Back in Black... Basta

zscaler.com/blogs/security-research/back-black-basta

Key Points

- BlackBasta emerged in February 2022 with double extortion ransomware attacks against organizations
- The threat group exfiltrates sensitive information from organizations before performing file encryption and demanding a ransom payment
- The previous version of BlackBasta shared many similarities to the now defunct Conti ransomware, although the malware code itself was novel
- In November 2022, BlackBasta ransomware received significant updates including the file encryption algorithms, introduction of stack-based string obfuscation, and per victim file extensions
- The ransomware code modifications are likely an attempt to better evade antivirus and EDR detection

Zscaler ThreatLabz has been tracking prominent ransomware families and their tactics, techniques and procedures (TTPs) including the BlackBasta ransomware family. On November 16, 2022, ThreatLabz identified new samples of the BlackBasta ransomware that had significantly lower antivirus detection rates. The latest BlackBasta code has numerous differences compared to the original BlackBasta ransomware. The changes from the previous version include replacing the file encryption algorithms and switching from the GNU Multiple Precision Arithmetic Library (GMP) to the Crypto++ encryption library. Many of the malware's strings have been obfuscated and the filenames have been randomized, which may hinder static-based antivirus detection and behavioral-based EDR detection. This blog focuses on these recent changes to BlackBasta. Since the current BlackBasta codebase is quite different from the original, ThreatLabz refers to this new version as BlackBasta 2.0.

Technical Analysis

The following sections analyze the changes to the BlackBasta ransomware including the string obfuscation, file encryption and compare various features that have been added, removed or modified.

String Obfuscation

Similar to Conti ransomware, the BlackBasta ransomware developer appears to be experimenting with stack-based string obfuscation using ADVObfuscator. Figure 1 shows an example obfuscated string that is constructed on the stack and decoded using an XOR operation with a single byte.

```
.text:1000ABD9 33 C0                                        xor     eax, eax
.text:1000ABDB B2 5B                                        mov     dl, 5Bh ; '['
.text:1000ABDD C7 45 A6 34 00 35 00                         mov     [ebp+var_5A], 350034h
.text:1000ABE4 C7 45 AA 3E 00 7B 00                         mov     [ebp+var_56], 7B003Eh
.text:1000ABEB 33 C9                                        xor     ecx, ecx
.text:1000ABED C7 45 AE 2F 00 32 00                         mov     [ebp+var_52], 32002Fh
.text:1000ABF4 C7 45 B2 36 00 3E 00                         mov     [ebp+var_4E], 3E0036h
.text:1000ABFB C7 45 B6 61 00 7B 00                         mov     [ebp+var_4A], 7B0061h
.text:1000AC02 C7 45 BA 7E 00 75 00                         mov     [ebp+var_46], 75007Eh
.text:1000AC09 C7 45 BE 6F 00 3D 00                         mov     [ebp+var_42], 3D006Fh
.text:1000AC10 C7 45 C2 7B 00 28 00                         mov     [ebp+var_3E], 28007Bh
.text:1000AC17 C7 45 C6 3E 00 38 00                         mov     [ebp+var_3A], 38003Eh
.text:1000AC1E C7 45 CA 34 00 35 00                         mov     [ebp+var_36], 350034h
.text:1000AC25 C7 45 CE 3F 00 28 00                         mov     [ebp+var_32], 28003Fh
.text:1000AC2C 66 89 45 D2                                  mov     [ebp+var_2E], ax
.text:1000AC30
.text:1000AC30                          loc_1000AC30:                           ; CODE XREF: VisibleEntry
.text:1000AC30 66 0F BE C2                                  movsx   ax, dl
.text:1000AC34 66 31 44 4D A4                               xor     word ptr [ebp+ecx*2+Format+2], ax
```

Figure 1. BlackBasta 2.0 stack-based string obfuscation example

Currently, not all strings in the ransomware are obfuscated, but it is likely that more strings will be obfuscated soon.

File Encryption

Perhaps the most significant modifications in BlackBasta 2.0 is to the encryption algorithms. Previous versions of BlackBasta ransomware used a per victim asymmetric 4,096-bit RSA public key and a per file ChaCha20 symmetric key. The RSA algorithm was implemented using the GNU Multiple Precision Arithmetic Library (GMP). In the latest version of BlackBasta ransomware, the encryption algorithms have been replaced with Elliptic Curve Cryptography (ECC) and XChaCha20. The encryption library used to implement these algorithms in BlackBasta 2.0 is Crypto++. The elliptic curve used by BlackBasta 2.0 is NIST P-521 (aka secp521r1). An example hardcoded NIST P-521 public key embedded in a BlackBasta 2.0 sample is shown below:

```
Public-Key: (521 bit)
pub:
    04:00:52:1f:d8:b3:65:b7:9c:30:bd:fa:1c:88:cc:
    77:77:81:f6:50:9d:d9:17:8d:17:d8:fa:3a:8c:b0:
    f2:6f:87:21:0c:95:db:94:f5:9c:bf:fd:ca:f0:8d:
    19:6a:9c:2f:9f:4b:96:20:31:95:41:54:3e:92:43:
    ed:7b:d1:81:8c:58:78:01:2e:31:b8:02:7a:c1:b9:
    7f:2f:b4:b2:ba:aa:df:ed:68:a2:df:eb:90:4a:4f:
    da:28:10:db:f5:ae:12:08:cf:dd:1f:10:80:48:00:
    32:38:1d:23:40:0c:ca:05:2c:5c:d2:79:1d:ae:8f:
    0a:74:a1:1c:79:b3:0c:38:21:aa:94:1a:4f

ASN1 OID: secp521r1
NIST CURVE: P-521
writing EC key
-----BEGIN PUBLIC KEY-----
MIGbMBAGByqGSM49AgEGBSuBBAAjA4GGAAQAUh/Ys2W3nDC9+hyIzHd3gfZQndkX
jRfY+jqMsPJvhyEMlduU9Zy//crwjRlqnC+fS5YgMZVBVD6SQ+170YGMWHgBLjG4
AnrBuX8vtLK6qt/taKLf65BKT9ooENv1rhIIz90fEIBIADI4HSNADMoFLFzSeR2u
jwp0oRx5sww4IaqUGk8=
-----END PUBLIC KEY-----
```

The encryption process used by BlackBasta 2.0 leverages the Crypto++ Elliptic Curve Integrated Encryption Scheme (ECIES) in Diffie-Hellman Augmented Encryption Scheme (DHAES) mode (also known as DHIES to avoid confusion with the Advanced Encryption Standard) to generate a per file XChaCha20 and a hash-based message authentication code (HMAC). BlackBasta appends a 314-byte footer to files after encryption has been completed as shown below in Figure 2.

```
00000810  6f 77 2e 63 6c 6f 73 65  28 29 7d 29 3b 0a 04 00  |ow.close()});...|
00000820  16 74 3a bb 39 48 b2 01  fe 72 2e 14 86 41 ae 1b  |.t:.9H...r...A..|
00000830  6f c6 cf da 59 56 b6 23  b1 e7 1e c4 93 a6 8f fd  |o...YV.#........|
00000840  d6 0d f0 37 9a 67 11 96  2c 1e a7 4a c1 ca 24 10  |...7.g..,..J..$.|
00000850  c0 ab 98 6c 7c bc 6c 48  57 ab f9 a5 2d ad 04 81  |...l|.lHW...-...|
00000860  ef 01 64 5c bb e5 8a 27  3c 52 06 92 a3 be de 43  |..d\...'<R.....C|
00000870  b5 e7 7a 0c b6 fd 3e 59  46 5c 04 c7 32 5e a8 37  |..z...>YF\..2^.7|
00000880  c3 c1 97 3b f9 99 0b 59  97 05 ac 6e 60 06 bd 67  |...;...Y...n`..g|
00000890  35 c6 9f fb 1e 26 a3 20  11 8f e8 de 1c 68 59 6e  |5....&. .....hYn|
000008a0  b3 88 65 c5 43 0e c7 f0  e5 d6 ca cf 8e 83 4b 07  |..e.C.........K.|
000008b0  b4 e5 d4 8d 4f 2f fc 1e  5e 54 81 4e 2b 3b 6c a6  |....O/..^T.N+;l.|
000008c0  3b 74 f1 a9 0c 12 e6 68  59 f3 79 3a 1e 56 07 a3  |;t.....hY.y:.V..|
000008d0  d6 02 e3 77 7f 14 6f 95  f2 eb 53 2a 52 ae 3f 50  |...w..o...S*R.?P|
000008e0  72 14 94 7a 12 fa 3c ea  95 9a e3 ed 23 45 da 00  |r..z..<.....#E..|
000008f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000940  00 00 00 00 00 00 00 00  00 00 d1 00 6a 34 66 74  |............j4ft|
00000950  6e 77 7a 78 62 72 66 00                           |nwzxbrf.|
```

Figure 2. Example BlackBasta 2.0 encrypted file footer

The first 133-bytes (in blue) are an ephemeral NIST P-521 public key generated per file. The next 56 bytes are an encrypted per file XChaCha20 32-byte key and 24-byte nonce (in green), followed by a 20-byte HMAC (in red). This is followed by NULL byte padding and a two-byte value (in orange) for the size of the cryptographic material. The last 12 bytes (in purple) are a marker (e.g., **j4ftnwzxbrf**), which changes per victim that the BlackBasta decryption tool can use to identify encrypted files.

The encryption process starts by generating an ephemeral NIST P-521 key pair. The corresponding private key is then used to generate a shared secret with the hardcoded public key using the Diffie-Hellman algorithm. The result is passed to the key derivation function KDF2 to produce 72 pseudorandom bytes. The first 16-bytes are used as a HMAC key and the subsequent 56 bytes are used as an XOR key to encrypt the file's XChaCha20 key and nonce (shown above in green). The per file XChaCha20 key and nonce are generated using the Crypto++ random number generator library. The HMAC is calculated with the ciphertext using the SHA1 hash algorithm. The result can be used for message verification with the 20 bytes in the footer (shown in red).

To optimize for speed, BlackBasta encrypts files differently with XChaCha20 based on the file's size. If the file is less than 5,000 bytes the full file is encrypted in blocks of 64 bytes. If the file size is greater than 64 bytes and not an even multiple of 64 bytes, the last 64 byte block will not be encrypted. If the file size is less than or equal to 1,073,741,824 (0x40000000) bytes (i.e., 1GB), BlackBasta alternates encrypting 64 byte blocks followed by 128 bytes that are skipped (i.e., not encrypted) until the end of the file is reached as shown in Figure 3.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  FA 30 E2 19 2A AE AB 4A 73 1B 42 17 DF F9 9C 9F  ú0â.*®«Js.B.ßùœŸ
00000010  77 B0 D9 A1 1A 46 F5 DB 2A A0 75 C4 13 64 C1 50  w°Ù¡.FõÛ* uÄ.dÁP
00000020  CB 31 12 11 40 B1 EC 03 26 C9 80 05 69 15 28 15  Ë1..@±ì.&É€.i.(.
00000030  58 E2 87 07 21 A7 BC 68 8D 16 1E E6 E5 13 9B 42  Xâ‡.!§¼h...æå.›B
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  FA 30 E2 19 2A AE AB 4A 73 1B 42 17 DF F9 9C 9F  ú0â.*®«Js.B.ßùœŸ
000000D0  77 B0 D9 A1 1A 46 F5 DB 2A A0 75 C4 13 64 C1 50  w°Ù¡.FõÛ* uÄ.dÁP
000000E0  CB 31 12 11 40 B1 EC 03 26 C9 80 05 69 15 28 15  Ë1..@±ì.&É€.i.(.
000000F0  58 E2 87 07 21 A7 BC 68 8D 16 1E E6 E5 13 9B 42  Xâ‡.!§¼h...æå.›B
00000100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 3. Example file with null bytes encrypted by BlackBasta 2.0 ransomware alternating between encrypted and unencrypted blocks

If the file is larger than 1GB, BlackBasta will first encrypt the first 5,064 bytes, skip 6,336 bytes, encrypt 64 bytes, skip 6,336 bytes, and so on until the end of the file has been reached. The XChaCha20 encryption code is shown in Figure 4.

```
.text:10013EB2 51                      push    ecx             ; lpOverlapped
.text:10013EB3 8D 4D EC                lea     ecx, [ebp+NumberOfBytesRead]
.text:10013EB6 89 45 F0                mov     [ebp+lpBuffer], eax
.text:10013EB9 51                      push    ecx             ; lpNumberOfBytesRead
.text:10013EBA 56                      push    esi             ; nNumberOfBytesToRead
.text:10013EBB 8B 75 08                mov     esi, [ebp+hFile]
.text:10013EBE 50                      push    eax             ; lpBuffer
.text:10013EBF 56                      push    esi             ; hFile
.text:10013EC0 66 0F 13 45 D8          movlpd  qword ptr [ebp+Overlapped.Internal], xmm0
.text:10013EC5 C7 45 E8 00 00 00 00    mov     [ebp+Overlapped.hEvent], 0
.text:10013ECC FF 15 50 E0 09 10       call    ds:ReadFile
.text:10013ED2 FF 75 14                push    [ebp+nNumberOfBytesToReadAndWrite]
.text:10013ED5 8B 7D 1C                mov     edi, [ebp+arg_14]
.text:10013ED8 8B 45 F0                mov     eax, [ebp+lpBuffer]
.text:10013EDB 50                      push    eax
.text:10013EDC 50                      push    eax
.text:10013EDD 8D 4F 04                lea     ecx, [edi+4]
.text:10013EE0 E8 EB 6F 02 00          call    CryptoPP__XChaCha20
.text:10013EE5 8D 45 D8                lea     eax, [ebp+Overlapped]
.text:10013EE8 50                      push    eax             ; lpOverlapped
.text:10013EE9 6A 00                   push    0               ; lpNumberOfBytesWritten
.text:10013EEB FF 75 14                push    [ebp+nNumberOfBytesToReadAndWrite] ; nNumberOfBytesToWrite
.text:10013EEE FF 75 F0                push    [ebp+lpBuffer]  ; lpBuffer
.text:10013EF1 56                      push    esi             ; hFile
.text:10013EF2 FF 15 7C E0 09 10       call    ds:WriteFile
```

Figure 4. BlackBasta 2.0 XChaCha20 file encryption code

After encryption is complete, BlackBasta 2.0 renames the filename with a hardcoded per-victim extension such as *.agnkdbd5y*, *.taovhsr3u* or *.tcw9lnz6q*. The previous version of BlackBasta used only *.basta* for the encrypted file extension.

The encrypted ransom files' icon image has also been modified from a white box to a red box as shown in Figure 5.

Original BlackBasta encrypted file icon　　New BlackBasta encrypted file icon

Figure 5. BlackBasta (original and new) encrypted file icon images

While this change is rather small, this may be sufficient to bypass static signatures that antivirus products may use to detect BlackBasta.

Ransom Note

BlackBasta 2.0 has modified the ransom note text as shown in Figure 6.

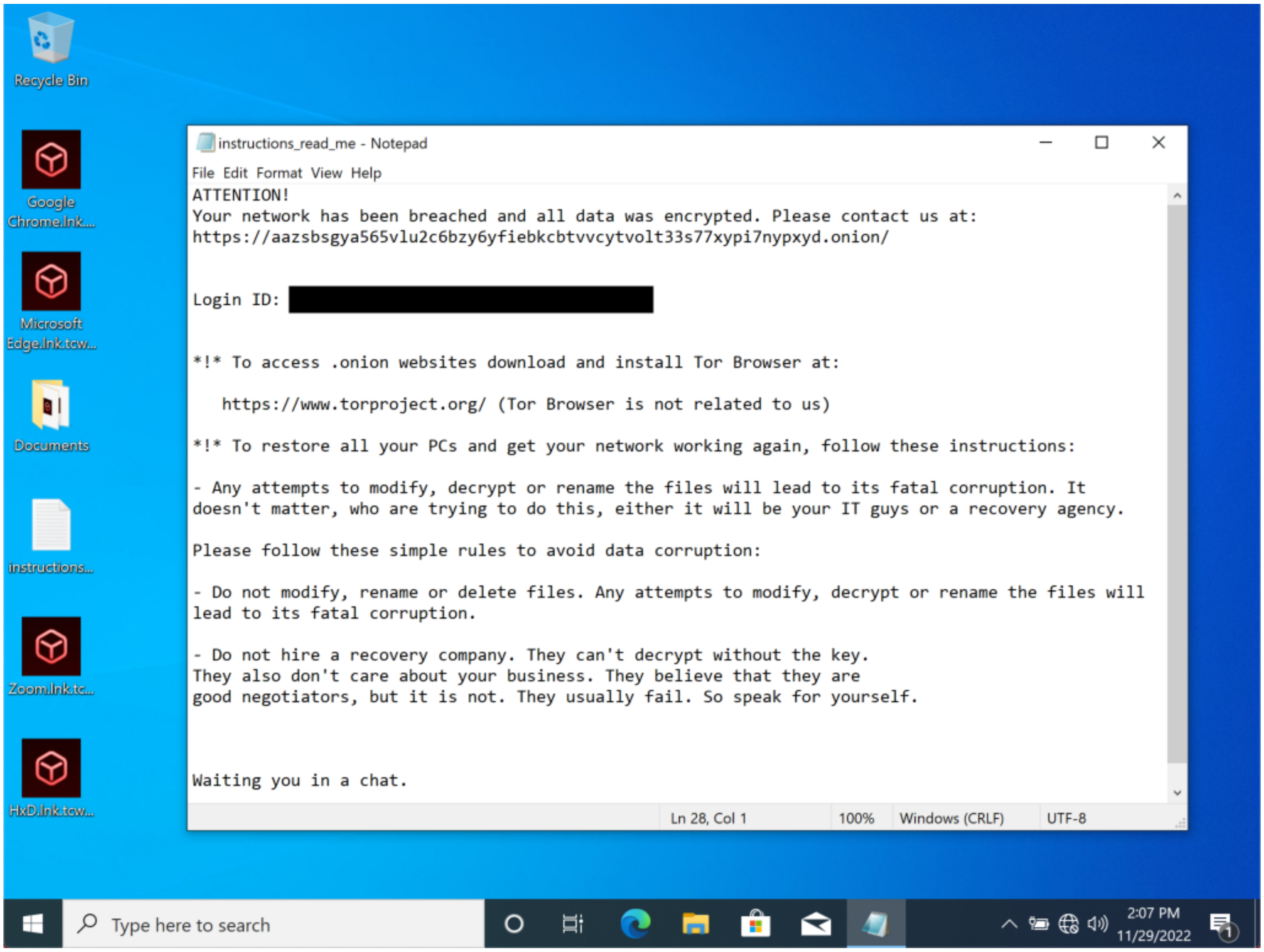


Figure 6. Example BlackBasta 2.0 ransom note (November 2022)

The ransom note filename has also changed from *readme.txt* to *instructions_read_me.txt*. BlackBasta 2.0 opens the ransom note in Windows Notepad via the command *cmd.exe /c start /MAX notepad.exe*.

BlackBasta Feature Parity

Table 1 compares the features between BlackBasta versions 1.0 and 2.0.

Feature	**BlackBasta 1.0**	**BlackBasta 2.0**
Encryption library	GMP	Crypto++
Asymmetric encryption	4,096-bit RSA	NIST P-521

Symmetric encryption	ChaCha20	XChaCha20
Change encrypted file icon	Yes	Yes
Encrypted file extension	.basta	.[a-z0-9]{9}
Change desktop wallpaper	Yes	No
Readme filename	readme.txt	instructions_read_me.txt
String obfuscation	No	Yes
Terminate processes and services	Yes	No
Delete shadow copies	Yes	Yes / No (varies between samples)
Encrypted file icon name	fkdjsadasd.ico	fkdjsadasd.ico
Mutex name	dsajdhas.0	ofijweiuhuewhcsaxs.mutex

Table 1. Feature parity between BlackBasta 1.0 and BlackBasta 2.0

In addition to the aforementioned differences, BlackBasta 2.0 no longer changes the victim's desktop wallpaper, nor terminates processes and services that may interfere with file encryption. The mutex name has also been updated.

The number of command-line parameters has also been modified as shown in Table 2.

Command-line parameter	BlackBasta 1.0	BlackBasta 2.0	Description
-threads	No	Yes	Number of threads to use for encryption
-nomutex	No	Yes	Do not create a mutex

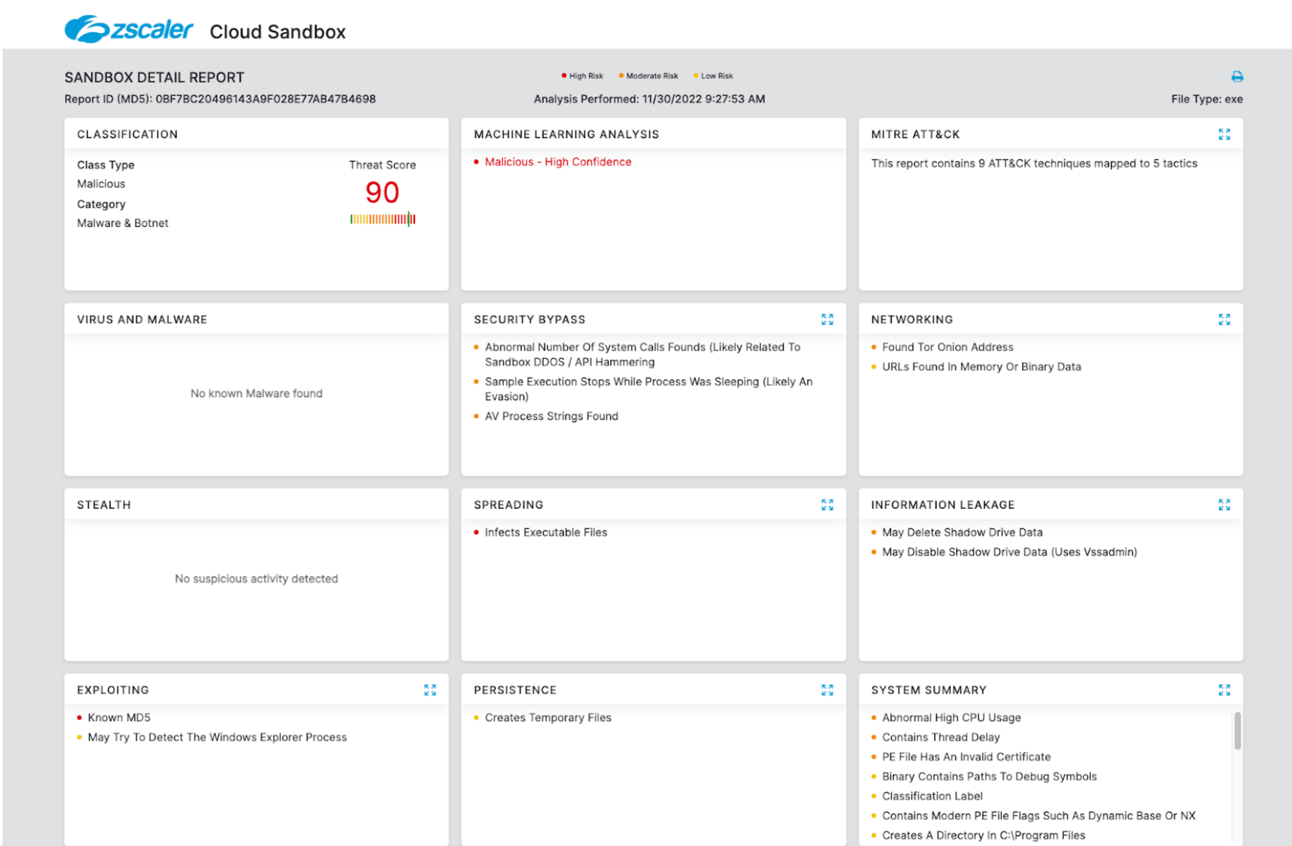
-forcepath	Yes	Yes	Encrypt files in the specified path
-bomb	Yes (in newer builds)	No	Spread via ActiveDirectory and launch ransomware

Table 2. Comparison between BlackBasta command-line parameters

Conclusion

Members of the Conti ransomware group appear to have splintered into multiple threat groups including BlackBasta, which has become one of the most significant ransomware threats. ThreatLabz has observed more than five victims that have been compromised by BlackBasta 2.0 since the new version's release in mid November 2022. This demonstrates that the threat group is very successful at compromising organizations and the latest version of the ransomware will likely enable them to better evade antivirus and EDRs.

Cloud Sandbox Detection



In addition to sandbox detections, Zscaler's multilayered cloud security platform detects indicators related to BlackBasta at various levels with the following threat names:

Indicators of Compromise

SHA256 Hash	Description
e28188e516db1bda9015c30de59a2e91996b67c2e2b44989a6b0f562577fd757	BlackBasta 2.0 sample (executable)
c4c8be0c939e4c24e11bad90549e3951b7969e78056d819425ca53e87af8d8ed	BlackBasta 2.0 sample (executable)
350ba7fca67721c74385faff083914ecdd66ef107a765dfb7ac08b38d5c9c0bd	BlackBasta 2.0 sample (executable)
51eb749d6cbd08baf9d43c2f83abd9d4d86eb5206f62ba43b768251a98ce9d3e	BlackBasta 2.0 sample (DLL)
07117c02a09410f47a326b52c7f17407e63ba5e6ff97277446efc75b862d2799	BlackBasta 2.0 sample (DLL)

These IOCs are also provided in the ThreatLabz GitHub repository here.

Security Research

Ransomware