# Rapidly Evolving BlackMatter Ransomware Tactics – Cipher Tech Solutions, Inc.

Archived: 2026-04-05 21:41:47 UTC


September 8, 2021 | By Cipher Tech ACCE Team

## Summary

Cipher Tech analysts monitoring VirusTotal for BlackMatter ransomware activity discovered new variants of BlackMatter malware self-reporting as versions 1.9 and 2.0. The new BlackMatter malware samples contain additional functionality, changes to the configuration data, and version 2.0 additionally introduces changes to the configuration decryption algorithm. Cipher Tech analysts developed an ACCE module to automate the extraction of BlackMatter malware's configuration data. Cipher Tech's analysis reveals new variants of BlackMatter malware and a refinement in BlackMatter's tactics, techniques, and procedures.

## Who is BlackMatter

BlackMatter is a ransomware affiliate program actively attacking victims that was first identified in July 2021 by RecordedFuture and a security researcher pancak3. BlackMatter claims to derive from the now inactive REvil and DarkSide ransomware affiliate programs. Supporting BlackMatter's claim, a joint code correlation effort between BleepingComputer and Emsisoft suggests that BlackMatter malware's' encryption routines contain strong similarities to DarkSide malware samples.

BlackMatter claims to target companies in the US, UK, Canada, or Australia with between 500 – 15,000 hosts and revenues of at least 100K. BlackMatter claims they will not target the following industries and will provide free decryption services to BlackMatter victims in these industries.

- Hospitals
- Critical infrastructure facilities

- Oil and gas industry
- Defense industry
- Non-profit companies
- Government sector

## Background

Cipher Tech analysts leveraged ACCE to discover, decrypt, and parse configuration data from BlackMatter ransomware samples with reported version numbers of 1.9 and 2.0. Cipher Tech has not identified detailed public reporting for samples with a version number above 1.2. This suggests either a fast development cycle or a refinement in TTPs with respect to versioning.

Cipher Tech's ACCE software enables the automated extraction of configuration data from supported malware samples. ACCE indicated that parsing the 1.9 and 2.0 samples configuration data differed from the expected format of ACCE's existing BlackMatter configuration parser. Based on the change in configuration, Cipher Analysts examined both samples to identify differences between BlackMatter's 1.2 ransomware sample, BlackMatter's 1.9 ransomware sample, and BlackMatter's 2.0 ransomware sample.

Analysis of the BlackMatter 1.9 sample and its configuration data revealed that BlackMatter added additional capabilities to BlackMatter 1.9. BlackMatter 1.9 contains the following new functionality:

- Print ransom notes to a locally installed computer
- Check the hostname of a compromised system prior to installation
- Additional verification checks to ensure BlackMatter does not encrypt their own ransom note
- Provide additional details and evidence of stolen information within the ransom note

Further analysis of the BlackMatter 2.0 sample, reported by @sisoma2 on Twitter, and its configuration data revealed that BlackMatter uses all the changes indicated for 1.9 above, and changed the decryption algorithm for the configuration data and some encrypted strings. BlackMatter 2.0 uses the following decryption methodology:

- A 64-bit seed/key for configuration and string decryption
- A modified MMIX-LCG pseudo-random number generator (PRNG) algorithm for generating the key used to decrypt each 8-byte block of the encrypted configuration data / string.
- A byte-swap algorithm for adjusting the generated MMIX-LCG state value into an XOR key for decrypting each 8-byte block.

## Samples Compared

This post will demonstrate these changes in BlackMatter Versions 1.9 and 2.0 in comparison to a BlackMatter Version 1.2 sample. For the purposes of the analysis presented below:

- Version 1.2 reflects the BlackMatter Ransomware with MD5 598c53bfef81e489375f09792e487f1a, compiled 2021-07-23T20:51:18+00:00
- Version 1.9 reflects the BlackMatter Ransomware with MD5 f1c260c31b9d3f9ff54a142d508ec602, compiled 2021-08-12T22:22:01+00:00
- Version 2.0 reflects the BlackMatter Ransomware with MD5 38035325b785329e3f618b2a0b90eb75, compiled 2021-08-16T07:13:07+00:00

## Configuration Decryption

The encrypted configuration data structure  for 1.9 remains the same as previously documented by Group-IB, the data is stored in the .rsrc PE section, although there are no actual resources, the following Python construct documents the layout:

```
SPEC = construct.Struct(
    "seed" / construct.Bytes(4),
    "encrypted" / construct.Prefixed(construct.Int32ul, construct.GreedyBytes)
)
```

*Figure 1: Version 1.9 Encrypted Configuration Structure*

The encrypted configuration data structure for 2.0, however, is modified to account for the change in decryption algorithm. The following Python structure defines the change, where the seed is now 8 bytes instead of 4:

```
SPEC = construct.Struct(
    "seed" / construct.Bytes(8),
    "encrypted" / construct.Prefixed(construct.Int32ul, construct.GreedyBytes)
)
```

*Figure 2: Version 2.0 Encrypted Configuration Structure*

**Algorithm Changes**

The decryption algorithm in Versions 1.2 and 1.9 used an XOR algorithm where the key for each 4-byte block was generated using a modified Delphi/Pascal-LCG PRNG. The standard Delphi/Pascal-LCG masks each round to generate a floating point number in the range 0-1, where the modified algorithm masks each round using the following, where “seed” is the initial seed value reflected in Figure 1:

```
(value * seed) >> 32
```

*Figure 3: Version 1.9 Delphi/Pascal-LCG Mask*

Version 2.0 uses a modified MMIX-LCG algorithm for key generation, where the standard parameters are represented in Figure 4 and the BlackMatter parameters are represented in Figure 5. Additionally, in the standard MMIX-LCG the value is not masked, and in BlackMatter the value is masked as seen in Figure 6, where “seed” is the initial seed value reflected in Figure 2.

```
a=6364136223846793005, c=1442695040888963407, m=2**64,
```

*Figure 4: Standard MMIX-LCG Parameters*

```
a=6364136223846793005, c=1, m=2**64,
```

*Figure 5: BlackMatter MMIX-LCG Parameters*

```
(value * seed) % 2**64
```

*Figure 6: BlackMatter MMIX-LCG Mask*

In addition to the LCG-PRNG algorithm changes, 2.0 decrypts in 8-byte blocks (previously 4-byte blocks) and conducts a byte-swap on the MMIX-LCG generated value each round before using it as an XOR key for the 8-byte block. For each MMIX-LCG generated value, the following byte-swap is performed:

```
key = next_key.to_bytes(8, "little")
key_bytes = bytes([key[0], key[5], key[1], key[4], key[2], key[7], key[3], key[6]])
key_int = int.from_bytes(key_bytes, "little")
```

*Figure 7: Version 2.0 Byte-Swap*

Following decryption, there are no additional changes in the procedure, and configuration data is aPLib decompressed in all versions (1.2, 1.9, and 2.0) and parsed.

### Configuration Parsing

The decrypted and decompressed configuration data yields several differences, as described below. There were no changes in the configuration data from 1.9 to 2.0, so all differences will reflect a comparison between 1.2 and 1.9.

#### Difference 1: Configuration Flag – Print Ransom Note

```
decrypted = DecryptBuffer(configuration.data);
compressed = decrypted;
if ( decrypted )
{
  config_data = Call_RtlAllocateHeap(4 * configuration.Size);
  if ( config_data )
  {
    if ( aPLibDecompression(compressed, config_data) != -1 )
    {
      LODWORD(int_128) = 128;
      memcpy(rsa_modulus, config_data, int_128);
      LODWORD(int_32) = 32;
      memcpy(&bot_company, config_data->bot_company, int_32);
      LODWORD(int_8) = 8;
      memcpy(&flag_encrypt_odd, &config_data->flags, int_8);
```

```
decrypted = DecryptBuffer(configuration.data);
compressed = decrypted;
if ( decrypted )
{
  config_data = Call_ReallocateHeap(4 * configuration.Size);
  if ( config_data )
  {
    if ( aPLibDecompress(compressed, config_data) != -1 )
    {
      LODWORD(int_128) = 128;
      memcpy(rsa_modulus, config_data, int_128);
      LODWORD(int_32) = 32;
      memcpy(bot_company, config_data->bot_company, int_32);
      LODWORD(int_9) = 9;
      memcpy(&flag_encrypt_odd, &config_data->flags, int_9);
```

*Figure 8: Version 1.2/1.9 Configuration Flag Parsing*

The first noticeable difference is in the number of parsed flags. In Version 1.2 on the left, 8 flags are parsed, while in Version 1.9 on the right, 9 flags are parsed.

```
00000000 config          struc ; (sizec
00000000 rsa_modulus     db 128 dup(?)
00000080 bot_company     db 16 dup(?)
00000090 aes_network_key db 16 dup(?)
000000A0 encrypt_odd     db ?
000000A1 authenticate    db ?
000000A2 volume_mount    db ?
000000A3 network_rsrcs   db ?
000000A4 terminate_procs db ?
000000A5 delete_services db ?
000000A6 create_mutex    db ?
000000A7 send_status     db ?
```

```
00000000 config          struc ; (sizeo
00000000 rsa_modulus     db 128 dup(?)
00000080 bot_company     db 16 dup(?)
00000090 aes_network_key db 16 dup(?)
000000A0 encrypt_odd     db ?
000000A1 authenticate    db ?
000000A2 volume_mount    db ?
000000A3 network_rsrcs   db ?
000000A4 terminate_procs db ?
000000A5 delete_services db ?
000000A6 create_mutex    db ?
000000A7 print_ransom    db ?
000000A8 send_status     db ?
```

*Figure 9: Version 1.2/1.9 Configuration Structure*

The additional flag is described above for Version 1.9 on the right as *print_ransom*, which is noticeably absent from Version 1.2 on the left. If the *print_ransom* field is set, during runtime the BlackMatter ransomware will invoke the method below, which calls the dynamically resolved API *GetDefaultPrinterW* (which is not resolved in Version 1.2) and if the default printer is not a “PDF” printer, it will print the ransom note to that printer.

```
HINSTANCE __stdcall PrintRansomNote(LPCWSTR lpFile)
{
  HINSTANCE result; // eax
  WCHAR pszBuffer[260]; // [esp+0h] [ebp-220h] BYREF
  wchar_t Operation[6]; // [esp+208h] [ebp-18h] BYREF
  wchar_t SubStr[4]; // [esp+214h] [ebp-Ch] BYREF
  DWORD pcchBuffer; // [esp+21Ch] [ebp-4h] BYREF

  pcchBuffer = 260;
  result = GetDefaultPrinterW(pszBuffer, &pcchBuffer);
  if ( result )
  {
    wcscpy(SubStr, L"PDF");
    result = wcsstr(pszBuffer, SubStr);
    if ( !result )
    {
      wcscpy(Operation, L"print");
      return ShellExecuteW(0, Operation, lpFile, 0, 0, 0);
    }
  }
  return result;
}
```

*Figure 10: Version 1.9 Print Ransom Note Function*

**Difference 2: Configuration Parameter – Invalid Hostnames**

```
00000000 config          struc ; (sizeo
00000000 rsa_modulus     db 128 dup(?)
00000080 bot_company     db 16 dup(?)
00000090 aes_network_key db 16 dup(?)
000000A0 encrypt_odd     db ?
000000A1 authenticate    db ?
000000A2 volume_mount    db ?
000000A3 network_rsrcs   db ?
000000A4 terminate_procs db ?
000000A5 delete_services db ?
000000A6 create_mutex    db ?
000000A7 print_ransom    db ?
000000A8 send_status     db ?
000000A9 bypass_directories dd ?
000000AD bypass_files    dd ?
000000B1 bypass_extensions dd ?
000000B5 invalid_hostnames dd ?
```

*Figure 11: Version 1.9 Configuration Structure*

The second configuration difference is in the inclusion of invalid hostname hashes, displayed in the structure above. The hostnames are leveraged in the “safe mode” function, as described by Sophos. The difference from Version 1.2 to Version 1.9 is the inclusion of a function which checks if the compromised system is an invalid host.

```
const CHAR *SafeInstall()
{
  const CHAR *result; // eax

  result = CheckAdminGroup();
  if ( result )
  {
    result = AddAutoAdminLoginUser();
    if ( result )
    {
      result = InstallHKLMRunOnce();
```

```
void *SafeInstall()
{
  void *result; // eax

  result = CheckAdminGroup();
  if ( result )
  {
    result = CheckInvalidHost();
    if ( !result )
    {
      result = AddAutoAdminLoginUser();
      if ( result )
      {
        result = InstallHKLMRunOnce();
```

*Figure 12: Version 1.2/1.9 Safe Install Function*

As displayed in the screenshot below, the comparison is achieved by hashing and comparing the hostname to the list of invalid hostname hashes, using the ROR-13-ADD hash algorithm (lowercase string, include null terminator).

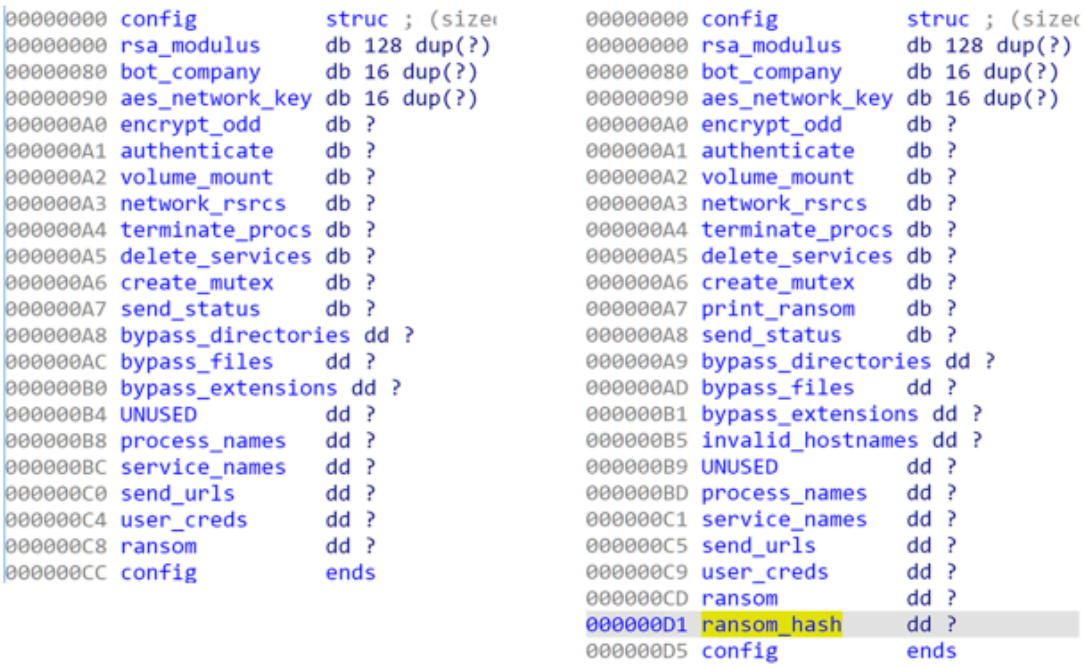
```
int CheckInvalidHost()
{
  int *hostname_hashes; // esi
  int host_hash; // ecx
  int curr_hash; // eax
  WCHAR *hostname; // [esp+4h] [ebp-8h]
  int invalid_host; // [esp+8h] [ebp-4h]

  invalid_host = 0;
  hostname_hashes = InvalidHostnameHashes;
  if ( InvalidHostnameHashes )
  {
    hostname = ObtainWideComputerName();
    if ( hostname )
    {
      host_hash = strlower_ror13_add_hash_algorithm(hostname, 0);
      do
      {
        curr_hash = *hostname_hashes++;
        if ( !curr_hash )
        {
          invalid_host = 0;
          goto LABEL_9;
        }
      }
      while ( curr_hash != host_hash );
      invalid_host = 1;
    }
LABEL_9:
    if ( hostname )
      call_free_heap(hostname);
  }
  return invalid_host;
}
```

*Figure 13: Version 1.9 Invalid Host Name Check*

The Version 1.9 analyzed sample was not configured with any invalid hostname hashes, however, it can be surmised that the malware authors are attempting to avoid running in Sandboxes or other Virtual Analysis Environments.

**Difference 3: Configuration Parameter – Ransom Note Hash**

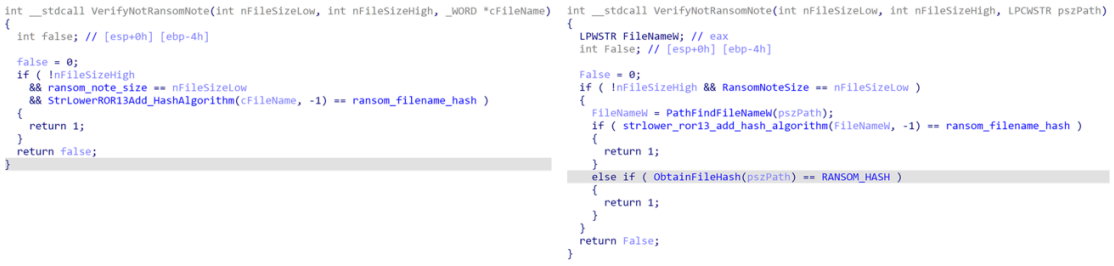


```
00000000 config          struc ; (size
00000000 rsa_modulus     db 128 dup(?)
00000080 bot_company     db 16 dup(?)
00000090 aes_network_key db 16 dup(?)
000000A0 encrypt_odd     db ?
000000A1 authenticate    db ?
000000A2 volume_mount    db ?
000000A3 network_rsrcs   db ?
000000A4 terminate_procs db ?
000000A5 delete_services db ?
000000A6 create_mutex    db ?
000000A7 send_status     db ?
000000A8 bypass_directories dd ?
000000AC bypass_files    dd ?
000000B0 bypass_extensions dd ?
000000B4 UNUSED          dd ?
000000B8 process_names   dd ?
000000BC service_names   dd ?
000000C0 send_urls       dd ?
000000C4 user_creds      dd ?
000000C8 ransom          dd ?
000000CC config          ends
```

```
00000000 config          struc ; (size
00000000 rsa_modulus     db 128 dup(?)
00000080 bot_company     db 16 dup(?)
00000090 aes_network_key db 16 dup(?)
000000A0 encrypt_odd     db ?
000000A1 authenticate    db ?
000000A2 volume_mount    db ?
000000A3 network_rsrcs   db ?
000000A4 terminate_procs db ?
000000A5 delete_services db ?
000000A6 create_mutex    db ?
000000A7 print_ransom    db ?
000000A8 send_status     db ?
000000A9 bypass_directories dd ?
000000AD bypass_files    dd ?
000000B1 bypass_extensions dd ?
000000B5 invalid_hostnames dd ?
000000B9 UNUSED          dd ?
000000BD process_names   dd ?
000000C1 service_names   dd ?
000000C5 send_urls       dd ?
000000C9 user_creds      dd ?
000000CD ransom          dd ?
000000D1 ransom_hash     dd ?
000000D5 config          ends
```

*Figure 14: Version 1.2/1.9 Configuration Structure*

The final configuration difference is the inclusion of the ransom note hash, using the ROR-13-ADD hash algorithm (include null terminator, use seed of -1).



```
int __stdcall VerifyNotRansomNote(int nFileSizeLow, int nFileSizeHigh, _WORD *cFileName)
{
  int false; // [esp+0h] [ebp-4h]

  false = 0;
  if ( !nFileSizeHigh
    && ransom_note_size == nFileSizeLow
    && StrLowerROR13Add_HashAlgorithm(cFileName, -1) == ransom_filename_hash )
  {
    return 1;
  }
  return false;
}
```

```
int __stdcall VerifyNotRansomNote(int nFileSizeLow, int nFileSizeHigh, LPCWSTR pszPath)
{
  LPWSTR FileNameW; // eax
  int False; // [esp+0h] [ebp-4h]

  False = 0;
  if ( !nFileSizeHigh && RansomNoteSize == nFileSizeLow )
  {
    FileNameW = PathFindFileNameW(pszPath);
    if ( strlower_ror13_add_hash_algorithm(FileNameW, -1) == ransom_filename_hash )
    {
      return 1;
    }
    else if ( ObtainFileHash(pszPath) == RANSOM_HASH )
    {
      return 1;
    }
  }
  return False;
}
```

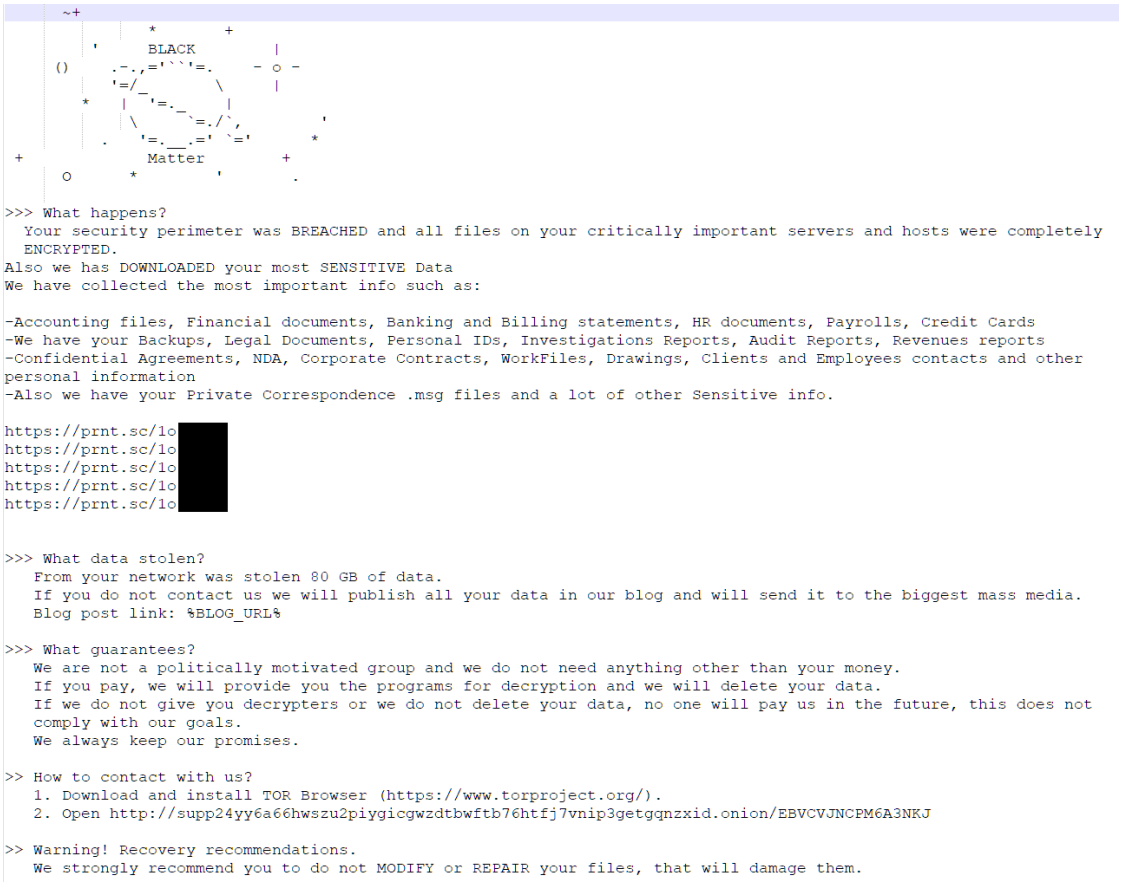
*Figure 15: Version 1.2/1.9 Verify Not Ransom Note Function*

In Version 1.2 on the left, we can observe that the BlackMatter ransomware will verify the input file is the same size as the ransom note, and that the filename hash (ROR-13-ADD hash algorithm (lowercase string, include null terminator, use seed of -1)) is the same as the previously calculated value. On the right, Version 1.9 contains the same checks, but additionally hashes the file data to compare against the configured hash value.

Ransom notes have been observed in varying sizes, with some containing extra information in comparison to others, so the purpose of this change is unknown.

**Difference 4: Ransom Note**

The ransom note obtained from Version 1.9 contains significantly more content than the one obtained from Version 1.2. The additional content includes details of the type of information stolen from the victim and *print.sc* links which presumably provide evidence that the information has been stolen. *print.sc* is a screenshot sharing service associated with the LightShot screenshot application.



```
      ~+
                *       +
          '     BLACK         |
     ()     .-.,="``"=.    - o -
            '=/_       \     |
         *   |  '=._    |
             |     `=./`,        '
          .   '=.__.=' `='      *
 +              Matter        +
      O      *        '       .

>>> What happens?
  Your security perimeter was BREACHED and all files on your critically important servers and hosts were completely
  ENCRYPTED.
Also we has DOWNLOADED your most SENSITIVE Data
We have collected the most important info such as:

-Accounting files, Financial documents, Banking and Billing statements, HR documents, Payrolls, Credit Cards
-We have your Backups, Legal Documents, Personal IDs, Investigations Reports, Audit Reports, Revenues reports
-Confidential Agreements, NDA, Corporate Contracts, WorkFiles, Drawings, Clients and Employees contacts and other
personal information
-Also we have your Private Correspondence .msg files and a lot of other Sensitive info.

https://prnt.sc/1o[redacted]
https://prnt.sc/1o[redacted]
https://prnt.sc/1o[redacted]
https://prnt.sc/1o[redacted]
https://prnt.sc/1o[redacted]


>>> What data stolen?
   From your network was stolen 80 GB of data.
   If you do not contact us we will publish all your data in our blog and will send it to the biggest mass media.
   Blog post link: %BLOG_URL%

>>> What guarantees?
   We are not a politically motivated group and we do not need anything other than your money.
   If you pay, we will provide you the programs for decryption and we will delete your data.
   If we do not give you decrypters or we do not delete your data, no one will pay us in the future, this does not
   comply with our goals.
   We always keep our promises.

>> How to contact with us?
   1. Download and install TOR Browser (https://www.torproject.org/).
   2. Open http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnzxid.onion/EBVCVJNCPM6A3NKJ

>> Warning! Recovery recommendations.
   We strongly recommend you to do not MODIFY or REPAIR your files, that will damage them.
```

*Figure 16: Version 1.9 Ransom Note*

```
      ~+
                *       +
          '     BLACK         |
     ()     .-.,="``"=.    - o -
            '=/_       \     |
         *   |  '=._    |
             |     `=./`,        '
          .   '=.__.=' `='      *
 +              Matter        +
      O      *        '       .

>>> What happens?
   Your network is encrypted, and currently not operational. We have downloaded 1TB from your fileserver.
   We need only money, after payment we will give you a decryptor for the entire network and you will restore all the
   data.

>>> What guarantees?
   We are not a politically motivated group and we do not need anything other than your money.
   If you pay, we will provide you the programs for decryption and we will delete your data.
   If we do not give you decrypters or we do not delete your data, no one will pay us in the future, this does not
   comply with our goals.
   We always keep our promises.

>> Data leak includes
1. Full emloyeers personal data
2. Network information
3. Schemes of buildings, active project information, architect details and contracts,
4. Finance info


>>> How to contact with us?
   1. Download and install TOR Browser (https://www.torproject.org/).
   2. Open http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnzxid.onion/7NT6LXKC1XQHW5039BLOV.

>>> Warning! Recovery recommendations.
   We strongly recommend you to do not MODIFY or REPAIR your files, that will damage them.
```

*Figure 17: Version 1.2 Ransom Note*

## Conclusion

The BlackMatter ransomware group is actively compromising victims and evolving their ransomware tooling and processes; network defenders need to remain just as vigilant. Cipher Tech's ACCE solution leverages more than a decade of defense

expertise to provide a service that can support your incident response and malware analysis teams in automating the process of identifying, unpacking, deobfuscating, decrypting, and parsing supported adversarial tooling.

Cipher Tech will continue to monitor the ransomware threat landscape and do our part to inform defenders on the threats to their networks and their customers' data.

## IOCs

### Hashes (MD5)

#### Version 1.2

18b8ab6af00f387f98b6d7f20253b87b
598c53bfef81e489375f09792e487f1a
639bb7abbd9bc6a9c275d0bf9555b610
6de1ba98baeda1f31e128f982e3878eb
9c26a90a84078a406babb3543d79b8b7
a55bc3368a10ca5a92c1c9ecae97ced9
ad260da314d2f8f3f1531cc5779cbba9
ba375d0625001102fc1f2ccb6f582d91
d0512f2063cbd79fb0f770817cc81ab3
e1f8b95beb02cd39e55cd8b31419b10f
e6b0276bc3f541d8ff1ebb1b59c8bd29
ed74126d9234ac9c1dd21483e82a0dff

#### Version 1.6

01aef1c692a50a9d0e0369a58b1516ff
10aa058a3ac49e016cad7987b8e09886
1dd464cbb3fbd6881eef3f05b8b1fbd5
3317daace715dc332622d883091cf68b

#### Version 1.9

f1c260c31b9d3f9ff54a142d508ec602

#### Version 2.0

38035325b785329e3f618b2a0b90eb75
7b125a148ce0e0c126b95395dbf02b0e
ead753c057b5c3888ed2484013400b82
38035325b785329e3f618b2a0b90eb75

#### C2 URLs

http://mojobiden.com
http://nowautomation.com
http://paymenthacks.com

#### TOR Ransom URLs

http://supp24maprinktc7uizgfyqhisx7lkszb6ogh6lwdzpac23w3mh4tvyd.onion/LEOYRMQLSRHFGFGYWF2T5
http://supp24maprinktc7uizgfyqhisx7lkszb6ogh6lwdzpac23w3mh4tvyd.onion/OR7OTLBK8D5UVHZ0Q
http://supp24maprinktc7uizgfyqhisx7lkszb6ogh6lwdzpac23w3mh4tvyd.onion/OYPF561W4U8HVA0NLVCKJCZB
http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnzxid.onion/7NT6LXKC1XQHW5039BLOV
http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnzxid.onion/EBVCVJNCPM6A3NKJ
http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnzxid.onion/GDBJS76DH3D4IKQD2QO7R
http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnzxid.onion/O3KTUJZRE6CB4Q1OBR

**TOR Blog URLs**

http://blackmax7su6mbwtcyo3xwtpfxpm356jjqrs34y4crcytpw7mifuedyd.onion/xscSyb9oue/b2ec5065190ebe423b201a9f2af97bb

---

Source: https://www.ciphertechsolutions.com/rapidly-evolving-blackmatter-ransomware-tactics/