

# Rapidly Evolving BlackMatter Ransomware Tactics

[ciphertechsolutions.com/rapidly-evolving-blackmatter-ransomware-tactics/](https://ciphertechsolutions.com/rapidly-evolving-blackmatter-ransomware-tactics/)



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

<pre> 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(&amp;bot_company, config_data-&gt;bot_company, int_32);             LODWORD(int_8) = 8;             memcpy(&amp;flag_encrypt_odd, &amp;config_data-&gt;flags, int_8);         }     } </pre>	<pre> 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-&gt;bot_company, int_32);             LODWORD(int_9) = 9;             memcpy(&amp;flag_encrypt_odd, &amp;config_data-&gt;flags, int_9);         }     } </pre>
--	---

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	config	struc ; (sizeo
00000000	rsa_modulus	db 128 dup(?)	00000000	rsa_modulus	db 128 dup(?)
00000080	bot_company	db 16 dup(?)	00000080	bot_company	db 16 dup(?)
00000090	aes_network_key	db 16 dup(?)	00000090	aes_network_key	db 16 dup(?)
000000A0	encrypt_odd	db ?	000000A0	encrypt_odd	db ?
000000A1	authenticate	db ?	000000A1	authenticate	db ?
000000A2	volume_mount	db ?	000000A2	volume_mount	db ?
000000A3	network_rsrcs	db ?	000000A3	network_rsrcs	db ?
000000A4	terminate_procs	db ?	000000A4	terminate_procs	db ?
000000A5	delete_services	db ?	000000A5	delete_services	db ?
000000A6	create_mutex	db ?	000000A6	create_mutex	db ?
000000A7	send_status	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 )
    {
        wcsncpy(SubStr, L"PDF");
        result = wcsstr(pszBuffer, SubStr);
        if ( !result )
        {
            wcsncpy(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 ; (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 ?

```

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	config	struc ; (size
00000000	rsa_modulus	db 128 dup(?)	00000000	rsa_modulus	db 128 dup(?)
00000080	bot_company	db 16 dup(?)	00000080	bot_company	db 16 dup(?)
00000090	aes_network_key	db 16 dup(?)	00000090	aes_network_key	db 16 dup(?)
000000A0	encrypt_odd	db ?	000000A0	encrypt_odd	db ?
000000A1	authenticate	db ?	000000A1	authenticate	db ?
000000A2	volume_mount	db ?	000000A2	volume_mount	db ?
000000A3	network_rsracs	db ?	000000A3	network_rsracs	db ?
000000A4	terminate_procs	db ?	000000A4	terminate_procs	db ?
000000A5	delete_services	db ?	000000A5	delete_services	db ?
000000A6	create_mutex	db ?	000000A6	create_mutex	db ?
000000A7	send_status	db ?	000000A7	print_ransom	db ?
000000A8	bypass_directories	dd ?	000000A8	send_status	db ?
000000AC	bypass_files	dd ?	000000A9	bypass_directories	dd ?
000000B0	bypass_extensions	dd ?	000000AD	bypass_files	dd ?
000000B4	UNUSED	dd ?	000000B1	bypass_extensions	dd ?
000000B8	process_names	dd ?	000000B5	invalid_hostnames	dd ?
000000BC	service_names	dd ?	000000B9	UNUSED	dd ?
000000C0	send_urls	dd ?	000000BD	process_names	dd ?
000000C4	user_creds	dd ?	000000C1	service_names	dd ?
000000C8	ransom	dd ?	000000C5	send_urls	dd ?
000000CC	config	ends	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).







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://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnxid.onion/7NT6LXKC1XQHW5039BLOV>  
<http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnxid.onion/EBVCVJNCPM6A3NKJ>  
<http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnxid.onion/GDBJS76DH3D4IKQD2QO7R>  
<http://supp24yy6a66hwszu2piygicgwzdtbwftb76htfj7vnip3getgqnxid.onion/O3KTUJZRE6CB4Q1OBR>

TOR Blog URLs

<http://blackmax7su6mbwtcyo3xwtpfxpm356jjqrs34y4rcrytpw7mifuedyd.onion/xscSyb9oue/b2ec5065190ebe423b201a9f2af97bb7>