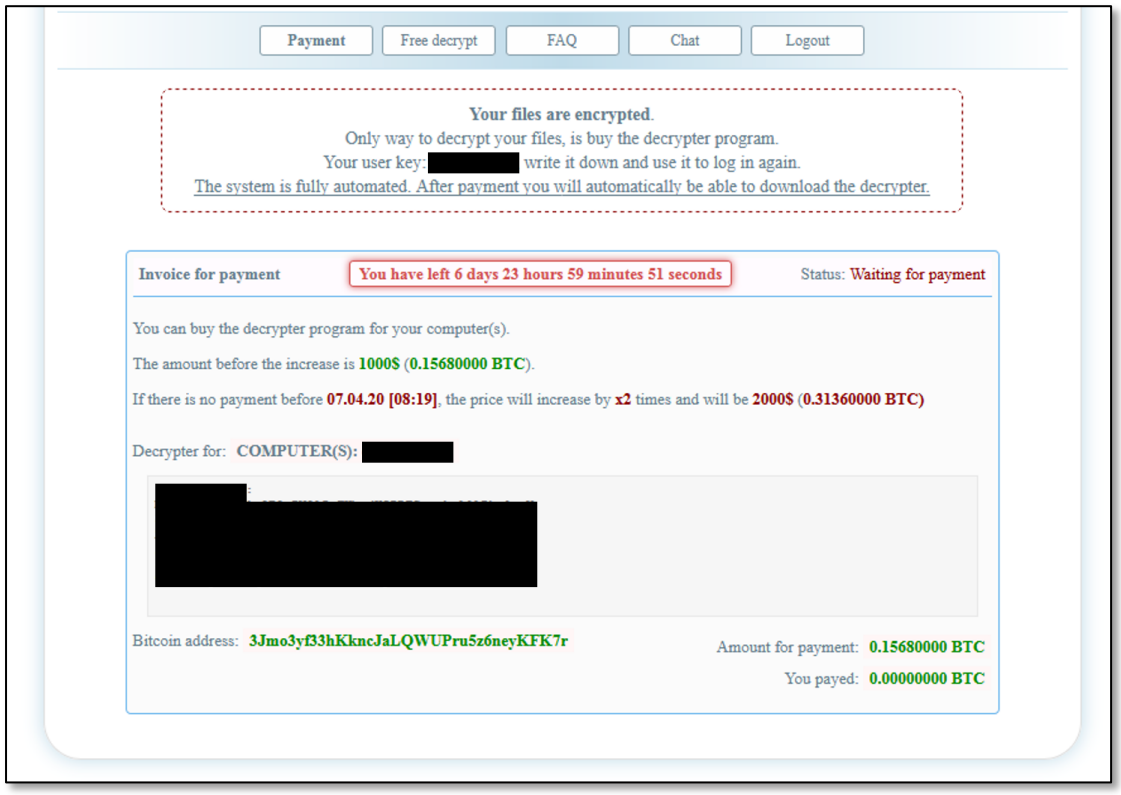

Figure 13. NetWalker Ransom Portal Main Page

The main portal page shows a countdown, ransom amount and the Bitcoin wallet address to where the ransom is expected to be paid. The victim has seven days from the time that the portal is activated before the ransom demand is doubled.

CIRCUS SPIDER offers free decryption of three files that are either images or Microsoft Word documents as proof that decryption of the files are indeed possible before paying any ransom as illustrated in Figure 14.

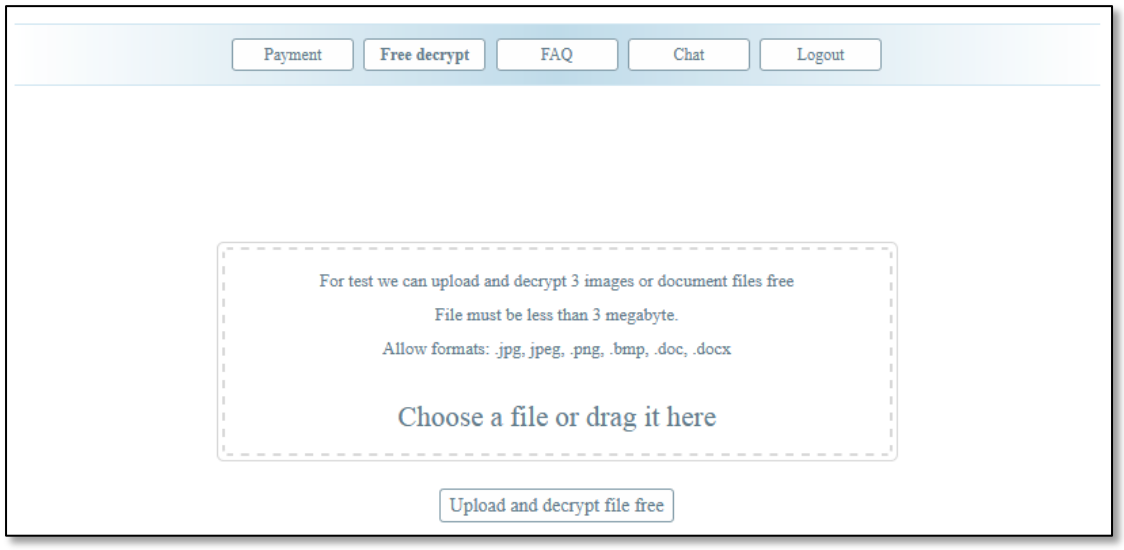

Figure 14. NetWalker Ransom Portal; Free Decryption

CrowdStrike Intelligence has been able to verify that the operators are able to decrypt encrypted files using the free decryption tool. Figure 15 shows NetWalker's frequently asked questions (FAQ) page on the ransom portal.

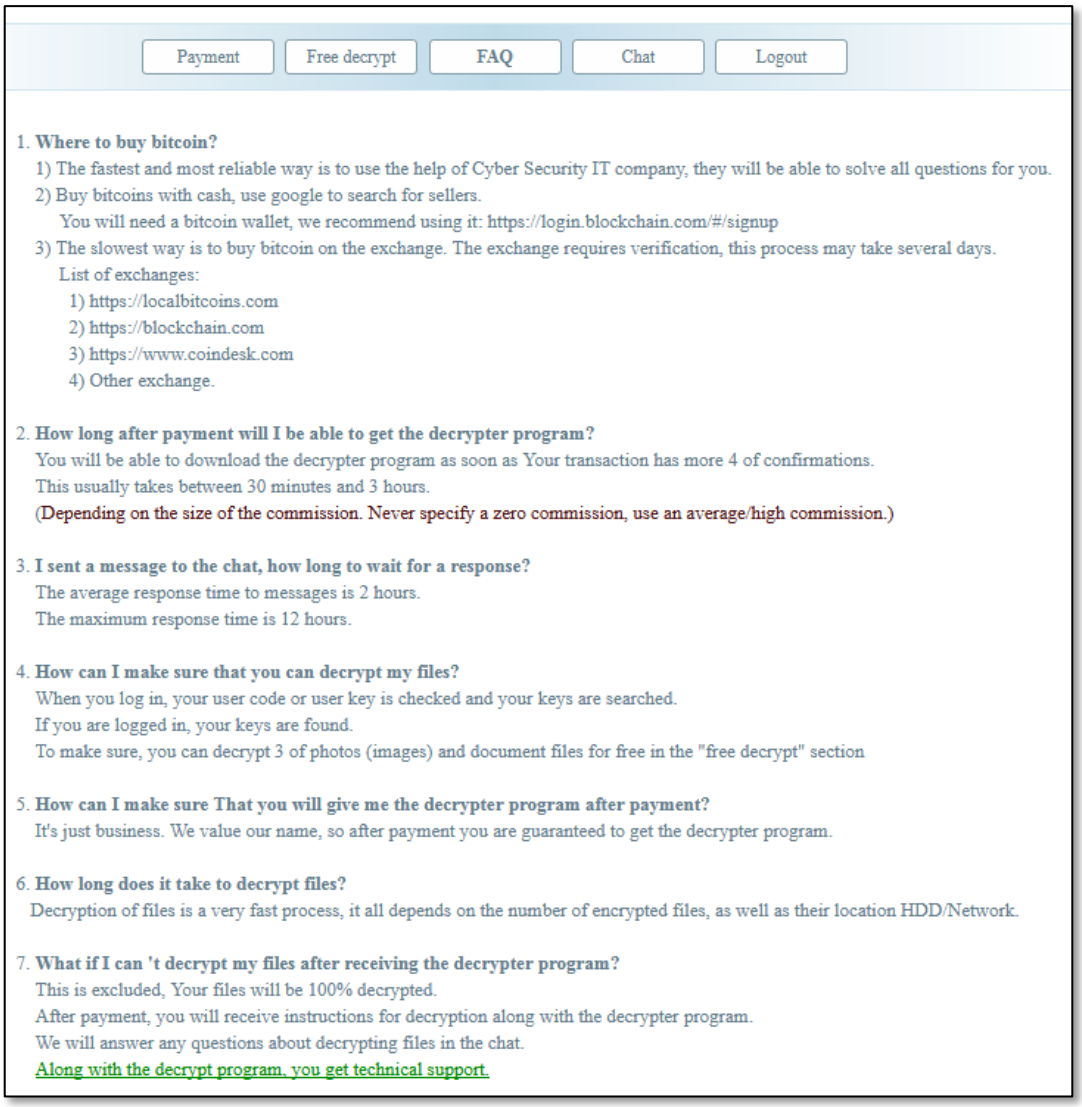

Figure 15. NetWalker Ransom Portal; FAQ Page

CIRCUS SPIDER offers the victim the opportunity to get in touch with the operators through a chat form as shown in Figure 16. This is similar to other ransomware operations such as REvil, Maze and DopperPaymer.

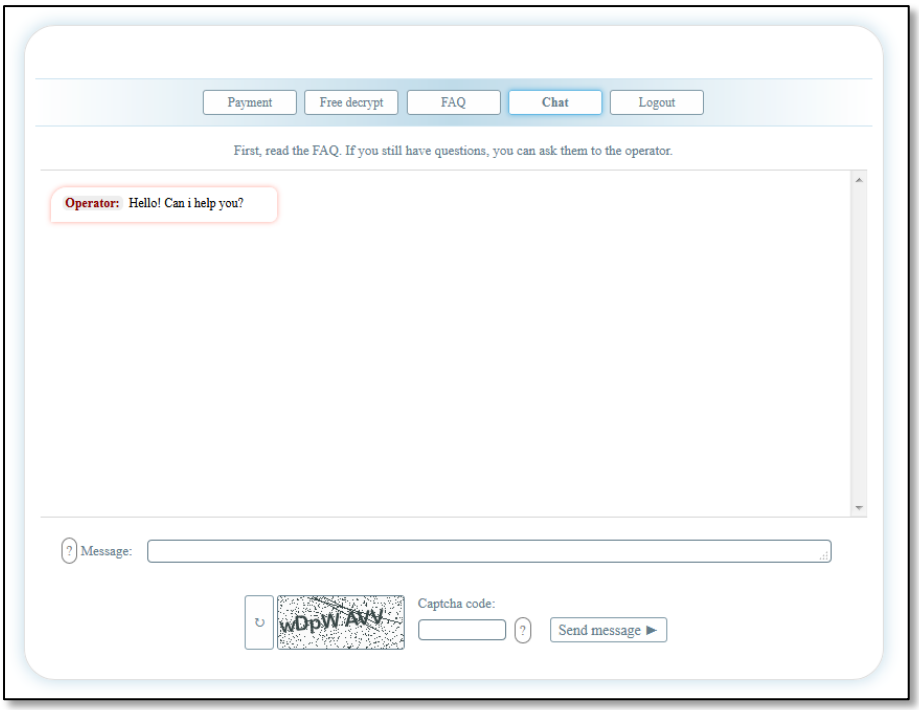

Figure 16. NetWalker Ransom Portal; Operator Chat

After the victim has successfully paid the ransom, the decrypter is available on the portal main page as illustrated in Figure 17.

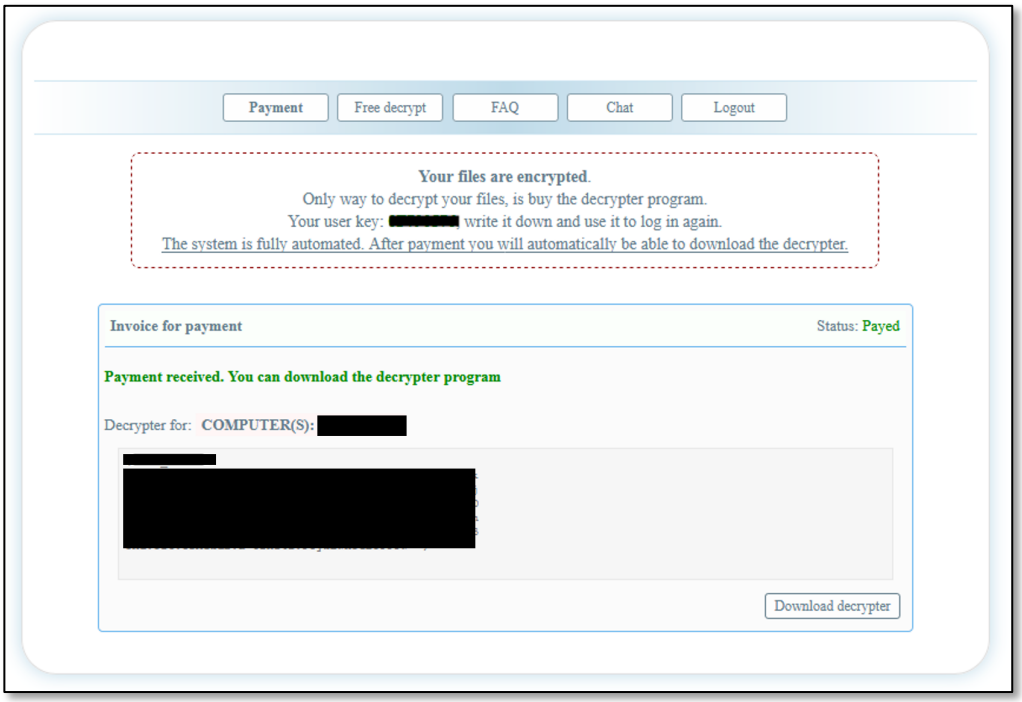

Figure 17. NetWalker Ransom Portal Main Page After Payment

Leak Site

The ransom portal also provides information about their leak site in which CIRCUS SPIDER is publishing data that has been exfiltrated from the victim's network. The leak site includes a countdown to when the data will be published (if the ransom is not paid) and a link from where the data archive can be downloaded along with the archive password. The post includes several screenshots that show a preview of files that have been exfiltrated during the breach. This information is provided regardless of whether the countdown has expired or not, likely aimed to intimidate the victim to pay the ransom. A screenshot of the leak site main page is illustrated in Figure 18.

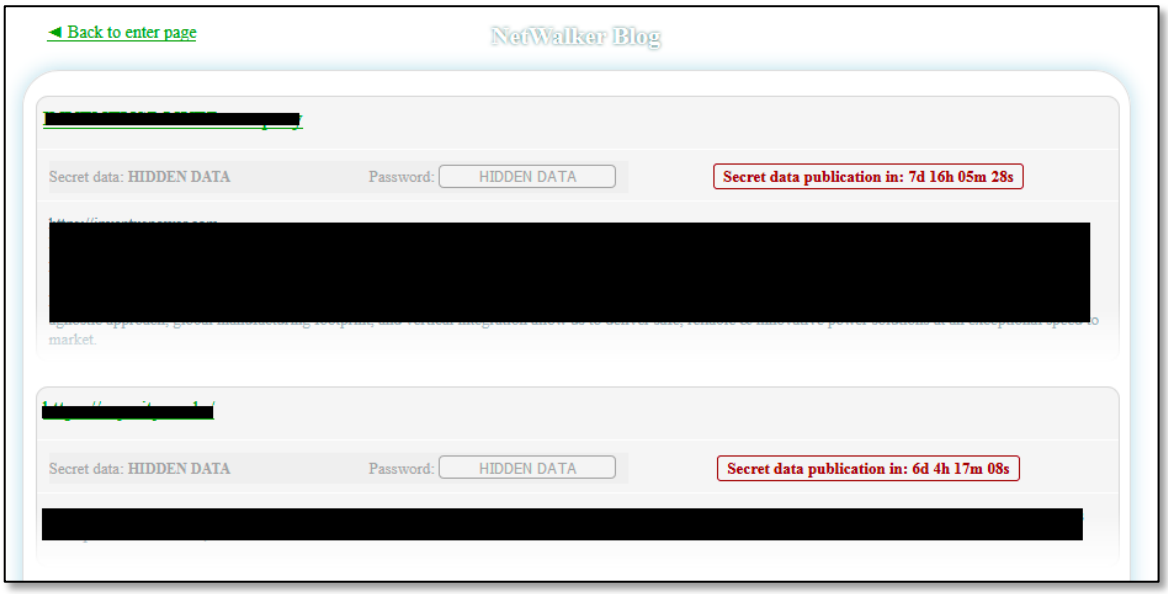

Figure 18. NetWalker Leak Site Main Page

Decrypter

The decrypter is supplied to the victim in a ZIP archive that includes two files; an executable and a text file, as shown in Figure 19.

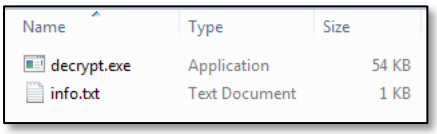

Figure 19. NetWalker Decrypter Archive Contents

The text file includes instructions on how to use the decrypter as shown in Figure 20.

```
This decrypt file for COMPUTER(S): code_XXYYZZ

Run decrypt.exe on PC which you want decrypt. Click "Auto decrypt" -> click
"delete crypter note file" -> click "decrypt".
The program will automatically decrypt all files on an encrypted PC.
The decryption program will fit all encrypted PCs.
```

```
After running the decryption in automatic or manual mode, the program can be
closed only when the close button becomes active,
never kill the process, if you kill the process your files will be damaged
and they will not be able to recover.

If you want to decrypt the entire network at once, use the following command:
psexec <params> "decrypt.exe" /S /D

/s - silent mode.
/d - delete lending(optional, not work without /s).

The program exit code will indicate the number of decrypted files.
```

Figure 20. NetWalker Decrypter Instructions

The `decrypt.exe` is the decrypter itself and comes with a graphical user interface in which the victim can choose to automatically decrypt all the encrypted files recursively. The user interface is illustrated in Figure 21.

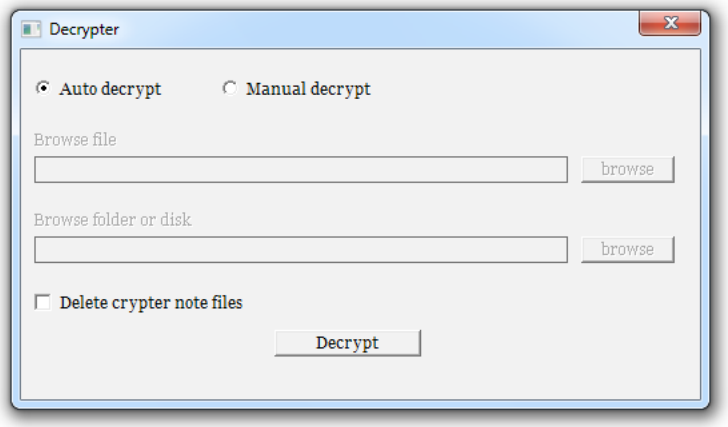

Figure 21. NetWalker Decrypter User Interface

CONCLUSION

CIRCUS SPIDER has demonstrated that they are a capable threat group with active development of their malware and ransomware affiliate service, advertising payouts of over a million dollars per breach. The author shows attention to detail as NetWalker not only terminates per-configured applications, but also applications which prevent a target file from being encrypted.

CIRCUS SPIDER claims that their entire operation is automated in which a victim will automatically be provided with a decryptor after the ransom amount has been paid to the operator-controlled Bitcoin wallet. This also includes the leak site where data is automatically published when the countdown timer expires, providing CIRCUS SPIDER with time to focus on improving their service rather than handling exfiltrated data and verifying ransom payments.

TACTICS, TECHNIQUES, AND PROCEDURES

The following TTPs may be used to characterize the NetWalker activity described in this Tipper:

- Uses ChaCha and Curve25519 encryption algorithms to encrypt file contents
- Stores the host-specific private key encrypted in the Windows registry
- Encrypts files on the local file system along with mapped and non-mapped network shares
- Encrypts all files that are not blacklisted in the NetWalker configuration
- Uses token impersonation in order to access network shares
- Uses a ransom portal on the TOR network to communicate with their victims
- Utilize data extortion to further pressure victims into paying a ransom

Host Indicators of Attack

The tables below detail files belonging to the NetWalker ransomware including SHA256 hash and build time.

Executables

SHA256 HASH	BUILD TIME (UTC)
cf80fd95183ce1305becf6ea91d4ccfa0d87d923499d08939324d63c0ed22dc4	2020-04-13 11:40:39
eb1470786fda58fc8291e099c7fcd5d36a04de85d1f6fe8683c1950b7119314e	2020-04-13 11:40:39
ee531cd7011cb5c2625d40892b70cf7e3860dbb92648391068e1f340e5d6c47f	2020-03-26 15:20:15
f2215e1a848bc5a5d172745201ea428b1d16fee7c814c5c5180a94a134592e86	2020-03-26 15:20:15
4a4c435cb63270787529355b7f48c3af14a7b3c466eafdf99fc3b9f35a83acc7	2020-03-26 15:20:15
c68fa4e70bb109d5e1270ebab3f1d64a3ad50a083f7f82e141156bdcb5f4ae16	2020-03-26 15:20:15
70e06890fa6619d761ef7bd890f4576fbd5371f06fcc8d12adb42dcb051a8490	2020-03-26 15:20:15
adf29a219ba2d948dd856ee7abaa51babaa30e11ad3ca56b58e66336c3c2369f	2020-03-17 08:24:17
15a4cd4a7baca3961fb0113164434c535af85cedd54744e14a4d4d7b106dd060	2020-03-01 12:46:56
a1115fa0c74f51d35063f8d3caf1d49156ab78872c3723d5e585a4ee8e4f8370	2020-03-01 12:46:56
d950a94534129202aa308f22d6c3d33f71af884d5556671a2b7f6ba8994cc995	2020-03-01 12:46:56
b2dc53aca54c595ad4dabf625f8da49d839a1370686bfcd46ccc12b3a6e8fa77	2020-03-01 12:46:56
55a32decdd9625245bf064c832962bf2271bff8bb5b8d8fb1bc6ec06dae4aea6	2020-03-01 12:46:56

b67bc1d9c7fe0672a79076e1546827e0642901cd62f7795d7a403bc3ba4a7117	2020-03-01 12:46:56
ce6fae0451e342c3280eeeedc487be09d15607b11f1d736798f7260d309fb55b	2020-03-01 12:46:56
07b89f41b347d639953fdbc0d98093670fc2d6ec50bad5c352165ddeca8ae9ad	2020-03-01 12:46:56
48eebda6ff2c95ae27983149e3b7537e00905ab932b3bbf09e17956325a2c172	2020-03-01 12:46:56
3f16af6271ed0cde4589012505e34a1ea3219ef39446829ace94d24391ba3993	2020-03-01 12:46:56

File System

The presence of one or more of the following files may indicate a NetWalker infection.

FILE PATH	DESCRIPTION
[A-F0-9]{5,8}-ReadMe.txt	Ransom note

Registry

KEY	VALUE(s)	DATA
HKCU\Software\[A-F0-9]{5,8}	[A-F0-9]{5,8}	Store the key recovery blob (encrypted host-specific private key)
HKLM\Software\[A-F0-9]{5,8}	[A-F0-9]{5,8}	Store the key recovery blob (encrypted host-specific private key)

YARA Rules

```
rule CrowdStrike_CSIT_20081_01 : circus_spider netwalker ransomware
{
    meta:
        copyright = "(c) 2020 CrowdStrike Inc."
        description = "Detects the NetWalker ransomware"
        reports = "CSIT-20081"
        version = "202004281747"
        last_modified = "2020-04-28"
        malware_family = "NetWalker"
    strings:
        $salsaconst = "expand 32-byte kexpand 16-byte k"
        $ins_getapi = { 55 8B EC A1 ?? ?? ?? ?? 5D C3 }
        $ins_crc32 = { 25 20 83 B8 ED 33 D0 }
        $ins_push1337 = { 68 39 05 00 00 68 69 7A 00 00 }
        $ins_rc4 = { 8B 45 ( E? |F? ) 83 C0 01 33 D2 B9 00 01 00 00 F7 F1 89
55 }
        $ins_c25519 = { 6A 00 68 41 DB 01 00 }
    condition:
        3 of them
}

rule CrowdStrike_CSIT_20081_02 : circus_spider netwalker ransomware
```

```
{
    meta:
        copyright = "(c) 2020 CrowdStrike Inc."
        description = "Detects the NetWalker ransomware"
        reports = "CSIT-20081"
        version = "202004281748"
        last_modified = "2020-04-28"
        malware_family = "NetWalker"
    strings:
        $ = "namesz"    fullword
        $ = "crmask"    fullword
        $ = "idsz"      fullword
        $ = "lend"      fullword
        $ = "lfile"     fullword
        $ = "mpk"       fullword
        $ = "namesz"    fullword
        $ = "pspath"    fullword
        $ = "rwsz"      fullword
        $ = "spsz"      fullword
        $ = "svcwait"   fullword
        $ = "unlocker"  fullword
        $ = "onion1"    fullword
    condition:
        10 of them
}
```

Network Artifacts

Infrastructure for NetWalker Ransomware

INFRASTRUCTURE	CONNECTION TYPE	DESCRIPTION
rnfdsgm6wb6j6su5txkekw4u4y47kp2eatvu7d6xhyn5cs4lt4pdrqqd[.]onion	HTTPS	Ransom portal on TOR
pb36hu4spl6cyjdfhing7h3pw6dhpk32ifemawkujj4gp33ejzdq3did[.]onion	HTTPS	Ransom portal on TOR

ATT&CK Framework

The following table maps reported NetWalker ransomware TTPs to the ATT&CK framework.

TACTIC	TECHNIQUE	OBSERVABLE
EXECUTION	**T1059:** Command Line Interface	NetWalker deletes the systems' shadow copies before encrypting files
PRIVILEGE ESCALATION	**T1134:** Access Token Manipulation **T1088:** Bypass User Account Control	NetWalker uses token manipulation to access network shares. NetWalker features UAC bypass using the Windows Event Viewer and the Windows Activation Client
DISCOVERY	**T1083:** File and Directory Discovery **T1135:** Network Share Discovery	NetWalker walks the system file system and network shares in order to encrypt files
IMPACT	**T1486:** Data Encrypted for Impact	NetWalker encrypts files that does not match its configured whitelist

APPENDIX A

JSON KEY	TYPE	DECRIPTION
lfile	String	Ransom note template string
spsz	Integer	Encryption chunk size
lend	String	Ransom note template (Base64 encoded)
namesz	Integer	Length of the infection ID to be used in the file extension of encrypted files
thr	Integer	Maximum number of concurrent encryption threads
mpk	String	The operator's ECC public key
pers	Boolean	Attempt UAC bypass
unlocker	Dictionary	Defines operation of the file unlocker
unlocker.use	Boolean	Use the unlocker feature to unlock locked files
unlocker.ignore	Dictionary	Includes processes and file paths to be ignored by the unlocker
unlocker.ignore.pspath	List	List of paths to ignore
unlocker.ignore.use	Boolean	Use the unlocker whitelist
unlocker.prc	List	List of processes to ignore
idsz	Integer	Length of the infection ID
mode	Integer	Unknown
net	Dictionary	Defines behavior of share enumeration
net.use	Boolean	Enable or disable encryption of enumerated network shares
net.ignore.use	Boolean	Use whitelist
net.ignore.share	List	Whitelisted file shares
net.ignore.disk	Boolean	Unknown
kill	Dictionary	Defines behavior for terminating tasks, processes and services
kill.use	Boolean	Enable or disable the termination feature
kill.svcwait	Integer	Number of seconds to wait for the service termination thread to complete before continuing execution
kill.svc	List	List of services to terminate
kill.prc	List	List of processes to terminate
kill.task	List	List of tasks to terminate
white	Dictionary	File encryption whitelist
white.path	List	List of whitelisted file paths

white.ext	List	List of whitelisted file extensions
white.file	List	List of whitelisted file names
onion1	String	Address of the ransom portal on TOR
onion2	String	Address of the ransom portal on TOR

APPENDIX B

NetWalker Configuration

```
{
      "lfile": "{id}-Readme.txt",
      "spsz": 15360,
      "lend":
"SGkhDQpZb3VyIGZpbGVzIGFyZSBlbmNyeXB0ZWQgYnkgTmV0d2Fsa2VyLg0KQWxsIGVuY3J5cHRl
ZCBmaWxlcyBmb3IgdGhpcyBjb21wdXRlciBoYXMgZXh0ZW5zaW9uOiAue2lkfQ0KDQotLQ0KSWYgZ
m9yIHNvbWUgcmVhc29uIHlvdSByZWFkIHRoaXMgdGV4dCBiZWZvcmUgdGhlIGVuY3J5cHRpb24gZW
5kZWQsDQp0aGlzIGNhbiBiZSB1bmRlcnN0b29kIGJ5IHRoZSBmYWN0IHRoYXQgdGhlIGNvbXB1dGV
yIHNsb3dzIGRvd24sDQphbmQgeW91ciBoZWFydCByYXRlIGhhcyBpbmNyZWFzZWQgZHVlIHRvIHlh
YmlsaXR5IHRvIHR1cm4gaXQgb2ZmLA0KdGhlbiB3ZSByZWNvbW1lbmQgdGhhdCB5b3UgbW92Z
SBhd2F5IGZyb20gdGhlIGNvbXB1dGVyIGFuZCBhY2NlcHQgdGhhdCB5b3UgaGF2ZSBiZWVuIGNvbX
Byb21pc2VkLg0KUmVib290aW5nL3NodXRkb3duIHdpbGwgY2F1c2UgeW91IHRvIGxvc2UgZmlsZXM
gd2l0aG91dCB0aGUgc9zc2liaWxpdHkgb2YgcmVjb3ZlcnkuDQoNCi0tDQoPdXIgZmlsZXMgYXJlIHZlcnkgbS4NCg0KRG8gbm90IHR
yeSB0byByZWNvdmVyIHlvdXIgZmlsZXMgd2l0aG91dCBhIGRlY3J5cHRlciBwcm9ncmFtLCB5b3Ug
bWF5IGRhbWFnZSB0aGVtIGFuZCB0aGVuIHRoZXkgd2lsbCBiZSBpbXBvc3NpYmxlIHRvIHJlY292Z
XIuDQoNCkZvciB1cyB0aGlzIGlzIGp1c3QgYnVzaW5lc3MgYW5kIHRvIHByb3ZlIHRvIHJlcHV
Igc2VyaW91c25lc3MsIHdlIHdpbGwgZGVjcnlwdCB5b3Ugb25lIGZpbGUgZm9yIGZyZWUuDQpKdXN
0IG9wZW4gb3VyIHdlYnNpdGUsIHVwbG9hZCB0aGUgZW5jcnlwdGVkIGZpbGUgYW5kIGdldCB0aGUg
ZGVjcnlwdGVkIGZpbGUgZm9yIGZyZWUuDQoNCi0tDQoNClN0ZXBzIHRvIGdldCBhY2Nlc3Mgb24gb
3VyIHdlYnNpdGU6DQoNCjEuRG93bmxvYWQgYW5kIGluc3RhbGwgdG9yLWJyb3dzZXI6IGh0dHBzOi
8vdG9ycHJvamVjdC5vcmcvDQoNCjIuT3BlbiBvdXIgd2Vic2l0ZToge29uaW9uMX0NCklmIHRoZSB
3ZWJzaXRlIGlzIG5vdCBhdmFpbGFibGUsIG9wZW4gYW5vdGhlciBvbmU6IHtvbmlvbjJ9DQoNCjMu
UHV0IHlvdXIgcGVyc29uYWwgY29kZSBpbiB0aGUgaW5wdXQgZm9ybToNCg0Ke2NvZGV9",
      "namesz": 8,
      "thr": 1500,
      "mpk": "<redacted>",
      "pers": false,
      "unlocker": {
            "ignore": {
                  "pspath": ["*:\\windows*", "*:\\winnt*", "*:\\program
file*\\vmwar*", "*\\Program File*\\Fortinet"],
                  "use": true,
                  "prc": ["psexec.exe", "system", "forti*.exe", "fmon.exe",
"fcaptmon.exe", "FCHelper64.exe"]
            },
            "use": true
      },
      "idsz": 6,
      "mode": 0,
      "net": {
            "ignore": {
                  "use": true,
                  "disk": true,
                  "share": ["ipc$", "admin$"]
            },
            "use": true
```

```
      },
      "kill": {
            "use": true,
            "svcwait": 0,
            "svc": ["Lotus*", "veeam*", "cbvscserv*", "hMailServer",
"backup*", "*backup*", "apach*", "firebird*", "ibmiasrw", "IBM Domino*",
"Simply Accounting Database Connection Manager", "IASJet", "QB*", "*sql*",
"sql*", "QuickBooksDB*", "IISADMIN", "omsad", "dc*32", "server
Administrator", "wbengine", "mr2kserv", "MSExchange*", "ShadowProtectSvc",
"SP*4", "teamviewer", "MMS", "AcronisAgent", "ARSM", "AcrSch2Svc",
"vsnapvss", "SPXService", "StorageCraft ImageManager", "wrsvc",
"stc_endpt_svc", "acrsch2svc*"],
            "prc": ["nslsvice.exe", "pg*", "nservice.exe", "cbvscserv*",
"ntrtscan.exe", "cbservi*", "hMailServer*", "IBM*", "bes10*", "black*",
"apach*", "bd2*", "db*", "ba*", "be*", "QB*", "oracle*", "wbengine*", "vee*",
"postg*", "sage*", "sap*", "b1*", "fdlaunch*", "msmdsrv*", "report*",
"msdtssr*", "coldfus*", "cfdot*", "swag*", "swstrtr*", "jetty.exe",
"wrsa.exe", "team*", "agent*", "store.exe", "sql*", "sqbcoreservice.exe",
"thunderbird.exe", "ocssd.exe", "encsvc.exe", "excel.exe", "synctime.exe",
"mspub.exe", "ocautoupds.exe", "thebat.exe", "dbeng50.exe", "*sql*",
"mydesktopservice.exe", "onenote.exe", "outlook.exe", "powerpnt.exe",
"msaccess.exe", "tbirdconfig.exe", "wordpad.exe", "ocomm.exe", "dbsnmp.exe",
"thebat64.exe", "winword.exe", "oracle.exe", "xfssvccon.exe",
"firefoxconfig.exe", "visio.exe", "mydesktopqos.exe", "infopath.exe",
"agntsvc.exe"],
            "task": ["reboot", "restart", "shutdown", "logoff", "back"]
      },
      "white": {
            "path": ["*system volume information", "*windows.old",
"*:\\users\\*\\*temp", "*msocache", "*:\\winnt", "*$windows.~ws",
"*perflogs", "*boot", "*:\\windows", "*:\\program file*\\vmware",
"\\\\*\\users\\*\\*temp", "\\\\*\\winnt", "\\\\*\\windows", "*\\program
file*\\vmware", "*appdata*microsoft", "*appdata*packages",
"*microsoft\\provisioning", "*dvd maker", "*Internet Explorer", "*Mozilla",
"*Mozilla*", "*Old Firefox data", "*\\program file*\\windows media*",
"*\\program file*\\windows portable*", "*windows defender", "*\\program
file*\\windows nt", "*\\program file*\\windows photo*", "*\\program
file*\\windows side*", "*\\program file*\\windowspowershell", "*\\program
file*\\cuass*", "*\\program file*\\microsoft games", "*\\program
file*\\common files\\system", "*\\program file*\\common files\\*shared",
"*\\program file*\\common files\\reference ass*", "*\\windows\\cache*",
"*temporary internet*", "*media player",
"*:\\users\\*\\appdata\\*\\microsoft",
"\\\\*\\users\\*\\appdata\\*\\microsoft"],
            "ext": ["msp", "exe", "sys", "msc", "mod", "clb", "mui",
"regtrans-ms", "theme", "hta", "shs", "nomedia", "diagpkg", "cab", "ics",
"msstyles", "cur", "drv", "icns", "diagcfg", "dll", "ocx", "lnk", "ico",
"idx", "ps1", "mpa", "cpl", "icl", "msu", "msi", "nls", "scr", "adv", "386",
"com", "hlp", "rom", "lock", "386", "wpx", "ani", "prf", "rtp", "ldf", "key",
"diagcab", "cmd", "spl", "deskthemepack", "bat", "themepack"],
            "file": ["ntuser.dat*", "iconcache.db", "gdipfont*.dat",
"ntuser.ini", "usrclass.dat", "usrclass.dat*", "boot.ini", "bootmgr",
"bootnxt", "desktop.ini", "ntuser.dat", "autorun.inf", "ntldr", "thumbs.db",
"bootsect.bak", "bootfont.bin"]
```

```
      },
      "onion2":
"rnfdsgm6wb6j6su5txkekw4u4y47kp2eatvu7d6xhyn5cs4lt4pdrqqd[.]onion",
      "onion1":
"pb36hu4spl6cyjdfhing7h3pw6dhpk32ifemawkujj4gp33ejzdq3did[.]onion"
 }
```