Popping Blisters for research: An overview of past payloads and exploring recent developments

blog.fox-it.com/2023/11/01/popping-blisters-for-research-an-overview-of-past-payloads-and-exploring-recent-developments/

November 1, 2023

Authored by Mick Koomen

Summary

Blister is a piece of malware that loads a payload embedded inside it. We provide an overview of payloads dropped by the Blister loader based on 137 unpacked samples from the past one and a half years and take a look at recent activity of Blister. The overview shows that since its support for environmental keying, most samples have this feature enabled, indicating that attackers mostly use Blister in a targeted manner. Furthermore, there has been a shift in payload type from Cobalt Strike to Mythic agents, matching with previous reporting. Blister drops the same type of Mythic agent which we thus far cannot link to any public Mythic agents. Another development is that its developers started obfuscating the first stage of Blister, making it more evasive. We provide YARA rules and scripts[1] to help analyze the Mythic agent and the packer we observed with it.

Recap of Blister

Blister is a loader that loads a payload embedded inside it and in the past was observed with activity linked to Evil Corp[2,3]. Matching with public reporting, we have also seen it as a follow-up in SocGholish infections. In the past, we observed Blister mostly dropping Cobalt Strike beacons, yet current developments show a shift to Mythic agents, another red teaming framework.

Elastic Security first documented Blister in December 2021 in a campaign that used malicious installers[4]. It used valid code signatures referencing the company Blist LLC to pose as a legitimate executable, likely leading to the name Blister. That campaign reportedly dropped Cobalt Strike and BitRat.

In 2022, Blister started solely using the x86-64 instruction set, versus including 32-bit as well. Furthermore, RedCanary wrote observing SocGholish dropping Blister[5], which was later confirmed by other vendors as well[6].

In August the same year, we observed a new version of Blister. This update included more configuration options, along with an optional domain hash for environmental keying, allowing attackers to deploy Blister in a targeted manner. Elastic Security recently wrote about this version[7].

2023 initially did not bring new developments for Blister. However, similar to its previous update, we observed development activity in August. Notably, we saw samples with added obfuscation to the first stage of Blister, i.e. the loader component that is injected into a legitimate executable. Additionally, in July, Unit 42[8] observed SocGholish dropping Blister with a Mythic agent.

In summary, 2023 brought new developments for Blister, with added obfuscations to the first stage and a new type of payload. The next part of this blog is divided into two parts: firstly, we look back at previous Blister payloads and configurations, and in the second part, we discuss the recent developments.

Looking back at Blister

In early 2023, we observed a SocGholish infection at our security operations center (SOC). We notified the customer and were given a binary that was related to the infection. This turned out to be a Blister sample, with Cobalt Strike as its payload.

We wrote an extractor that worked on the sample encountered at the SOC, but for certain other Blister samples it did not. It turned out that the sample from the SOC investigation belonged to a version of Blister that was introduced in August, 2022, while older samples had a different configuration. After writing an extractor for these older versions, we made an overview of what Blister had been dropping in roughly the past two years.

The samples we analyzed are all available on VirusTotal, the platform we used to find samples. We focus on 64-bit Blister samples, newer samples are not using 32-bit anymore, as far as we know. In total, we found 137 samples we could unpack, 33 samples with the older version and 104 samples with the newer version from 2022.

In the Appendix, we list these samples, where version 1 and 2 refer to the old and new version respectively. The table is sorted on the first seen date of a sample in VirusTotal, where you clearly see the introduction of the update.

Because we want to keep the tables comprehensible, we have split up the data into four tables. For now, it is important to note that Table 2 provides information per Blister sample we unpacked, including the date it was first uploaded to VirusTotal, the version, the label of the payload it drops, the type of payload, and two configuration flags. Furthermore, to have a list of Blister and payload hashes in clear text in the blog, we included these in Table 6. We also included a more complete data set at https://github.com/fox-it/blister-research.

Discussing payloads

Looking at the dropped payloads, we see that it mostly conforms with what has already been reported. In Figure 1, we provide a timeline based on the first seen date of a sample in VirusTotal and the family of the payload. The observed payloads consist of Cobalt Strike, Mythic, Putty, and a test application. Initially, Blister dropped various flavors of Cobalt Strike and later dropped a Mythic agent, which we refer to as BlisterMythic. Recently, we also observed a packer that unpacks BlisterMythic, which we refer to as MythicPacker. Interestingly, we did not observe any samples drop BitRat.

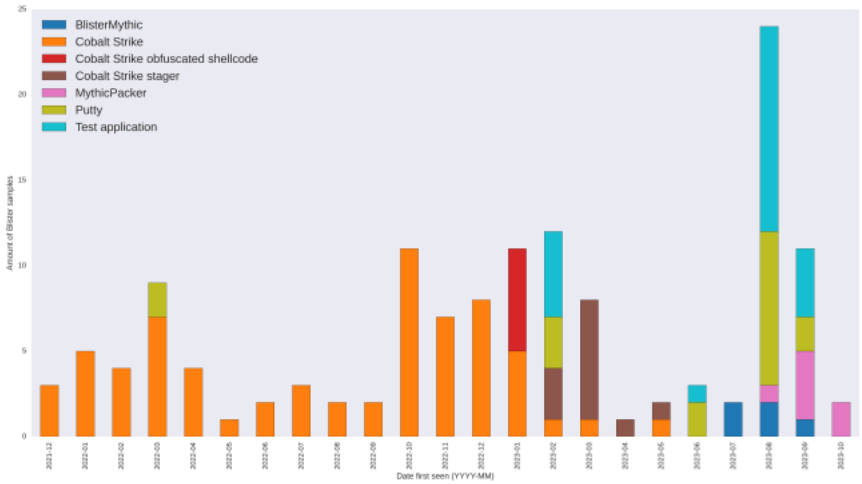

Figure 1, Overview of Blister samples we were able to unpack, based on the first seen date reported in VirusTotal.

From the 137 samples, we were able to retrieve 74 unique payloads. This discrepancy in amount of unique Blister samples versus unique payloads is mainly caused by various Blister samples that drop the same Putty or test application, namely 18 and 22 samples, respectively. This summer has shown a particular increase in test payloads.

Cobalt Strike

Cobalt Strike was dropped through three different types of payloads, generic shellcode, DLL stagers, or obfuscated shellcode. In total, we retrieved 61 beacons, in Table 1 we list the Cobalt Strike watermarks we observed. Watermarks are a unique value linked to a license key. It should be noted that Cobalt Strike watermarks can be changed and hence are not a sound way to identify clusters of activity.

Watermark (decimal)	Watermark (hexadecimal)	Nr. of beacons
206546002	0xc4fa452	2
1580103824	0x5e2e7890	21
1101991775	0x41af0f5f	38

Table 1, Counted Cobalt Strike watermarks observed in beacons dropped by Blister.

The watermark 206546002, though only used twice, shows up in other reports as well, e.g. a report on an Emotet intrusion[9] and a report linking it to Royal, Quantum, and Play ransomware activity[10,11]. The watermark 1580103824 is mentioned in reports on Gootloader[12], but also Cl0p[13] and also is the 9th most common beacon watermark, based on our dataset of Cobalt Strike beacons[14]. Interestingly, 1101991775, the watermark that is most common, is not mentioned in public reporting as far as we can tell.

Cobalt Strike profile generators

In Table 3, we list information on the extracted beacons. In there, we also list the submission path. Most of the submission paths contain `/safebrowsing/` and `/rest/2/meetings`, matching with paths found in SourcePoint[15], a Cobalt Strike command-and-control (C2) profile generator. This is only, however, for the regular shellcode beacons, when we look at the obfuscated shellcode and the DLL stager beacons, it seems to use a different C2 profile. The C2 profiles for these payloads match with another public C2 profile generator[16].

Domain fronting

Some of the beacons are configured to use “domain fronting”, which is a technique that allows malicious actors to hide the true destination of their network traffic and evade detection by security systems. It involves routing malicious traffic through a content delivery network (CDN) or other intermediary server, making it appear as if the traffic is going to a legitimate or benign domain, while in reality, it’s communicating with a malicious C2 server.

Certain beacons have subdomains of fastly[.]net as their C2 server, e.g. backend.int.global.prod.fastly[.]net or python.docs.global.prod.fastly[.]net. However, the domains they connect to are admin.reddit[.]com or admin.wikihow[.]com, which are legitimate domains hosted on a CDN.

Obfuscated shellcode

In five cases, we observed Blister drop Cobalt Strike by first loading obfuscated shellcode. We included a YARA rule for this particular shellcode in the Appendix.

Performing a retrohunt on VirusTotal yielded only 12 samples, with names indicating potential test files and at least one sample dropping Cobalt Strike. We are unsure whether this is an obfuscator solely used by Evil Corp or whether it is used by other threat actors as well.

Figure 2, Layout of particular shellcode, with denoted steps.

The shellcode is fairly simple, we provide an overview of it in Figure 2. The entrypoint is at the start of the buffer, which calls into the decoding stub. This `call` instruction automatically pushes the next instruction's address on the stack, which the decoding stub uses as a starting point to start mutating memory. Figure 3 shows some of these instructions, which are quite distinctive.

```
E 4C 89 52 50                           mov     [rdx+50h], r10
2 4C 8B 7A 58                           mov     r15, [rdx+58h]
6 49 FF C7                              inc     r15
9 4C 89 7A 58                           mov     [rdx+58h], r15
D 81 6A 60 B8 1A 91 0B                  sub     dword ptr [rdx+60h], 0B911AB8h
4 48 C1 4A 68 0D                        ror     qword ptr [rdx+68h], 0Dh
9 81 72 70 B8 76 4A D2                  xor     dword ptr [rdx+70h], 0D24A76B8h
0 48 C1 4A 78 1C                        ror     qword ptr [rdx+78h], 1Ch
5 48 FF 8A 80 00 00 00                  dec     qword ptr [rdx+80h]
C 81 82 88 00 00 00 AA FE 28 27         add     dword ptr [rdx+88h], 2728FEAAh
6 49 89 D0                              mov     r8, rdx
9 49 81 C0 90 00 00 00                  add     r8, 90h
```

Figure 3, Decoding instructions observed in particular shellcode.

At the end of the decoding stub, it either jumps or calls back and then invokes the decryption function. This decryption function uses RC4, but the S-Box is already initialized, thus no key-scheduling algorithm is implemented. Lastly, it jumps to the final payload.

BlisterMythic

Matching with what was already reported by Unit 42[8], Blister recently started using Mythic agents as its payload. Mythic is one of the many red teaming frameworks on GitHub[18]. You can use various agents, which are listed on GitHub as well[19] and can roughly be compared to a Cobalt Strike beacon. It is possible to write your own Mythic agent, as long as you comply with a set of constraints. Thus far, we keep seeing the same Mythic agent, which we discuss in more detail later on. The first sample dropping Mythic agents was uploaded to VirusTotal on July 24th 2023, just days before initial reportings of SocGholish infections leading to Mythic. In Table 4, we provide the C2 information from the observed Mythic agents.

We observed Mythic either as a Portable Executable (PE) or as shellcode. The shellcode seems to be rare and unpacks a PE file which thus far always resulted in a Mythic agent, in our experience. We discuss this packer later on as well and provide scripts that help with retrieving the PE file it packs. We refer to this specific Mythic agent as BlisterMythic and to the packer as MythicPacker.

In Table 5, we list the BlisterMythic C2 servers we were able to find. Interestingly, the domains were all registered at DNSPod. We also observed this in the past with Cobalt Strike domains we linked to Evil Corp. Apart from this, we also see similarities in the domain names used, e.g. domains consisting of two or three words concatenated to each other and using com as top-level domain (TLD).

Test payloads

Besides red team tooling like Mythic and Cobalt Strike, we also observed Putty and a test application as payloads. Running Putty through Blister does not seem logical and is likely linked to testing. It would only result in Putty not touching the disk and running in memory, which in itself is not useful. Additionally, when we look at the domain hashes in the Blister samples, only the Putty and test application samples in some cases share their domain hash.

Blister configurations

We also looked at the configurations of Blister, from this we can to some extent derive how it is used by attackers. Note that the collection also contains “test samples” from the attacker. Except for the more obvious Putty and test application, some samples that dropped Mythic, for instance, could also be linked to testing. We chose to leave out samples that drop Putty or the test application, leaving 97 samples in total. This means that the samples paint a partly biased picture, though we think it is still valuable and provides a view into how Blister is used.

Environmental keying

Since its update in 2022, Blister includes an optional domain hash, that it computes over the DNS search domain of the machine (`ComputerNameDnsDomain`). It only continues executing if the hash matches with its configuration, enabling environmental keying.

By looking at the amount of samples that have domain hash verification enabled, we can say something about how Blister is deployed. From the 66 Blister samples, only 6 samples did not have domain hash verification enabled. This indicates it is mostly used in a targeted manner, corresponding with using SocGholish for initial access and reconnaissance and then deploying Blister, for example.

Persistence

Of the 97 samples, 70 have persistence enabled. For persistence, Blister still uses the same method as described by Elastic Security[20]. It mostly uses IFileOperation COM interface to copy `rundll32.exe` and itself to the Startup folder, this is significant for detection, as it means that these operations are done by the process `DllHost.exe`, not the `rundll32.exe` process that hosts Blister.

Blister trying new things

Blister’s previous update altered the core payload, however, the loader that is injected into the legitimate executable remained unchanged. In August this year, we observed experimental samples on VirusTotal with an obfuscated loader component, hinting at developer activity. Interestingly, we could link these samples to another sample on VirusTotal which solely contained the function body of the loader and another sample that contained a loader with a large set of `INT 3` instructions added to it. Perhaps the developer was experimenting with different mutations to see how it influences the detection rate.

Obfuscating the first stage

Recent samples from September 2023 have the loader obfuscated in the same manner, with bogus instructions and excessive jump instructions. These changes make it harder to detect Blister using YARA, as the loader instructions are now intertwined with junk instructions and sometimes are followed by junk data due to the added jump instructions.

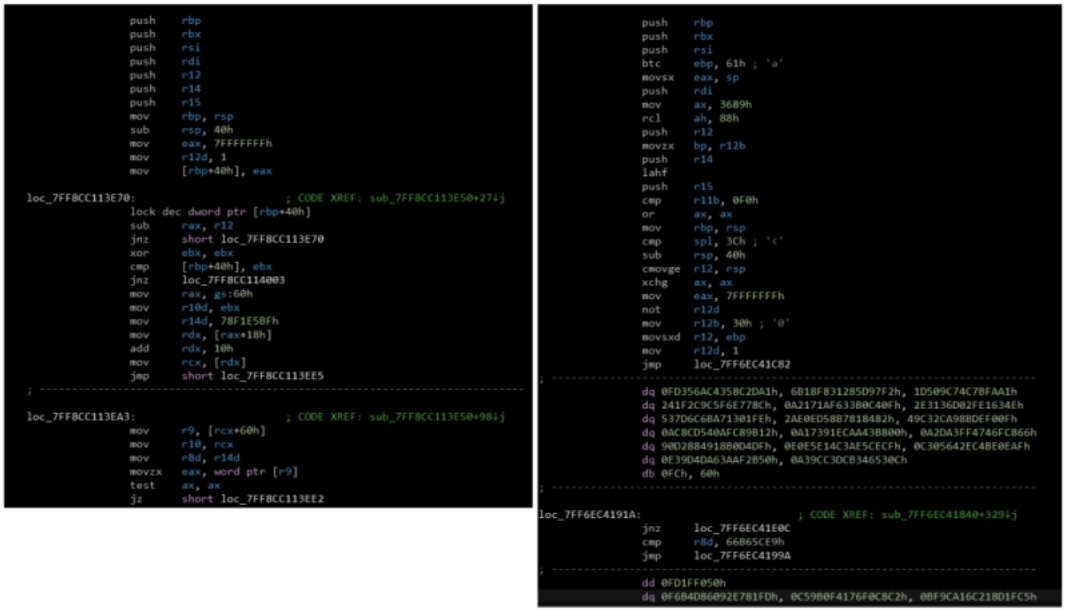

Figure 4, Comparison of two loader components from recent Blister samples, left is without obfuscation and right is with obfuscation.

In Figure 4, we compare the two function bodies of the loader, one body which is normally seen in Blister samples and one obfuscated function body, observed in the recent samples. The comparison shows that naive YARA rules are less likely to trigger on the obfuscated function body. In the Appendix, we provide a Blister rule that tries to detect these obfuscated samples. The added bogus instructions include instructions, such as `btc`, `bts`, `lahf` and `cqo`, bogus instructions we also observed in the Blister core before, see the core component of SHA256 4faf362b3fe403975938e27195959871523689d0bf7fba757ddfa7d00d437fd4, for example.

Dropping Mythic agents

Apart from an obfuscated loader, Mythic agents currently are the payload of choice. In September and October, we found obfuscated Blister samples only dropping Mythic. Certain samples have low or zero detections on VirusTotal[21] at the time of writing, showing that obfuscation does pay off.

We now discuss one sample[22] that drops a shellcode eventually executing a Mythic agent. The shellcode unpacks a PE file and executes it. We provide a YARA rule for this packer in the Appendix, which we refer to as MythicPacker. Based on this rule, we did not find other samples, suggesting it is a custom packer. Until now, we have only seen this packer unpacking Mythic agents.

The dropped Mythic agents are all similar and we cannot link them to any public agents thus far. This could mean that Blister developers created their own Mythic agent, though this is uncertain. We provided a YARA rule that matches on all agents we encountered, a VirusTotal retrohunt over the past year resulted in only four samples, all linked to Blister. We think this Mythic agent is likely custom-made.

```
config_buf = config_buf_start;
dec_bytes = v20;
c = config_size ^ 0x48E12000;
j = (unsigned int)(v4 + 10);
do
{
  if ( config_buf )
  {
    v = *(_DWORD *)config_buf;
    config_buf += 4i64;
    *(_DWORD *)dec_bytes = ((c | v) & ~(c & v)) - z;
  }
  else
  {
    config_buf = config_buf_start;
  }
  dec_bytes += 4;
  i = v7 & 7;
  v7 = (v7 & 7) + 1;
  z = __ROL4__(v, i);
  --j;
}
while ( j );
```

Figure 5, BlisterMythic configuration decryption.

The agents all share a similar structure, namely an encrypted configuration in the `.bss` section of the executable. The agent has an encrypted configuration which is decrypted by XORing the size of the configuration with a constant that differs per sample, it seems. For PE files, we have a Python script that can decrypt a configuration. Figure 5 denotes this decryption loop, where the XOR constant is `0x48E12000`.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000030   32 00 73 73 70 69 63 6C 69 00 63 72 79 70 74 64  2.sspicli.cryptd
00000040   6C 6C 00 6B 65 72 62 65 72 6F 73 00 30 31 32 33  ll.kerberos.0123
00000050   34 35 36 37 38 39 61 62 63 64 65 66 00 25 30 34  456789abcdef.%04
00000060   75 25 30 32 75 25 30 32 75 25 30 32 75 25 30 32  u%02u%02u%02u%02
00000070   75 25 30 32 75 5A 00 30 05 A0 03 01 01 01 00 25  u%02uZ.0. .....%
00000080   73 00 25 75 00 30 78 25 78 00 5F 55 73 65 72 33  s.%u.0x%x._User3
00000090   32 4C 6F 67 6F 6E 50 72 6F 63 65 73 73 00 32 30  2LogonProcess.20
000000A0   33 37 30 39 31 33 30 32 34 38 30 35 5A 00 6B 65  3709130248052.ke
000000B0   72 6E 65 6C 33 32 00 61 64 76 61 70 69 33 32 00  rnel32.advapi32.
000000C0   77 00 69 00 6E 00 6C 00 6F 00 67 00 6F 00 6E 00  w.i.n.l.o.g.o.n.
000000D0   2E 00 65 00 78 00 65 00 00 00 6E 74 64 6C 6C 00  ..e.x.e...ntdll.
000000E0   6C 19 00 00 00 00 00 00 4A 4E 6F 64 18 19 00 00  l.......JNod....
000000F0   00 00 00 00 85 AC 99 9C 30 00 00 00 24 00 00 00  .....¬™œ0...$...
00000100   89 33 3D 0B 38 64 36 63 61 62 34 37 2D 61 32 65  ‰3=.8d6cab47-a2e
00000110   65 2D 34 66 39 63 2D 39 36 32 61 2D 64 32 30 34  e-4f9c-962a-d204
00000120   34 66 64 34 36 31 62 32 20 00 00 00 14 00 00 00  4fd461b2 .......
00000130   AC E9 1A 17 7B 27 64 69 73 61 62 6C 65 5F 65 74  ¬é..{'disable_et
00000140   77 27 3A 20 27 32 27 7D B0 18 00 00 A4 18 00 00  w': '2'}°...¤...
00000150   E0 B0 D6 C5 7B 27 47 45 54 27 3A 20 7B 27 53 65  à°ÖÅ{'GET': {'Se
00000160   72 76 65 72 42 6F 64 79 27 3A 20 5B 7B 27 66 75  rverBody': [{'fu
00000170   6E 63 74 69 6F 6E 27 3A 20 27 62 61 73 65 36 34  nction': 'base64
```

Figure 6, Decrypted BlisterMythic configuration

Dumping the configuration results in a binary blob that contains various information, including the C2 server. Figure 6 shows a hexdump of a snippet from the decrypted configuration. We created a script to dump the decrypted configuration of the BlisterMythic agent in PE format and also a script that unpacks MythicPacker shellcode and outputs a reconstructed PE file, see https://github.com/fox-it/blister-research.

Conclusion

In this post, we provided an overview of observed Blister payloads from the past one and a half years on VirusTotal and also gave insight into recent developments. Furthermore, we provided scripts and YARA rules to help analyze Blister and the Mythic agent it drops.

From the analyzed payloads, we see that Cobalt Strike was the favored choice, but that lately this has been replaced by Mythic. Cobalt Strike was mostly dropped as shellcode and briefly run through obfuscated shellcode or a DLL stager. Apart from Cobalt Strike and Mythic, we saw that Blister test samples are uploaded to VirusTotal as well.

The custom Mythic agent together with the obfuscated loader, are new Blister developments that happened in the past months. It is likely that its developers were aware that the loader component was still a weak spot in terms of static detection. Additionally, throughout the years, Cobalt Strike has received a lot of attention from the security community, with available dumpers and C2 feeds readily available. Mythic is not as popular and allows you to write your own agent, making it an appropriate replacement for now.

References

Appendix

YARA rules

```
rule shellcode_obfuscator
{
    meta:
        os = "Windows"
        arch = "x86-64"
        description = "Detects shellcode packed with unknown obfuscator observed in Blister samples."
        reference_sample = "178ffbdd0876b99ad1c2d2097d9cf776eca56b540a36c8826b400cd9d5514566"
    strings:
        $rol_ror = { 48 C1 ?? ?? ?? 48 C1 ?? ?? ?? 48 C1 ?? ?? ?? }
        $mov_rol_mov = { 4d ?? ?? ?? 49 c1 ?? ?? ?? 4d ?? ?? ?? }
        $jmp = { 49 81 ?? ?? ?? ?? ?? 41 ?? }
    condition:
        #rol_ror > 60 and $jmp and filesize < 2MB and #mov_rol_mov > 60
}

import "pe"
import "math"

rule blister_x64_windows_loader {
    meta:
        os = "Windows"
        arch = "x86-64"
        family = "Blister"
        description = "Detects Blister loader component injected into legitimate executables."
        reference_sample = "343728792ed1e40173f1e9c5f3af894feacd470a9cdc72e4f62c0dc9cbf63fc1,
8d53dc0857fa634414f84ad06d18092dedeb110689a08426f08cb1894c2212d4, a5fc8d9f9f4098e2cecb3afc66d8158b032ce81e0be614d216c9deaf20e888ac"
    strings:
        // 65 48 8B 04 25 60 00 00 00                          mov     rax, gs:60h
        $inst_1 = {65 48 8B 04 25 60 00 00 00}
        // 48 8D 87 44 6D 00 00                                lea     rax, [rdi+6D44h]
        $inst_2 = {48 8D 87 44 6D 00 00}
        // 44 69 C8 95 E9 D1 5B                                imul    r9d, eax, 5BD1E995h
        $inst_3 = {44 ?? ?? 95 E9 D1 5B}
        // 41 81 F9 94 85 09 64                                cmp     r9d, 64098594h
        $inst_4 = {41 ?? ?? 94 85 09 64}
        // B8 FF FF FF 7F                                      mov     eax, 7FFFFFFFh
        $inst_5 = {B8 FF FF FF 7F}
        // 48 8D 4D 48                                         lea     rcx, [rbp+48h]
        $inst_6 = {48 8D 4D 48}
    condition:
        uint16(0) == 0x5A4D and
        all of ($inst_*) and
        pe.number_of_resources > 0 and
        for any i in (0..pe.number_of_resources - 1):
            ( (math.entropy(pe.resources[i].offset, pe.resources[i].length) > 6) and
                pe.resources[i].length > 200000
            )
}

rule blister_mythic_payload {
    meta:
        os = "Windows"
        arch = "x86-64"
        family = "BlisterMythic"
        description = "Detects specific Mythic agent dropped by Blister."
        reference_samples = "2fd38f6329b9b2c5e0379a445e81ece43fe0372dec260c1a17eefba6df9ffd55,
3d2499e5c9b46f1f144cfbbd4a2c8ca50a3c109496a936550cbb463edf08cd79, ab7cab5192f0bef148670338136b0d3affe8ae0845e0590228929aef70cb9b8b,
f89cfbc1d984d01c57dd1c3e8c92c7debc2beb5a2a43c1df028269a843525a38"
    strings:
        $start_inst = { 48 83 EC 28 B? [4-8] E8 ?? ?? 00 00 }
        $for_inst = { 48 2B C8 0F 1F 00 C6 04 01 00 48 2D 00 10 00 00 }
    condition:
        all of them
}

rule mythic_packer
{
    meta:
        os = "Windows"
        arch = "x86-64"
        family = "MythicPacker"
        description = "Detects specific PE packer dropped by Blister."
        reference_samples = "9a08d2db7d0bd7d4251533551d4def0f5ee52e67dff13a2924191c8258573024,
759ac6e54801e7171de39e637b9bb525198057c51c1634b09450b64e8ef47255"
    strings:
        // 41 81 38 72 47 65 74        cmp     dword ptr [r8], 74654772h
        $a = { 41 ?? ?? 72 47 65 74 }
        // 41 81 38 72 4C 6F 61        cmp     dword ptr [r8], 616F4C72h
        $b = { 41 ?? ?? 72 4C 6F 61 }
```

```
        // B8 01 00 00 00             mov     eax, 1
        // C3                         retn
        $c = { B8 01 00 00 00 C3 }
    condition:
        all of them and uint8(0) == 0x48
}
```

Blister payloads listing

First seen	Version	Payload family	Payload type	Environmental keying	Persistence
2021-12-03	1	Cobalt Strike	shellcode	N/a	0
2021-12-05	1	Cobalt Strike	shellcode	N/a	0
2021-12-14	1	Cobalt Strike	shellcode	N/a	0
2022-01-10	1	Cobalt Strike	shellcode	N/a	1
2022-01-11	1	Cobalt Strike	shellcode	N/a	1
2022-01-19	1	Cobalt Strike	shellcode	N/a	1
2022-01-19	1	Cobalt Strike	shellcode	N/a	1
2022-01-31	1	Cobalt Strike	shellcode	N/a	1
2022-02-14	1	Cobalt Strike	shellcode	N/a	1
2022-02-17	1	Cobalt Strike	shellcode	N/a	1
2022-02-22	1	Cobalt Strike	shellcode	N/a	1
2022-02-26	1	Cobalt Strike	shellcode	N/a	1
2022-03-10	1	Cobalt Strike	shellcode	N/a	1
2022-03-14	1	Cobalt Strike	shellcode	N/a	1
2022-03-15	1	Cobalt Strike	shellcode	N/a	0
2022-03-15	1	Cobalt Strike	shellcode	N/a	0
2022-03-18	1	Cobalt Strike	shellcode	N/a	0
2022-03-18	1	Cobalt Strike	shellcode	N/a	1
2022-03-24	1	Putty	exe	N/a	0
2022-03-24	1	Putty	exe	N/a	0
2022-03-30	1	Cobalt Strike	shellcode	N/a	1
2022-04-01	1	Cobalt Strike	shellcode	N/a	0
2022-04-11	1	Cobalt Strike	shellcode	N/a	1
2022-04-22	1	Cobalt Strike	shellcode	N/a	1
2022-04-25	1	Cobalt Strike	shellcode	N/a	0
2022-06-01	1	Cobalt Strike	shellcode	N/a	0
2022-06-02	1	Cobalt Strike	shellcode	N/a	1
2022-06-14	1	Cobalt Strike	shellcode	N/a	1
2022-07-04	1	Cobalt Strike	shellcode	N/a	1
2022-07-19	1	Cobalt Strike	shellcode	N/a	0
2022-07-21	1	Cobalt Strike	shellcode	N/a	0
2022-08-05	1	Cobalt Strike	shellcode	N/a	1
2022-08-29	2	Cobalt Strike	shellcode	0	1
2022-09-02	2	Cobalt Strike	shellcode	0	0

First seen	Version	Payload family	Payload type	Environmental keying	Persistence
2022-09-29	2	Cobalt Strike	shellcode	1	0
2022-10-18	2	Cobalt Strike	shellcode	1	1
2022-10-18	2	Cobalt Strike	shellcode	1	1
2022-10-18	2	Cobalt Strike	shellcode	1	0
2022-10-18	2	Cobalt Strike	shellcode	1	1
2022-10-21	2	Cobalt Strike	shellcode	1	1
2022-10-21	2	Cobalt Strike	shellcode	1	0
2022-10-24	2	Cobalt Strike	shellcode	1	1
2022-10-26	2	Cobalt Strike	shellcode	1	1
2022-10-26	2	Cobalt Strike	shellcode	1	1
2022-10-28	2	Cobalt Strike	shellcode	1	0
2022-10-31	2	Cobalt Strike	shellcode	1	1
2022-11-02	2	Cobalt Strike	shellcode	1	1
2022-11-03	2	Cobalt Strike	shellcode	1	1
2022-11-07	2	Cobalt Strike	shellcode	1	1
2022-11-08	2	Cobalt Strike	shellcode	1	1
2022-11-17	2	Cobalt Strike	shellcode	1	1
2022-11-22	2	Cobalt Strike	shellcode	1	1
2022-11-30	2	Cobalt Strike	shellcode	1	1
2022-12-01	2	Cobalt Strike	shellcode	1	1
2022-12-01	2	Cobalt Strike	shellcode	1	0
2022-12-01	2	Cobalt Strike	shellcode	1	0
2022-12-02	2	Cobalt Strike	shellcode	1	1
2022-12-05	2	Cobalt Strike	shellcode	1	1
2022-12-12	2	Cobalt Strike	shellcode	1	1
2022-12-13	2	Cobalt Strike	shellcode	1	1
2022-12-23	2	Cobalt Strike	shellcode	1	1
2023-01-06	2	Cobalt Strike	shellcode	1	1
2023-01-16	2	Cobalt Strike obfuscated shellcode	shellcode	1	1
2023-01-16	2	Cobalt Strike obfuscated shellcode	shellcode	1	1
2023-01-16	2	Cobalt Strike obfuscated shellcode	shellcode	1	1
2023-01-17	2	Cobalt Strike	shellcode	0	1
2023-01-17	2	Cobalt Strike obfuscated shellcode	shellcode	1	1
2023-01-20	2	Cobalt Strike obfuscated shellcode	shellcode	1	1
2023-01-20	2	Cobalt Strike obfuscated shellcode	shellcode	1	1
2023-01-24	2	Cobalt Strike	shellcode	1	1
2023-01-26	2	Cobalt Strike	shellcode	1	1
2023-01-26	2	Cobalt Strike	shellcode	1	1
2023-02-02	2	Cobalt Strike	shellcode	1	1

First seen	Version	Payload family	Payload type	Environmental keying	Persistence
2023-02-02	2	Test application	shellcode	1	0
2023-02-02	2	Test application	shellcode	1	0
2023-02-02	2	Putty	exe	1	0
2023-02-02	2	Test application	shellcode	1	0
2023-02-15	2	Putty	exe	1	0
2023-02-15	2	Test application	shellcode	1	0
2023-02-15	2	Putty	exe	1	0
2023-02-15	2	Test application	shellcode	1	0
2023-02-17	2	Cobalt Strike stager	exe	1	1
2023-02-27	2	Cobalt Strike stager	exe	1	1
2023-02-28	2	Cobalt Strike stager	exe	1	1
2023-03-06	2	Cobalt Strike stager	exe	1	1
2023-03-06	2	Cobalt Strike stager	exe	1	1
2023-03-06	2	Cobalt Strike stager	exe	1	1
2023-03-15	2	Cobalt Strike stager	exe	1	0
2023-03-19	2	Cobalt Strike stager	exe	1	1
2023-03-23	1	Cobalt Strike	shellcode	N/a	1
2023-03-28	2	Cobalt Strike stager	exe	1	1
2023-03-28	2	Cobalt Strike stager	exe	1	0
2023-04-03	2	Cobalt Strike stager	exe	1	1
2023-05-25	2	Cobalt Strike stager	exe	0	1
2023-05-26	2	Cobalt Strike	shellcode	1	1
2023-06-11	2	Test application	shellcode	1	0
2023-06-11	2	Putty	exe	1	0
2023-06-11	2	Putty	exe	1	0
2023-07-24	2	BlisterMythic	exe	1	1
2023-07-27	2	BlisterMythic	exe	1	1
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-09	2	Test application	shellcode	1	0
2023-08-10	2	Putty	shellcode	1	0

First seen	Version	Payload family	Payload type	Environmental keying	Persistence
2023-08-10	2	Putty	shellcode	1	0
2023-08-10	2	Putty	shellcode	1	0
2023-08-10	2	Putty	shellcode	1	0
2023-08-10	2	Putty	shellcode	1	0
2023-08-10	2	Putty	shellcode	1	0
2023-08-10	2	Putty	shellcode	1	0
2023-08-10	2	Putty	shellcode	1	0
2023-08-10	2	Putty	shellcode	1	0
2023-08-11	2	BlisterMythic	exe	1	0
2023-08-15	2	Test application	shellcode	1	0
2023-08-17	2	BlisterMythic	exe	1	1
2023-08-18	2	MythicPacker	shellcode	1	0
2023-09-05	2	MythicPacker	shellcode	0	0
2023-09-05	2	MythicPacker	shellcode	0	1
2023-09-08	2	Test application	shellcode	1	0
2023-09-08	2	Test application	shellcode	1	0
2023-09-08	2	Test application	shellcode	1	0
2023-09-08	2	Putty	shellcode	1	0
2023-09-08	2	Putty	shellcode	1	0
2023-09-08	2	Test application	shellcode	1	0
2023-09-19	2	BlisterMythic	exe	1	1
2023-09-21	2	MythicPacker	shellcode	1	0
2023-09-21	2	MythicPacker	shellcode	1	0
2023-10-03	2	MythicPacker	shellcode	1	0
2023-10-10	2	MythicPacker	shellcode	1	0

Table 2, Information on unpacked Blister samples.

Cobalt Strike beacons

Watermark	Domain	URI
1101991775	albertonne[.]com	/safebrowsing/d4alBmGBO/HafYg4QZaRhMBwuLAjVmSPc
1101991775	astradamus[.]com	/Collect/union/QXMY8BHNIPH7
1101991775	backend.int.global.prod.fastly[.]net	/Detect/devs/NJYO2MUY4V
1101991775	cclastnews[.]com	/safebrowsing/d4alBmGBO/UaIzXMVGvV3tS2OJiKxSzyzbh4u1
1101991775	cdp-chebe6efcxhvd0an.z01.azurefd[.]net	/Detect/devs/NJYO2MUY4V
1101991775	deep-linking[.]com	/safebrowsing/fDeBjO/2hmXORzLK7PkevU1TehrmzD5z9
1101991775	deep-linking[.]com	/safebrowsing/fDeBjO/dMfdNUdgjjii3Ccalh10Mh4qyAFw5mS
1101991775	deep-linking[.]com	/safebrowsing/fDeBjO/vnZNyQrwUjndCPsCUXSaI
1101991775	diggin-fzbvcfcyagemchbq.z01.azurefd[.]net	/restore/how/3RG4G5T87

Watermark	Domain	URI
1101991775	edubosi[.]com	/safebrowsing/bsaGbO6l/ybGoI3wmK2uF9w9aL5qKmnS8lZlWsJqhp
1101991775	e-sistem[.]com	/Detect/devs/NJYO2MUY4V
1101991775	ewebsofts[.]com	/safebrowsing/3Tqo/UMskN3Lh0LyLy8BfpG1Bsvp
1101991775	expreshon[.]com	/safebrowsing/fDeBjO/2hmXORzLK7PkevU1TehrmzD5z9
1101991775	eymenelektronik[.]com	/safebrowsing/dfKa/B58qAhJ0AEF7aNwauoqpAL8
1101991775	gotoknysna.com.global.prod.fastly[.]net	/safebrowsing/fDeBjO/2hmXORzLK7PkevU1TehrmzD5z9
1101991775	henzy-h6hxfpfhcaguhyf5.z01.azurefd[.]net	/Detect/devs/NJYO2MUY4V
1101991775	lepont-edu[.]com	/safebrowsing/dfKa/9T1BuXpqEDg9tx53mQRU6
1101991775	lindecolas[.]com	/safebrowsing/d4alBmGBO/UaIzXMVGvV3tS2OJiKxSzyzbh4u1
1101991775	lodhaamarathane[.]com	/safebrowsing/dfKa/9T1BuXpqEDg9tx53mQRU6
1101991775	mail-adv[.]com	/safebrowsing/bsaGbO6l/dl1sskHxt1uGDGUnLDB5gxn4vYZQK1kaG6
1101991775	mainecottagebythesea[.]com	/functionalStatus/cjdl-CLe4j-XHyiEaDqQx
1101991775	onscenephotos[.]com	/restore/how/3RG4G5T87
1101991775	promedia-usa[.]com	/safebrowsing/d4alBmGBO/HafYg4QZaRhMBwuLAjVmSPc
1101991775	python.docs.global.prod.fastly[.]net	/Collect/union/QXMY8BHNIPH7
1101991775	realitygangnetwork[.]com	/functionalStatus/qPprp9dtVhrGV3R3re5Xy4M2cfQo4wB
1101991775	realitygangnetwork[.]com	/functionalStatus/vFi8EPnc9zJTD0GgRPxggCQAaNb
1101991775	sanfranciscowoodshop[.]com	/safebrowsing/dfKa/GgVYon5zhYu5L7inFbI1MZEv7RGOnsS00b
1101991775	sohopf[.]com	/apply/admin_/99ZSSAHDH
1101991775	spanish-home-sales[.]com	/safebrowsing/d4alBmGBO/EB-9sfMPmsHmH-A7pmll9HbV0g
1101991775	steveandzina[.]com	/safebrowsing/d4alBmGBO/mr3lHbohEvZa0mKDWWdwTV5FIsxh
1101991775	steveandzina[.]com	/safebrowsing/d4alBmGBO/YwTM1CK0mBV1Y7UDagpjP
1101991775	websterbarn[.]com	/safebrowsing/fDeBjO/CGZcHKnX3arVCfFp98k8
1580103824	10.158.128[.]50	
1580103824	bimelectrical[.]com	/safebrowsing/7IAMO/hxNTeZ8lBNYqjAsQ2tBRS
1580103824	bimelectrical[.]com	/safebrowsing/7IAMO/Jwee0NMJNKn9sDD8sUEem4g8jcB2v44UINpCIj
1580103824	bookmark-tag[.]com	/safebrowsing/eMUgI4Z/3RzgDBAvgg3DQUn8XtN8l
1580103824	braprest[.]com	/safebrowsing/d5pERENa/3tPCoNwoGwXAvV1w1JAS-OOPyVYxL1K2styHFtbXar7ME
1580103824	change-land[.]com	/safebrowsing/TKc3hA/DzwHHcc8y8O9kAS7cl4SDK0e6z0KHKIX9w7
1580103824	change-land[.]com	/safebrowsing/TKc3hA/nLTHCIhzOKpdFp0GFHYBK-0bRwdNDlZz6Qc
1580103824	clippershipintl[.]com	/safebrowsing/sj0IWAb/YhcZADXFB3NHbxFtKgpqBtK9BllJiGEL
1580103824	couponbrothers[.]com	/safebrowsing/Jwjy4/mzAoZyZk7qHIyw3QrEpXij5WFhIo1z8JDUVA0N0
1580103824	electronic-infinity[.]com	/safebrowsing/TKc3hA/t-nAkENGu9rpZ9ebRRXr79b
1580103824	final-work[.]com	/safebrowsing/AvuvAkxsR/8I6ikMUvdNd8HOgMeD0sPfGpwSZEMr
1580103824	geotypico[.]com	/safebrowsing/d5pERENa/f5oBhEk7xS3cXxstp6Kx1G7u3N546UStcg9nEnzJn2k
1580103824	imsensors[.]com	/safebrowsing/eMUgI4Z/BOhKRIMsJsuPnn3IQvgrEc3XLQUB3W
1580103824	intradayinvestment[.]com	/safebrowsing/dpNqi/nXeFgGufr9VqHjDdsIZbw-ZH0
1580103824	medicare-cost[.]com	/safebrowsing/dpNqi/F3QExtY65SvTVK1ewA26

Watermark	Domain	URI
1580103824	optiontradingsignal[.]com	/safebrowsing/dpNqi/7CtHhF-isMMQ6m7NmHYNb0N7E7Fe
1580103824	setechnowork[.]com	/safebrowsing/fBm1b/JbcKDYjMWcQNjn69LnGggFe6mpjn5xOQ
1580103824	sikescomposites[.]com	/safebrowsing/Jwjy4/cmr4tZ7IyFGbgCiof2tHMO
1580103824	technicollit[.]com	/safebrowsing/b0kKKIjr/AzX9ZHB37oJfPsUBUaxBJjzzi132cYRZhUZc81g
1580103824	wasfatsahla[.]com	/safebrowsing/IsXNCJJfH/5x0rUIrn–r85sLJIuEY7C9q
206546002	smutlr[.]com	/functionalStatus/qPprp9dtVhrGV3R3re5Xy4M2cfQo4wB
206546002	spanish-home-sales[.]com	/functionalStatus/fb8ClEdmm-WwYudk-zODoQYB7DX3wQYR

Table 3, Information on observed Cobalt Strike beacons dropped by Blister.

BlisterMythic payloads

Domain	URI
139-177-202-78.ip.linodeusercontent[.]com	/etc.clientlibs/sapdx/front-layer/dist/resources/sapcom/919.9853a7ee629d48b1ddbe.js
23-92-30-58.ip.linodeusercontent[.]com	/etc.clientlibs/sapdx/front-layer/dist/resources/sapcom/919.9853a7ee629d48b1ddbe.js
avidityceIlars[.]com	/etc.clientlibs/sapdx/front-layer/dist/resources/sapcom/919.9853a7ee629d48b1ddbe.js
boxofficeseer[.]com	/s/0.7.8/clarity.js
d1hp6ufzqrj3xv.cloudfront[.]net	/organizations/oauth2/v2.0/authorize
makethumbmoney[.]com	/s/0.7.8/clarity.js
rosevalleylimousine[.]com	/login.sophos.com/B2C_1A_signup_signin/api/SelfAsserted/confirmed

Table 4, Information on observed Mythic agents dropped by Blister.

BlisterMythic C2 servers

IP	Domain
37.1.215[.]57	angelbusinessteam[.]com
92.118.112[.]100	danagroupegypt[.]com
104.238.60[.]11	shchiswear[.]com
172.233.238[.]215	N/a
96.126.111[.]127	N/a
23.239.11[.]145	N/a
45.33.98[.]254	N/a
45.79.199[.]4	N/a
45.56.105[.]98	N/a
149.154.158[.]243	futuretechfarm[.]com
104.243.33[.]161	sms-atc[.]com
104.243.33[.]129	makethumbmoney[.]com
138.124.180[.]241	vectorsandarrows[.]com
94.131.101[.]58	pacatman[.]com
198.58.119[.]214	N/a
185.174.101[.]53	personmetal[.]com
185.45.195[.]30	avidityceIlars[.]com

IP	Domain
185.250.151[.]145	bureaudecreationalienor[.]com
23.227.194[.]115	bitscoinc[.]com
88.119.175[.]140	boxofficeseer[.]com
88.119.175[.]137	thesheenterprise[.]com
37.1.214[.]162	remontisto[.]com
45.66.248[.]99	N/a
88.119.175[.]104	visioquote[.]com
45.66.248[.]13	cannabishang[.]com
92.118.112[.]8	turanmetal[.]com
37.1.211[.]150	lucasdoors[.]com
185.72.8[.]219	displaymercials[.]com
172.232.172[.]128	N/a
82.117.253[.]168	digtupu[.]com
104.238.60[.]112	avblokhutten[.]com
173.44.141[.]34	hom4u[.]com
170.130.165[.]140	rosevalleylimousine[.]com
172.232.172[.]110	N/a
5.8.63[.]79	boezgrt[.]com
172.232.172[.]125	N/a
162.248.224[.]56	hatchdesignsnh[.]com
185.174.101[.]13	formulaautoparts[.]com
23.152.0[.]193	ivermectinorder[.]com
192.169.6[.]200	szdeas[.]com
194.87.32[.]85	licencesolutions[.]com
185.45.195[.]205	motorrungoli[.]com

Table 5, Detected BlisterMythic C2 servers

Blister samples

SHA256	Payload family	Payload SHA256
0a73a9ee3650821352d9c4b46814de8f73fde659cae6b82a11168468becb68d1	Cobalt Strike	397c08f5cdc59085a48541c89d23a8880d4155
0bbf1a3a8dd436fda213bc126b1ad0b8704d47fd8f14c75754694fd47a99526c	BlisterMythic	ab7cab5192f0bef148670338136b0d3affe8ae0
0e8458223b28f24655caf37e5c9a1c01150ac7929e6cb1b11d078670da892a5b	Cobalt Strike	4420bd041ae77fce2116e6bd98f4ed6945514fa
0f07c23f7fe5ff918ee596a7f1df320ed6e7783ff91b68c636531aba949a6f33	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
a3cb53ddd4a5316cb02b7dc4ccd1f615755b46e86a88152a1f8fc59efe170497	Cobalt Strike	e85a2e8995ef37acf15ea79038fae70d4566bd9
a403b82a14b392f8485a22f105c00455b82e7b8a3e7f90f460157811445a8776	Cobalt Strike	e0c0491e45dda838f4ac01b731dd39cc706467
a5fc8d9f9f4098e2cecb3afc66d8158b032ce81e0be614d216c9deaf20e888ac	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
a9ea85481e178cd35ae323410d619e97f49139dcdb2e7da72126775a89a8464f	Cobalt Strike	c7accad7d8da9797788562a3de228186290b0f

SHA256	Payload family	Payload SHA256
ac232e7594ce8fbbe19fc74e34898c562fe9e8f46d4bfddc37aefeb26b85c02b	Cobalt Strike obfuscated shellcode	cef1a88dfc436dab9ae104f0770a434891bbd60
acdaac680e2194dd8fd06f937847440e7ab83ce1760eab028507ee8eba557291	Cobalt Strike	b96d4400e9335d80dedee6f74ffaa4eca9ffce24
ae148315cec7140be397658210173da372790aa38e67e7aa51597e3e746f2cb2	Cobalt Strike	f245b2bc118c3c20ed96c8a9fd0a7b659364f9e
aeecc65ac8f0f6e10e95a898b60b43bf6ba9e2c0f92161956b1725d68482721d	Cobalt Strike	797abd3de3cb4c7a1ceb5de5a95717d84333be
b062dd516cfa972993b6109e68a4a023ccc501c9613634468b2a5a508760873e	Cobalt Strike	122b77fd4d020f99de66bba8346961b565e804
b10db109b64b798f36c717b7a050c017cf4380c3cb9cfeb9acd3822a68201b5b	Cobalt Strike	902d29871d3716113ca2af5caa6745cb4ab9d0
b1d1a972078d40777d88fb4cd6aef1a04f29c5dd916f30a6949b29f53a2d121c	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
b1f3f1c06b1cc9a249403c2863afc132b2d6a07f137166bdd1e4863a0cece5b1	Cobalt Strike	e63807daa9be0228d90135ee707ddf03b00353
b4c746e9a49c058ae3843799cdd6a3bb5fe14b413b9769e2b5a1f0f846cb9d37	Cobalt Strike stager	063191c49d49e6a8bdcd9d0ee2371fb1b90f17
b4f37f13a7e9c56ea95fa3792e11404eb3bdb878734f1ca394ceed344d22858f	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
b956c5e8ec6798582a68f24894c1e78b9b767aae4d5fb76b2cc71fc9c8befed8	Cobalt Strike	6fc283acfb7dda7bab02f5d23dc90b318f4c73a8
b99ba2449a93ab298d2ec5cacd5099871bacf6a8376e0b080c7240c8055b1395	Cobalt Strike	96fab57ef06b433f14743da96a5b874e96d8c97
b9e313e08b49d8d2ffe44cb6ec2192ee3a1c97b57c56f024c17d44db042fb9eb	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
bc238b3b798552958009f3a4ce08e5ce96edff06795281f8b8de6f5df9e4f0fe	Cobalt Strike stager	191566d8cc119cd6631d353eab0b8c1b8ba267
bcd64a8468762067d8a890b0aa7916289e68c9d8d8f419b94b78a19f5a74f378	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
c113e8a1c433b4c67ce9bce5dea4b470da95e914de4dc3c3d5a4f98bce2b7d6c	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
c1261f57a0481eb5d37176702903025c5b01a166ea6a6d42e1c1bdc0e5a0b04b	Cobalt Strike obfuscated shellcode	189b7afdd280d75130e633ebe2fcf8f54f28116a
c149792a5e5ce4c15f8506041e2f234a9a9254dbda214ec79ceef7d0911a3095	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
c2046d64bcfbab5afcb87a75bf3110e0fa89b3e0f7029ff81a335911cf52f00a	Cobalt Strike	d048001f09ad9eedde44f471702a2a0f453c573
c3509ba690a1fcb549b95ad4625f094963effc037df37bd96f9d8ed5c7136d94	Cobalt Strike	e0c0491e45dda838f4ac01b731dd39cc706467
c3cfbede0b561155062c2f44a9d44c79cdb78c05461ca50948892ff9a0678f3f	Cobalt Strike	bcb32a0f782442467ea8c0bf919a28b58690c68
c79ab271d2abd3ee8c21a8f6ad90226e398df1108b4d42dc551af435a124043c	Cobalt Strike	749d061acb0e584df337aaef26f3b555d5596a9
cab95dc6d08089dcd24c259f35b52bca682635713c058a74533501afb94ab91f	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
cea5c060dd8abd109b478e0de481f0df5ba3f09840746a6a505374d526bd28dc	MythicPacker	759ac6e54801e7171de39e637b9bb52519805
cfa604765b9d7a93765d46af78383978251486d9399e21b8e3da4590649c53e4	Cobalt Strike stager	57acdb7a22f5f0c6d374be2341dbef97efbcc61f
d1afca36f67b24eae7f2884c27c812cddc7e02f00f64bb2f62b40b21ef431084	Cobalt Strike	f570bd331a3d75e065d1825d97b922503c83a5
d1b6671fc0875678ecf39d737866d24aca03747a48f0c7e8855a5b09fc08712d	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
d3d48aa32b062b6e767966a8bab354eded60e0a11be5bc5b7ad8329aa5718c76	Cobalt Strike	60905c92501ec55883afc3f6402a05bddfd3353
d3eab2a134e7bd3f2e8767a6285b38d19cd3df421e8af336a7852b74f194802c	BlisterMythic	2fd38f6329b9b2c5e0379a445e81ece43fe0372
d439f941b293e3ded35bf52fac7f20f6a2b7f2e4b189ad2ac7f50b8358110491	Cobalt Strike	18a9eafb936bf1d527bd4f0bfae623400d63671
dac00ec780aabaffed1e89b3988905a7f6c5c330218b878679546a67d7e0eef2	Cobalt Strike	adc73af758c136e5799e25b4d3d69e462e090c
db62152fe9185cbd095508a15d9008b349634901d37258bc3939fe3a563b4b3c	MythicPacker	7f71d316c197e4e0aa1fce9d40c6068ada4249(
db81e91fc05991f71bfd5654cd60b9093c81d247ccd8b3478ab0ebef61efd2ad	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94

SHA256	Payload family	Payload SHA256
dd42c1521dbee54173be66a5f98a811e5b6ee54ad1878183c915b03b68b7c9bb	Cobalt Strike	d988a867a53c327099a4c9732a1e4ced6fe6ec
e0888b80220f200e522e42ec2f15629caa5a11111b8d1babff509d0da2b948f4	Cobalt Strike	915503b4e985ab31bc1d284f60003240430b3b
e30503082d3257737bba788396d7798e27977edf68b9dba7712a605577649ffb	Cobalt Strike	df01b0a8112ca80daf6922405c3f4d1ff7a8ff052
e521cad48d47d4c67705841b9c8fa265b3b0dba7de1ba674db3a63708ab63201	Cobalt Strike stager	40cac28490cddfa613fd58d1ecc8e676d9263a4
e62f5fc4528e323cb17de1fa161ad55eb451996dec3b31914b00e102a9761a52	Cobalt Strike	19e7bb5fa5262987d9903f388c4875ff2a37658
ebafb35fd9c7720718446a61a0a1a10d09bf148d26cdcd229c1d3d672835335c	Cobalt Strike	5cb2683953b20f34ff26ddc0d3442d07b4cd863
ebf40e12590fcc955b4df4ec3129cd379a6834013dae9bb18e0ec6f23f935bba	Cobalt Strike	d99bac48e6e347fcfd56bbf723a73b0b6fb5272f
ef7ff2d2decd8e16977d819f122635fcd8066fc8f49b27a809b58039583768d2	Cobalt Strike	adc73af758c136e5799e25b4d3d69e462e090c
efbffc6d81425ffb0d81e6771215c0a0e77d55d7f271ec685b38a1de7cc606a8	Cobalt Strike	47bd5fd96c350f5e48f5074ebee98e8b0f4efb8a
f08fdb0633d018c0245d071fa79cdc3915da75d3c6fc887a5ca6635c425f163a	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
f3bfd8ab9e79645babf0cb0138d51368fd452db584989c4709f613c93caf2bdc	Cobalt Strike	cd7135c94929f55e19e5d66359eab46422c3c5
f58de1733e819ea38bce21b60bb7c867e06edb8d4fd987ab09ecdbf7f6a319b9	MythicPacker	19eae7c0b7a1096a71b595befa655803c73500
f7fa532ad074db4a39fd0a545278ea85319d08d8a69c820b081457c317c0459e	Cobalt Strike	902d29871d3716113ca2af5caa6745cb4ab9d0
fce9de0a0acf2ba65e9e252a383d37b2984488b6a97d889ec43ab742160acce1	Cobalt Strike stager	40cac28490cddfa613fd58d1ecc8e676d9263a4
ffb255e7a2aa48b96dd3430a5177d6f7f24121cc0097301f2e91f7e02c37e6bf	Cobalt Strike	5af6626a6bc7265c21adaffb23cc58bc52c4ebfe
1a50c358fa4b725c6e0e26eee3646de26ba38e951f3fe414f4bf73532af62455	Cobalt Strike	8f1cc6ab8e95b9bfdf22a2bde77392e706b6fb7
1be3397c2a85b4b9a5a111b9a4e53d382df47a0a09065639d9e66e0b55fe36fc	Cobalt Strike stager	3f28a055d56f46559a21a2b0db918194324a13
1d058302d1e747714cac899d0150dcc35bea54cc6e995915284c3a64a76aacb1	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
02b1bd89e9190ff5edfa998944fd6048d32a3bde3a72d413e8af538d9ad770b4	Cobalt Strike obfuscated shellcode	3760db55a6943f4216f14310ab10d404e5c0a5
2cf125d6f21c657f8c3732be435af56ccbe24d3f6a773b15eccd3632ea509b1a	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
2f2e62c9481ba738a5da7baadfc6d029ef57bf7a627c2ac0b3e615cab5b0cfa2	Cobalt Strike	39ed516d8f9d9253e590bad7c5daecce9df21f1
3bc8ce92409876526ad6f48df44de3bd1e24a756177a07d72368e2d8b223bb39	Cobalt Strike	20e43f60a29bab142f050fab8c5671a0709ee4e
3dffb7f05788d981efb12013d7fadf74fdf8f39fa74f04f72be482847c470a53	Cobalt Strike	8e78ad0ef549f38147c6444910395b053c533a
3f6e3e7747e0b1815eb2a46d79ebd8e3cb9ccdc7032d52274bc0e60642e9b31e	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
3fff407bc45b879a1770643e09bb99f67cdcfe0e4f7f158a4e6df02299bac27e	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
4b3cd3aa5b961791a443b89e281de1b05bc3a9346036ec0da99b856ae7dc53a8	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
4faf362b3fe403975938e27195959871523689d0bf7fba757ddfa7d00d437fd4	Cobalt Strike	60905c92501ec55883afc3f6402a05bddfd3353
5d72cc2e47d3fd781b3fc4e817b2d28911cd6f399d4780a5ff9c06c23069eae1	MythicPacker	9a08d2db7d0bd7d4251533551d4def0f5ee52e
5ea74bca527f7f6ea8394d9d78e085bed065516eca0151a54474fffe91664198	Cobalt Strike	be314279f817f9f000a191efb8bcc2962fcc614b
5fc79a4499bafa3a881778ef51ce29ef015ee58a587e3614702e69da304395db	BlisterMythic	3d2499e5c9b46f1f144cfbbd4a2c8ca50a3c109
06cd6391b5fcf529168dc851f27bf3626f20e038a9c0193a60b406ad1ece6958	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
6a7ae217394047c17d56ec77b2243d9b55617a1ff591d2c2dfc01f2da335cbbf	MythicPacker	1e3b373f2438f1cc37e15fdede581bdf2f7fc220
6e75a9266e6bbfd194693daf468dd86d106817706c57b1aad95d7720ac1e19e3	Cobalt Strike	4adf3875a3d8dd3ac4f8be9c83aaa7e3e35a8d
7e61498ec5f0780e0e37289c628001e76be88f647cad7a399759b6135be8210a	Test application	43308bde79e71b2ed14f318374a80fadf201cc3

SHA256	Payload family	Payload SHA256
7f7b9f40eea29cfefc7f02aa825a93c3c6f973442da68caf21a3caae92464127	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
8b6eb2853ae9e5faff4afb08377525c9348571e01a0e50261c7557d662b158e1	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
8d53dc0857fa634414f84ad06d18092dedeb110689a08426f08cb1894c2212d4	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
8e6c0d338f201630b5c5ba4f1757e931bc065c49559c514658b4c2090a23e57b	Cobalt Strike	f2329ae2eb28bba301f132e5923282b74aa7a9
8f9289915b3c6f8bf9a71d0a2d5aeb79ff024c108c2a8152e3e375076f3599d5	BlisterMythic	f89cfbc1d984d01c57dd1c3e8c92c7debc2beb5
9c5c9d35b7c2c448a610a739ff7b85139ea1ef39ecd9f51412892cd06fde4b1b	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
13c7f28044fdb1db2289036129b58326f294e76e011607ca8d4c5adc2ddddb16	Cobalt Strike	19e7bb5fa5262987d9903f388c4875ff2a37658
19b0db9a9a08ee113d667d924992a29cd31c05f89582953eff5a52ad8f533f4b	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
19d4a7d08176119721b9a302c6942718118acb38dc1b52a132d9cead63b11210	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
22e65a613e4520a6f824a69b795c9f36af02247f644e50014320857e32383209	Cobalt Strike	18a9eafb936bf1d527bd4f0bfae623400d63671
028da30664cb9f1baba47fdaf2d12d991dcf80514f5549fa51c38e62016c1710	Cobalt Strike	8e78ad0ef549f38147c6444910395b053c533a
37b6fce45f6bb52041832eaf9c6d02cbc33a3ef2ca504adb88e19107d2a7aeaa	Cobalt Strike	902d29871d3716113ca2af5caa6745cb4ab9d0
42beac1265e0efc220ed63526f5b475c70621573920968a457e87625d66973af	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
43c1ee0925ecd533e0b108c82b08a3819b371182e93910a0322617a8acf26646	Cobalt Strike	5cb2683953b20f34ff26ddc0d3442d07b4cd863
44ce7403ca0c1299d67258161b1b700d3fa13dd68fbb6db7565104bba21e97ae	MythicPacker	f3b0357562e51311648684d381a23fa2c1d090
49ba10b4264a68605d0b9ea7891b7078aeef4fa0a7b7831f2df6b600aae77776	Cobalt Strike	0603cf8f5343723892f08e990ae2de8649fcb4f2
54c7c153423250c8650efc0d610a12df683b2504e1a7a339dfd189eda25c98d4	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
58fdee05cb962a13c5105476e8000c873061874aadbc5998887f0633c880296a	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
73baa040cd6879d1d83c5afab29f61c3734136bffe03c72f520e025385f4e9a2	Cobalt Strike	17392d830935cfad96009107e8b034f952fb528
78d93b13efd0caa66f5d91455028928c3b1f44d0f2222d9701685080e30e317d	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
83c121db96d99f0d99b9e7a2384386f3f6debcb01d977c4ddca5bcdf2c6a2daa	Cobalt Strike stager	39323f9c0031250414cb4683662e1c533960de
84b245fce9e936f1d0e15d9fca8a1e4df47c983111de66fcc0ad012a63478c8d	Cobalt Strike stager	d961e9db4a96c87226dbc973658a14082324e
84b2d16124b690d77c5c43c3a0d4ad78aaf10d38f88d9851de45d6073d8fcb65	Cobalt Strike	0091186459998ad5b699fdd54d57b1741af738
85d3f81a362a3df9ba2f0a00dd12cd654e55692feffc58782be44f4c531d9bb9	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
96e8b44ec061c49661bd192f279f7b7ba394d03495a2b46d3b37dcae0f4892f1	Cobalt Strike stager	6f7d7da247cac20d5978f1257fdd420679d0ce1
96ebacf48656b804aed9979c2c4b651bbb1bc19878b56bdf76954d6eff8ad7ca	Cobalt Strike	d988a867a53c327099a4c9732a1e4ced6fe6ec
113c9e7760da82261d77426d9c41bc108866c45947111dbae5cd3093d69e0f1d	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
149c3d044abc3c3a15ba1bb55db7e05cbf87008bd3d23d7dd4a3e31fcfd7af10	Cobalt Strike	e63807daa9be0228d90135ee707ddf03b00353
307fc7ebde82f660950101ea7b57782209545af593d2c1115c89f328de917dbb	Cobalt Strike stager	40cac28490cddfa613fd58d1ecc8e676d9263a4
356efe6b10911d7daaffed64278ba713ab51f7130d1c15f3ca86d17d65849fa5	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
394ce0385276acc6f6c173a3dde6694881130278bfb646be94234cc7798fd9a9	Cobalt Strike	60e2fe4eb433d3f6d590e75b2a767755146aca
396dce335b16111089a07ecb2d69827f258420685c2d9f3ea9e1deee4bff9561	Test application	43308bde79e71b2ed14f318374a80fadf201cc3

SHA256	Payload family	Payload SHA256
541eab9e348c40d510db914387068c6bfdf46a6ff84364fe63f6e114af8d79cf	Cobalt Strike stager	4e2a011922e0060f995bfde375d75060bed001
745a3dcdda16b93fedac8d7eefd1df32a7255665b8e3ee71e1869dd5cd14d61c	Cobalt Strike obfuscated shellcode	cef1a88dfc436dab9ae104f0770a434891bbd60
753f77134578d4b941b8d832e93314a7159455193127057014080567 5c6e9ad3	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
863de84a39c9f741d8103db83b076695d0d10a7384e4e3ba319c05a6018d9737	Cobalt Strike	3a1e65d7e9c3c23c41cb1b7d1117be4355bebf
902fa7049e255d5c40081f2aa168ac7b36b56041612150c3a5d2b6df707a3cff	Cobalt Strike	397c08f5cdc59085a48541c89d23a8880d4155
927e04371fa8b8d8a1de58533053c305bb73a8df8765132a932efd579011c375	Cobalt Strike	2e0767958435dd4d218ba0bc99041cc9f12c94
2043d7f2e000502f69977b334e81f307e2fda742bbc5b38745f6c1841757fddc	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
02239cac2ff37e7f822fd4ee57ac909c9f541a93c27709e9728fef2000453afe	Cobalt Strike	18a9eafb936bf1d527bd4f0bfae623400d63671
4257bf17d15358c2f22e664b6112437b0c2304332ff0808095f1f47cf29fc1a2	Cobalt Strike	3a1e65d7e9c3c23c41cb1b7d1117be4355bebf
6558ac814046ecf3da8c69affea28ce93524f93488518d847e4f03b9327acb44	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
8450ed10b4bef6f906ff45c66d1a4a74358d3ae857d3647e139fdaf0e3648c10	BlisterMythic	ab7cab5192f0bef148670338136b0d3affe8ae0
9120f929938cd629471c7714c75d75d30daae1f2e9135239ea5619d77574c1fe	Cobalt Strike	647e992e24e18c14099b68083e9b04575164e
28561f309d208e885a325c974a90b86741484ba5e466d59f01f660bed1693689	Cobalt Strike	397c08f5cdc59085a48541c89d23a8880d4155
30628bcb1db7252bf710c1d37f9718ac37a8e2081a2980bead4f21336d2444bc	Cobalt Strike obfuscated shellcode	13f23b5db4a3d0331c438ca7d516d565a08cac
53121c9c5164d8680ae1b88d95018a553dff871d7b4d6e06bd69cbac047fe00f	Cobalt Strike	902d29871d3716113ca2af5caa6745cb4ab9d0
67136ab70c5e604c6817105b62b2ee8f8c5199a647242c0ddbf261064bb3ced3	Cobalt Strike obfuscated shellcode	0aecd621b386126459b39518f157ee240866c6
79982f39ea0c13eeb93734b12f395090db2b65851968652cab5f6b0827b49005	MythicPacker	152455f9d970f900eb237e1fc2c29ac4c726164
87269a95b1c0e724a1bfe87ddcb181eac402591581ee2d9b0f56dedbaac04ff8	Cobalt Strike	f3d42e4c1a47f0e1d3812d5f912487d04662152
89196b39a0edebdf2026053cb4e87d703b9942487196ff9054ef775fdcad1899	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
91446c6d3c11074e6ff0ff42df825f9ffd5f852c2e6532d4b9d8de340fa32fb8	Test application	43308bde79e71b2ed14f318374a80fadf201cc3
96823bb6befe5899739bd69ab00a6b4ae1256fd586159968301a4a69d675a5ec	Cobalt Strike	3b3bdd819f4ee8daa61f07fc9197b2b39d04342
315217b860ab46c6205b36e49dfaa927545b90037373279723c3dec165dfaf11	Cobalt Strike	96fab57ef06b433f14743da96a5b874e96d8c97
427481ab85a0c4e03d1431a417ceab66919c3e704d7e017b355d8d64be2ccf41	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
595153eb56030c0e466cda0becb1dc9560e38601c1e0803c46e7dfc53d1d2892	Cobalt Strike	f245b2bc118c3c20ed96c8a9fd0a7b659364f9e
812263ea9c6c44ef6b4d3950c5a316f765b62404391ddb6482bdc9a23d6cc4a6	Cobalt Strike	18a9eafb936bf1d527bd4f0bfae623400d63671
1358156c01b035f474ed12408a9e6a77fe01af8df70c08995393cbb7d1e1f8a6	Cobalt Strike	b916749963bb08b15de7c302521fd0ffec1c666
73162738fb3b9cdd3414609d3fe930184cdd3223d9c0d7cb56e4635eb4b2ab67	Cobalt Strike	19e7bb5fa5262987d9903f388c4875ff2a37658
343728792ed1e40173f1e9c5f3af894feacd470a9cdc72e4f62c0dc9cbf63fc1	Putty	0581160998be30f79bd9a0925a01b0ebc4cb94
384408659efa1f87801aa494d912047c26259cd29b08de990058e6b45619d91a	Cobalt Strike stager	824914bb34ca55a10f902d4ad2ec931980f560
49925637250438b05d3aebaac70bb180a0825ec4272fbe74c6fecb5e085bcf10	Cobalt Strike	e0c0491e45dda838f4ac01b731dd39cc706467

Table 6, Hashes of Blister samples and of the payload it drops, including the payload label.