
dissect.cobaltstrike

Release 1.0.1.dev11

Fox-IT part of NCC Group

Jan 08, 2024

OVERVIEW

1	Installation	3
1.1	Installing from source	3
1.2	Running tests	4
1.3	Linting	4
1.4	Documentation	4
2	Examples	5
2.1	Beacon Configuration	5
2.2	Memory dumps	8
2.3	PE Artifacts	8
2.4	C2 Profiles	9
2.5	BeaconConfig to C2 Profile	10
2.6	Stager URIs and checksum8	12
3	Tutorials	15
3.1	A Minimal Beacon Client	15
3.2	Decrypt Cobalt Strike PCAPs	22
4	Scripts	29
4.1	example_client.py	29
4.2	checksum8-accesslogs.py	31
4.3	dump_beacon_keys.py	33
5	beacon-artifact	35
5.1	beacon-artifact - CLI interface	35
6	beacon-client	37
6.1	Dumping saved records	38
7	beacon-dump	41
7.1	XOR keys	41
7.2	Output format	41
8	beacon-pcap	43
8.1	beacon-pcap - CLI interface	44
9	beacon-xordecode	45
9.1	Nonce offset	45
9.2	MZ header	45
10	c2profile-dump	47

10.1	c2profile-dump - CLI interface	47
11	dissect.cobaltstrike	49
11.1	Submodules	49
12	Structure definitions	101
12.1	dissect.cobaltstrike.beacon.CS_DEF	101
12.2	dissect.cobaltstrike.pe.PE_DEF	105
12.3	dissect.cobaltstrike.c_c2.C2_DEF	108
13	C2Profile grammar	113
14	Indices and tables	119
	Python Module Index	121
	Index	123

Welcome! This is the official documentation for `dissect.cobaltstrike`.

dissect.cobaltstrike is a Python library for dissecting and parsing Cobalt Strike related data such as beacon payloads and Malleable C2 Profiles.

Source code can be found here:

- <https://github.com/fox-it/dissect.cobaltstrike>

Note: *dissect.cobaltstrike* is released under the MIT license.

INSTALLATION

The easiest way to install `dissect.cobaltstrike` is to use **pip**:

```
$ pip install dissect.cobaltstrike
```

Python 3.7 or higher is required and it has the following dependencies:

- `dissect.cstruct` - for structure parsing
- `lark` - for parsing malleable c2 profiles

The following pip *extras* flavours are provided as well:

```
$ pip install dissect.cobaltstrike[c2]
$ pip install dissect.cobaltstrike[pcap]
$ pip install dissect.cobaltstrike[full]
```

- `[c2]` for if you want to communicate with Cobalt Strike Team Servers, eg: *beacon-client*.
- `[pcap]` for if you want to parse and decrypt PCAPS containing Beacon traffic, eg: *beacon-pcap*.
- `[full]` provides the above but also installs `rich` for prettier console logging.

1.1 Installing from source

If you want to install `dissect.cobaltstrike` from source, you can use the following steps:

```
$ git clone https://github.com/fox-it/dissect.cobaltstrike.git
$ cd dissect.cobaltstrike
$ pip install --editable .[full]
```

Using a virtual environment is recommended. Using the `--editable` flag ensures that any changes you make to the source code directly affects the installed package.

1.2 Running tests

The test suite uses `pytest` and using `tox` is the recommended way to run the test suite:

```
$ pip install tox
$ tox
```

This will run tests on both Python 3 and PyPy3. To limit to Python 3 only, run:

```
$ tox -e py3
```

You can also specify custom arguments to `pytest` by appending the arguments after `--` (two dashes), e.g. to only run tests with `checksum8` in the name including verbose and stdout logging:

```
$ tox -e py3 -- -vs -k checksum8
```

Note: The test suite contains zipped beacon payloads that are used as test fixtures and can be unzipped during some tests. Running the test suite on Windows could trigger Windows Defender or your Antivirus.

1.3 Linting

For linting (black and flake8):

```
$ tox -e lint
```

1.4 Documentation

To generate the documentation locally (sphinx):

```
$ tox -e docs
```


EXAMPLES

Some examples showing how to use the `dissect.cobaltstrike` Python API.

2.1 Beacon Configuration

The main class for dealing with Cobalt Strike Beacon configuration is `BeaconConfig`. It's recommended to instantiate the class by using one of the following constructors:

- `BeaconConfig.from_file()`
- `BeaconConfig.from_path()`
- `BeaconConfig.from_bytes()`

These `from_` constructors will handle `XorEncoded` beacons by default and tries the default `XOR` keys used for obfuscating the beacon configuration. It raises `ValueError` if no beacon config was found.

For example to load the configuration of a Beacon payload on disk and access it's settings:

```
In [1]: from dissect.cobaltstrike.beacon import BeaconConfig

In [2]: bconfig = BeaconConfig.from_path("beacon_92.bin")

In [3]: bconfig
Out[3]: <BeaconConfig ['londonteea.com']>

In [4]: bconfig.version
Out[4]: <BeaconVersion 'Cobalt Strike 4.2 (Nov 06, 2020)', tuple=(4, 2), date=2020-11-06>

In [5]: hex(bconfig.watermark)
Out[5]: '0x5109bf6d'

In [6]: bconfig.settings["SETTING_C2_REQUEST"]
Out[6]:
[('_HEADER', b'Connection: close'),
 ('_HEADER', b'Accept-Language: en-US'),
 ('BUILD', 'metadata'),
 ('MASK', True),
 ('BASE64', True),
 ('PREPEND', b'wordpress_ed1f617bbd6c004cc09e046f3c1b7148='),
 ('HEADER', b'Cookie')]
```

If the beacon uses a non standard `XOR` key it will not find the beacon configuration and will raise `ValueError`:

```
In [7]: %xmode Minimal
Exception reporting mode: Minimal
```

```
In [8]: bconfig = BeaconConfig.from_path("beacon_xor.bin")
ValueError: No valid Beacon configuration found
```

Specify `all_xor_keys=True` to automatically try all single-byte XOR keys when the default keys fail:

```
In [9]: bconfig = BeaconConfig.from_path("beacon_xor.bin", all_xor_keys=True)

In [10]: bconfig
Out[10]: <BeaconConfig ['group.ccb.com.dsa.dnsv1.com']>

In [11]: bconfig.xorkey.hex()
Out[11]: 'cc'

In [12]: bconfig.version
Out[12]: <BeaconVersion 'Cobalt Strike 4.4 (Aug 04, 2021)', tuple=(4, 4), date=2021-08-
→04>
```

Or if you want to speed things up and you know a set of candidate XOR keys, just specify them using `xor_keys` to override the `DEFAULT_XOR_KEYS`:

```
In [13]: BeaconConfig.from_path("beacon_xor.bin", xor_keys=[b"\xcc"])
Out[13]: <BeaconConfig ['group.ccb.com.dsa.dnsv1.com']>

In [14]: _.xorkey.hex()
Out[14]: 'cc'
```

If you have extracted a Beacon configuration block manually, for example via *x64dbg*, you can pass it directly to `BeaconConfig()`. However, this only works with configuration bytes that is not obfuscated.

If the configuration block is obfuscated with a single-byte XOR key, use the `BeaconConfig.from_bytes()` constructor:

```
In [15]: data = '00000000000000002e2f2e2f2e2c2e262e2c2e2f2e2c2f952e2d2e2c2e2a2e2ec44e2e2a2e2c2e2a2e3b7b76'

In [16]: BeaconConfig.from_bytes(bytes.fromhex(data))
Out[16]: <BeaconConfig ['sinitude.com']>

In [17]: config = _

In [18]: config.protocol
Out[18]: 'https'

In [19]: config.domain_uri_pairs
Out[19]: [('sinitude.com', '/web/chatr.portal')]

In [20]: config.settings
Out[20]:
mappingproxy({'SETTING_PROTOCOL': 8,
              'SETTING_PORT': 443,
              'SETTING_SLEEPTIME': 60000,
              'SETTING_MAXGET': 1398104,
```

(continues on next page)

(continued from previous page)

```

        'SETTING_JITTER': 30,
        'SETTING_PUBKEY':
↪ 'a83298f790d9a47e425ce5d67a148ba87498e7ed5eb4b4f4f1e0c5e177b274a2',
        'SETTING_DOMAINS': 'sinitude.com,/web/chatr.portal',
        'SETTING_SPAWNT0': '00000000000000000000000000000000',
        'SETTING_SPAWNT0_X86': '%windir%\syswow64\dlhost.exe',
        'SETTING_SPAWNT0_X64': '%windir%\sysnative\dlhost.exe',
        'SETTING_CRYPT0_SCHEME': 0,
        'SETTING_C2_VERB_GET': 'GET',
        'SETTING_C2_VERB_POST': 'POST',
        'SETTING_C2_CHUNK_POST': 0,
        'SETTING_WATERMARK': 1359593325,
        'SETTING_CLEANUP': 0,
        'SETTING_CFG_CAUTION': 0,
        'SETTING_USERAGENT': 'Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.36_
↪ (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36',
        'SETTING_SUBMITURI': '/web/logon.aspx',
        'SETTING_C2_RECOVER': [('print', True), ('base64', True)],
        'SETTING_C2_REQUEST': [('_HEADER', b'Content-Type: text/html'),
        ('_HEADER', b'Cache-Control: no-cache'),
        ('BUILD', 'metadata'),
        ('NETBIOS', True),
        ('URI_APPEND', True)],
        'SETTING_C2_POSTREQ': [('_HEADER',
        b'Content-Type: multipart/form-data'),
        ('_HEADER', b'Cache-Control: no-cache'),
        ('BUILD', 'id'),
        ('NETBIOSU', True),
        ('URI_APPEND', True),
        ('BUILD', 'output'),
        ('BASE64URL', True),
        ('PRINT', True)],
        'SETTING_HOST_HEADER': '',
        'SETTING_HTTP_NO_COOKIES': 0,
        'SETTING_PROXY_BEHAVIOR': 2,
        'SETTING_TCP_FRAME_HEADER': b'\x90',
        'SETTING_SMB_FRAME_HEADER': b'p',
        'SETTING_EXIT_FUNK': 0,
        'SETTING_KILLDATE': 0,
        'SETTING_GARGLE_NOOK': 0,
        'SETTING_PROCIJN_PERMS_I': 4,
        'SETTING_PROCIJN_PERMS': 32,
        'SETTING_PROCIJN_MINALLOC': 17500,
        'SETTING_PROCIJN_TRANSFORM_X86': [('append', b'\x90\x90'),
        ('prepend', b'')],
        'SETTING_PROCIJN_TRANSFORM_X64': [('append', b'\x90\x90'),
        ('prepend', b'')],
        'SETTING_PROCIJN_STUB': '0ce2f55444e4793516b5afe967be9255',
        'SETTING_PROCIJN_EXECUTE': ['CreateThread "ntdll!RtlUserThreadStart+0x26"',
        'CreateThread',
        'NtQueueApcThread_s',
        'CreateRemoteThread "kernel32.dll!LoadLibraryA+0x51"',

```

(continues on next page)

(continued from previous page)

```
'RtlCreateUserThread'],
'SETTING_PROCINJ_ALLOCATOR': 1})
```

```
In [21]: config.version
```

```
Out[21]: <BeaconVersion 'Cobalt Strike 4.1 (Jun 25, 2020)', tuple=(4, 1), date=2020-06-
↪25>
```

2.2 Memory dumps

While you can use *BeaconConfig* to load Beacon payloads directly, it can also load a memory dump (or any other file) and check for beacon configurations. However, the default constructors will only return the first found beacon configuration.

If you have a memory dump that could contain multiple beacons, use *iter_beacon_config_blocks()* to iterate over all found beacon configuration blocks and instantiate *BeaconConfig* manually:

```
import sys
from dissect.cobaltstrike import beacon

with open(sys.argv[1], "rb") as f:
    for config_block, extra_data in beacon.iter_beacon_config_blocks(f):
        try:
            bconfig = beacon.BeaconConfig(config_block)
            if not len(bconfig.domains):
                continue
        except ValueError:
            continue
        print(bconfig, bconfig.domain_uri_pairs)
```

This will try to find all beacon *config_block* bytes in the file and try to instantiate a *BeaconConfig* from it. For verification it will check if the beacon has a domain to ensure that *config_block* was not just some random data.

2.3 PE Artifacts

Use the *dissect.cobaltstrike.pe* module to extract PE artifacts. If the payload is *XorEncoded* you need to load it using *XorEncodedFile* first.

```
In [22]: from dissect.cobaltstrike import xordecode

In [23]: from dissect.cobaltstrike import pe

In [24]: import time

In [25]: xf = xordecode.XorEncodedFile.from_path("beacon_93.bin")

In [26]: pe.find_architecture(xf)
Out[26]: 'x64'

In [27]: pe.find_compile_stamps(xf)
```

(continues on next page)

(continued from previous page)

```

Out[27]: (1628256615, 1614696183)

In [28]: compile_stamp, export_stamp = _

In [29]: time.ctime(compile_stamp)
Out[29]: 'Fri Aug  6 13:30:15 2021'

In [30]: pe.find_magic_mz(xf)
Out[30]: b'MZAR'

In [31]: pe.find_magic_pe(xf)
Out[31]: b'PE'

In [32]: pe.find_stage_prepend_append(xf)
Out[32]: (b'\x90\x90\x90\x90\x90\x90\x90\x90', None)

```

2.4 C2 Profiles

Loading Cobalt Strike Malleable C2 Profiles is also supported, to load a profile from disk:

```

In [33]: from dissect.cobaltstrike.c2profile import C2Profile

In [34]: profile = C2Profile.from_path("amazon.profile")

```

To access the C2Profile configuration settings use the *C2Profile.as_dict* method or the *C2Profile.properties* attribute. For example:

```

In [35]: profile.as_dict()
Out[35]:
{'sleeptime': ['5000'],
 'jitter': ['0'],
 'maxdns': ['255'],
 'useragent': ['Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko'],
 'http-get.uri': ['/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books'],
 'http-get.client.header': [('Accept', '*/*'), ('Host', 'www.amazon.com')],
 'http-get.client.metadata': ['base64',
 ('prepend', b'session-token='),
 ('prepend', b'skin=noskin;'),
 ('append', b'csm-hit=s-24KU11BB82RZSYGJ3BDK|1419899012996'),
 ('header', b'Cookie')],
 'http-get.server.header': [('Server', 'Server'),
 ('x-amz-id-1', 'THKUYEZKCKPGY5T42PZT'),
 ('x-amz-id-2',
 'a21yZ2xrNDNtdGRsa212bGV3YW85amZuZW9ydG5rZmRuZ2tmZGl4aHRvNDVpbgo='),
 ('X-Frame-Options', 'SAMEORIGIN'),
 ('Content-Encoding', 'gzip')],
 'http-get.server.output': ['print'],
 'http-post.uri': ['/N4215/adj/amzn.us.sr.aps'],
 'http-post.client.header': [('Accept', '*/*'),
 ('Content-Type', 'text/xml'),

```

(continues on next page)

(continued from previous page)

```
(('X-Requested-With', 'XMLHttpRequest'),
 ('Host', 'www.amazon.com')),
'http-post.client.parameter': [('sz', '160x600'),
 ('oe', 'oe=ISO-8859-1;'),
 ('s', '3717'),
 ('dc_ref', 'http%3A%2F%2Fwww.amazon.com')],
'http-post.client.id': [('parameter', b'sn')],
'http-post.client.output': ['base64', 'print'],
'http-post.server.header': [('Server', 'Server'),
 ('x-amz-id-1', 'THK9YEZJCKPGY5T42OZT'),
 ('x-amz-id-2',
 'a21JZ1xrNDNtdGRsa219bGV3YW85amZuZW9zdG5rZmRuZ2tmZGl4aHRvNDVpbgo='),
 ('X-Frame-Options', 'SAMEORIGIN'),
 ('x-ua-compatible', 'IE=edge')],
'http-post.server.output': ['print']}
```

In [36]: profile.properties["useragent"]

Out[36]: ['Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko']

In [37]: profile.properties["http-get.uri"]

Out[37]: ['/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books']

In [38]: profile.properties["http-post.client.parameter"]

Out[38]:

```
[('sz', '160x600'),
 ('oe', 'oe=ISO-8859-1;'),
 ('s', '3717'),
 ('dc_ref', 'http%3A%2F%2Fwww.amazon.com')]
```

Note: Currently all the values in the dictionary are lists, this might change in the future.

2.5 BeaconConfig to C2 Profile

Use *C2Profile.from_beacon_config* to load settings from a *BeaconConfig*. This allows for dumping the Beacon Configuration to a more readable (and reusable) C2 Profile.

In [39]: config

Out[39]: <BeaconConfig ['sinitude.com']>

In [40]: profile = C2Profile.from_beacon_config(config)

In [41]: print(profile)

```
set sleeptime "60000";
set jitter "30";
set spawn_to_x86 "%windir%\syswow64\dlhost.exe";
set spawn_to_x64 "%windir%\sysnative\dlhost.exe";
set useragent "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.36 (KHTML, like Gecko)
↳ Chrome/90.0.4430.85 Safari/537.36";
set tcp_frame_header "\x90";
```

(continues on next page)

(continued from previous page)

```
set smb_frame_header "p";

http-get {
  set uri "/web/chatr.portal";
  set verb "GET";

  server {

    output {
      base64;
      print;
    }
  }

  client {
    header "Content-Type" "text/html";
    header "Cache-Control" "no-cache";

    metadata {
      netbios;
      uri-append;
    }
  }
}

http-post {
  set verb "POST";
  set uri "/web/logon.aspx";

  client {
    header "Content-Type" "multipart/form-data";
    header "Cache-Control" "no-cache";

    id {
      netbiosu;
      uri-append;
    }

    output {
      base64url;
      print;
    }
  }
}

stage {
  set cleanup "0";
}

process-inject {
  set starttrwx "false";
  set userwx "false";
}
```

(continues on next page)

(continued from previous page)

```

set min_alloc "17500";

transform-x86 {
    append "\x90\x90";
}

transform-x64 {
    append "\x90\x90";
}

execute {
    CreateThread "ntdll!RtlUserThreadStart+0x26";
    CreateThread;
    CreateRemoteThread "kernel32.dll!LoadLibraryA+0x51";
    RtlCreateUserThread;
}

set allocator "NtMapViewOfSection";
}

```

2.6 Stager URIs and checksum8

checksum8 URIs are used for payload staging and used in Cobalt Strike shellcode stagers for retrieving the final Beacon payload from the Team Server.

Note: Metasploit also uses *checksum8*, it exists in Cobalt Strike to be compatible with Metasploit.

The following *checksum8* values are used by Cobalt Strike:

checksum8	architecture
92	beacon x86
93	beacon x64

To calculate the *checksum8* of an URI:

```

In [42]: from dissect.cobaltstrike import utils

In [43]: utils.checksum8("/rLEZ")
Out[43]: 93

In [44]: utils.is_stager_x64("/rLEZ")
Out[44]: True

In [45]: utils.is_stager_x86("/yearbook")
Out[45]: True

```

To easily generate valid Cobalt Strike stager URIs, use *utils.random_stager_uri*:


```
In [46]: from dissect.cobaltstrike import utils

In [47]: utils.random_stager_uri(x64=True)
Out[47]: '/J6sj'

In [48]: utils.random_stager_uri(length=30)
Out[48]: '/Wt1tX8Yk2ZryVJUL10vpw2uIMaWxC3'
```

Or, a fun script to check a dictionary or word list for valid *stager x86* words:

```
import sys
from dissect.cobaltstrike import utils

for line in sys.stdin:
    line = line.strip().lower()
    if utils.is_stager_x86(line):
        print(line)
```

```
$ cat /usr/share/dict/words | python is_stager_x86.py | head -n 10
abortive
abshenry
accommodative
acosmism
acquirer
acroaesthesia
adance
adiposis
adoptive
adulator
```


TUTORIALS

These tutorials show how you can utilize `dissect.cobaltstrike` for some specific use cases.

3.1 A Minimal Beacon Client

This tutorial shows how to implement your own minimal beacon client that can handle tasks from the Cobalt Strike Team Server and send back custom responses (also known as callbacks).

While the CLI tool *beacon-client* is already a fully working client that can connect to a Team Server given a beacon payload, it does not have any task handlers. While this is very useful for testing and monitoring, it might be useful to have a client that can handle tasks and send custom callback responses back to the Team Server.

We can make our own custom beacon client by using the `dissect.cobaltstrike.client` module.

Note: Currently only the HTTP and HTTPS protocol is supported, so DNS beacons are not yet supported.

See also *scripts/example_client.py* for a more detailed implemented client.

3.1.1 Installation

First we install `dissect.cobaltstrike` with the `[c2]` extra, as we are going to communicate with C2 Servers:

```
$ pip install dissect.cobaltstrike[c2]
```

This installs the necessary dependencies such as *PyCryptodome* and *httplib*.

3.1.2 Basic client

There are two ways of implementing a Beacon client, first is to subclass `HttpBeaconClient` and second one is to instantiate a `HttpBeaconClient` and use *decorators* to register task handlers on this instance. We will use the decorator method in the following steps but also show how to implement it using a subclass at the end of this tutorial.

Here is a basic client that can do check-ins but has no task handlers:

Listing 1: myclient.py

```
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options  
  
client = HttpBeaconClient()
```

(continues on next page)

(continued from previous page)

```
args, options = parse_commandline_options()
client.run(**options)
```

Let's break down what the current script is doing:

Listing 2: myclient01.py

```
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options

client = HttpBeaconClient()

args, options = parse_commandline_options()
client.run(**options)
```

We first import *HttpBeaconClient* and *parse_commandline_options()*.

Listing 3: myclient02.py

```
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options

client = HttpBeaconClient()

args, options = parse_commandline_options()
client.run(**options)
```

We instantiate a *HttpBeaconClient* and store this in the (global) variable *client*.

Listing 4: myclient03.py

```
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options

client = HttpBeaconClient()

args, options = parse_commandline_options()
client.run(**options)
```

We now call *parse_commandline_options()* which uses a builtin *ArgumentParser* with common beacon client options and return this as *args* and a dictionary *options* which can be passed to our *run()* method.

Listing 5: myclient03.py

```
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options

client = HttpBeaconClient()

args, options = parse_commandline_options()
client.run(**options)
```

We then run the client with our options, the double ***options* expands the *options* as keyword arguments so you don't have to manually pass keyword options arguments to *run()* like this:

```
client.run(bconfig=options["bconfig"], beacon_id=options["beacon_id"], ...)
```

When `client.run()` is executed it will start the beacon loop to actively connect to the Cobalt Strike Team Server and retrieve tasks. However, there are no task handlers yet and basically this acts the same as the *beacon-client* CLI tool.

Let's implement a task handler in the next section!

Task handler

We are going to implement a Task handler when the `ls` command is issued from the Team Server to our Beacon client:

Listing 6: myclient_ls_01.py

```
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options
from dissect.cobaltstrike.client import BeaconCommand, TaskPacket

client = HttpBeaconClient()

@client.handle(BeaconCommand.COMMAND_FILE_LIST)
def handle_file_list(task: TaskPacket):
    print(f"Received: {task}!")

args, options = parse_commandline_options()
client.run(**options)
```

We import *BeaconCommand* and *TaskPacket* here to make things easier when using an IDE such as VSCode for autocompletion.

Listing 7: myclient_ls_02.py

```
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options
from dissect.cobaltstrike.client import BeaconCommand, TaskPacket

client = HttpBeaconClient()

@client.handle(BeaconCommand.COMMAND_FILE_LIST)
def handle_file_list(task: TaskPacket):
    print(f"Received: {task}!")

args, options = parse_commandline_options()
client.run(**options)
```

We define our task handler function called `handle_file_list` with a single argument `task`. The handler function must accept a single argument which is a *TaskPacket* object and is a simple wrapper around a `dissect.cstruct` instance with the following structure:

```
typedef struct TaskPacket {
    uint32 epoch;
    uint32 total_size;
    BeaconCommand command;
    uint32 size;
    char data[size];
};
```

In this handler we just print the received task.

Listing 8: myclient_ls_03.py

```

from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options
from dissect.cobaltstrike.client import BeaconCommand, TaskPacket

client = HttpBeaconClient()

@client.handle(BeaconCommand.COMMAND_FILE_LIST)
def handle_file_list(task: TaskPacket):
    print(f"Received task: {task}!")

args, options = parse_commandline_options()
client.run(**options)

```

Next we decorate this function with `@client.handle()` passing in the `COMMAND_FILE_LIST` command id. This registers the method as a handler for when a `COMMAND_FILE_LIST` command is Tasked by the Team Server. For a complete list of *COMMANDS* you can refer to [BeaconCommand](#).

When we now run the client and receive a `ls` Task we will see:

```

$ python myclient_ls_03.py beacon.bin -v
...
Received task: <TaskPacket epoch=0x635bba6a, total_size=0x24, command=<BeaconCommand.
COMMAND_FILE_LIST: 53>, size=0xb, data=b'\xff\xff\xff\xfe\x00\x00\x00\x03.\\"*\'>

```

Parsing Task data

The `task.data` attribute contains the raw Task data bytes, and must still be parsed if you want to do anything with it. Currently you need to parse this manually as there are many different Tasks and they all have a different structure.

Here is an example on how to parse the `task.data` for a `COMMAND_FILE_LIST` TaskPacket:

Listing 9: myclient_parse_ls_01.py

```

from io import BytesIO
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options
from dissect.cobaltstrike.client import BeaconCommand, TaskPacket
from dissect.cobaltstrike.utils import u32be

client = HttpBeaconClient()

@client.handle(BeaconCommand.COMMAND_FILE_LIST)
def handle_file_list(task: TaskPacket):
    # Parse task data for file listing, which is structured as:
    #
    # |<request_number>|<size_of_folder>|<folder>|
    with BytesIO(task.data) as data:
        req_no = u32be(data.read(4))
        size = u32be(data.read(4))
        folder = data.read(size).decode()

    print(f"Received ls for {folder}!")

```

(continues on next page)

(continued from previous page)

```
args, options = parse_commandline_options()
client.run(**options)
```

We first need some extra imports so we can easier parse data, BytesIO for reading bytes as a file-like object and u32be to read bytes as an uint32 value in Big Endian.

Listing 10: myclient_parse_ls_02.py

```
from io import BytesIO
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options
from dissect.cobaltstrike.client import BeaconCommand, TaskPacket
from dissect.cobaltstrike.utils import u32be

client = HttpBeaconClient()

@client.handle(BeaconCommand.COMMAND_FILE_LIST)
def handle_file_list(task: TaskPacket):
    # Parse task data for file listing, which is structured as:
    #
    # |<request_number>|<size_of_folder>|<folder>|
    with BytesIO(task.data) as data:
        req_no = u32be(data.read(4))
        size = u32be(data.read(4))
        folder = data.read(size).decode()

    print(f"Received ls for {folder}!")

args, options = parse_commandline_options()
client.run(**options)
```

We create a BytesIO instance from the `task.data` bytes so it acts more a like file-like object. And then we read the first uint32 value as the *request_number*, second uint32 is the size of the *folder* name buffer that is being requested. And finally we read the *folder* name using that size.

We finally print the parsed folder name that is being requested for `ls`.

Sending Callbacks

Instead of printing stuff locally, let's make it more interesting by sending back some data to the Team Server. Also known as a *Callback*.

Listing 11: myclient_ls_callback_01.py

```
from io import BytesIO
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options
from dissect.cobaltstrike.client import BeaconCommand, TaskPacket
from dissect.cobaltstrike.client import CallbackDebugMessage
from dissect.cobaltstrike.utils import u32be

client = HttpBeaconClient()

@client.handle(BeaconCommand.COMMAND_FILE_LIST)
def handle_file_list(task: TaskPacket):
```

(continues on next page)

(continued from previous page)

```

# Parse task data for file listing, which is structured as:
#
# |<request_number>|<size_of_folder>|<folder>|
with BytesIO(task.data) as data:
    req_no = u32be(data.read(4))
    size = u32be(data.read(4))
    folder = data.read(size).decode()

# Return a debug message that prints which folder was requested for `ls`.
return CallbackDebugMessage(f"You requested to list files in folder: {folder}")

args, options = parse_commandline_options()
client.run(**options)

```

We first import a new helper function called `CallbackDebugMessage()` which we can use to create a debug message that is printed on the Team Server console.

Listing 12: myclient_ls_callback_02.py

```

from io import BytesIO
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options
from dissect.cobaltstrike.client import BeaconCommand, TaskPacket
from dissect.cobaltstrike.client import CallbackDebugMessage
from dissect.cobaltstrike.utils import u32be

client = HttpBeaconClient()

@client.handle(BeaconCommand.COMMAND_FILE_LIST)
def handle_file_list(task: TaskPacket):
    # Parse task data for file listing, which is structured as:
    #
    # |<request_number>|<size_of_folder>|<folder>|
    with BytesIO(task.data) as data:
        req_no = u32be(data.read(4))
        size = u32be(data.read(4))
        folder = data.read(size).decode()

    # Return a debug message that prints which folder was requested for `ls`.
    return CallbackDebugMessage(f"You requested to list files in folder: {folder}")

args, options = parse_commandline_options()
client.run(**options)

```

Here we return a `CallbackDebugMessage()` with our custom string, which the Team Server will receive and output as a debug message.

Ofcourse this is not your standard response to a `ls` command, you can see `scripts/example_client.py` that does implement a proper `ls` response.


```
beacon> ls c:\windows\
[*] Tasked beacon to list files in c:\windows\
[+] host called home, sent: 28 bytes
[-] DEBUG: You requested to list files in folder: c:\windows\*
```

Fig. 1: Debug message shown on the Team Server console

3.1.3 Subclassed client

Instead of using the `@client.handle` decorator to register task handlers you can also subclass `HttpBeaconClient` and adding your own handlers by defining a `on_<command>` method within your class:

Listing 13: echo_client.py

```
from io import BytesIO

from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options
from dissect.cobaltstrike.client import TaskPacket
from dissect.cobaltstrike.client import CallbackDebugMessage
from dissect.cobaltstrike.utils import u32be

class EchoClient(HttpBeaconClient):
    def on_sleep(self, task: TaskPacket):
        with BytesIO(task.data) as data:
            self.sleep_time = u32be(data.read(4))
            self.jitter = u32be(data.read(4))

        return CallbackDebugMessage(
            f"Set new sleep_time: {self.sleep_time}, jitter: {self.jitter}"
        )

    def on_catch_all(self, task: TaskPacket):
        if task is None:
            return

        return CallbackDebugMessage(f"Received {task}")

if __name__ == "__main__":
    client = EchoClient()
    args, options = parse_commandline_options()
    client.run(**options)
```

When the command `COMMAND_SLEEP` is tasked it will call `on_sleep` and we modify the internal sleep timers on the client.

The `on_catch_all` is a special handler that will be called when no handlers are registered for the given Task, acting as a `catch_all` function. This is the same as the `@client.catch_all` decorator.

When we run the `echo_client.py` we see our issued Tasks being echoed back on the Team Server console:

```
$ python3 echo_client.py beacon.bin -v
```

```

beacon> sleep 0
[*] Tasked beacon to become interactive
[+] host called home, sent: 16 bytes
[-] DEBUG: Set new sleeptime: 100, jitter: 90
beacon> run whoami
[*] Tasked beacon to run: whoami
[+] host called home, sent: 24 bytes
[-] DEBUG: Received <TaskPacket epoch=0x635bc5e5, total_size=0x60, command=<BeaconCommand.COMMAND_EXECUTE_JOB: 78>, size=0x10, data=b'\x00\x00\x00\x00\x00\x00\x06whoami\x00\x00'>
beacon> ls
[*] Tasked beacon to list files in .
[+] host called home, sent: 19 bytes
[-] DEBUG: Received <TaskPacket epoch=0x635bc5e7, total_size=0x63, command=<BeaconCommand.COMMAND_FILE_LIST: 53>, size=0xb, data=b'\xff\xff\xff\xfe\x00\x00\x00\x03.\*'>
[DANIEL-PC] daniel.thomas/1720 (x64) last: 142ms
beacon>

```

Fig. 2: Echo beacon client echoing all TaskPackets back to the Team Server console

3.1.4 Next steps

This concludes the tutorial that showed how to implement a beacon client using a decorator and a subclass. It also showed how to parse Task data and send back Callback data to the Team Server.

You can take a look at [scripts/example_client.py](#) for a more detailed implemented client.

3.2 Decrypt Cobalt Strike PCAPs

In this tutorial we will show how to decrypt a beacon session in a PCAP file using a known RSA Private key with the CLI tool `beacon-pcap` that is installed by the `dissect.cobaltstrike` package.

There are some prerequisites to be able to decrypt Cobalt Strike C2 traffic:

- The beacon payload of the session that can be loaded by [BeaconConfig](#).
 - If not specified it will try to find a staged beacon payload in the PCAP.
- One of the following Cryptographic keys is required:
 - AES key of the beacon session (HMAC key is optional)
 - AES rand bytes of the beacon session (this can derive both the AES and HMAC key)
 - RSA Private key of the Team Server (this can decrypt the BeaconMetadata for all sessions)
- If the C2 traffic is over HTTPS/TLS then a `SSLKEYLOGFILE` is also required.

3.2.1 Installation

This tutorial will take care of the requirements above, so let's get started. First we ensure that we have `dissect.cobaltstrike` installed with PCAP support:

```
$ pip install dissect.cobaltstrike[pcap]
```

This also installs the `pyshark` Python package but it still requires the `tshark` binary from [Wireshark](#) to work. Ensure that you have [Wireshark](#) installed or install the `tshark` binary, for example on Ubuntu or Debian systems you can install it with:

```
$ apt install tshark
```

Verify that `beacon-pcap` runs by passing it `--help`:

```
$ beacon-pcap --help
usage: beacon-pcap [-h] [-f FILTER] [-c C2] [-n NSS_KEYLOG_FILE] [-a AES] [-m HMAC] [-k]
[-p PRIVATE_KEY] [-b BEACON] [-A] [-v] [-e] [-w WRITER] PCAP

positional arguments:
  PCAP                  PCAP to parse

optional arguments:
  -h, --help            show this help message and exit
  -f FILTER, --filter FILTER
                        Wireshark display filter to apply while parsing PCAP (default:
↳ None)
  -c C2, --c2 C2        Cobalt Strike C2 ip address (default: None)
  -n NSS_KEYLOG_FILE, --nss-keylog-file NSS_KEYLOG_FILE
                        NSS keylog file to use for decrypting SSL traffic (default: None)
  -a AES, --aes AES     AES key to use (in hex) (default: None)
  -m HMAC, --hmac HMAC  HMAC key to use (in hex) (default: None)
  -k, --no-hmac-verify  Disable HMAC signature verification (default: False)
  -p PRIVATE_KEY, --private-key PRIVATE_KEY
                        Path to RSA private key (default: None)
  -b BEACON, --beacon BEACON
                        Use the BeaconConfig from this Beacon (default: None)
  -A, --all-metadata    Dump all metadata and not only unique (default: False)
  -v, --verbose         Increase verbosity (default: 0)
  -e, --extract-beacons
                        Extract found beacons in pcap (default: False)
  -w WRITER, --writer WRITER
                        Record writer (default: None)
```

3.2.2 Getting the Beacon

The PCAP we are going to use is from *Malware Traffic Analysis* and can be downloaded from here:

- <https://www.malware-traffic-analysis.net/2021/06/15/index.html>

```
$ wget https://www.malware-traffic-analysis.net/2021/06/15/2021-06-15-Hancitor-with-
↳ Ficker-Stealer-and-Cobalt-Strike.pcap.zip
$ 7z x 2021-06-15-Hancitor-with-Ficker-Stealer-and-Cobalt-Strike.pcap.zip -pinfected
```

After the PCAP is extracted we can do a preliminary analysis to find any staged beacon payloads in the PCAP and extract them:

```
$ beacon-pcap --extract-beacons 2021-06-15-Hancitor-with-Ficker-Stealer-and-Cobalt-
↳ Strike.pcap
[+] Found <BeaconConfig ['5.252.177.17']> at b'/ZsDK', extracted beacon payload to
↳ 'beacon-ZsDK.bin'
[+] Found <BeaconConfig ['5.252.177.17']> at b'/8mJm', extracted beacon payload to
↳ 'beacon-8mJm.bin'
```

We see two beacons being extracted, this most likely indicates that there are two beacon sessions in the PCAP. If you don't provide `--extract-beacons` then it will try to find the (first) staged beacon payload in the PCAP and uses that to parse the C2 traffic.

Hint: If no beacons are found in a PCAP you could try looking for the beacon config in the [Cobalt Strike Beacon Dataset](#) and extract the `config_block` field.

The beacon is required as it contains the configuration on how to communicate with its Team Server and thus is needed to be able to correctly parse and decrypt the C2 traffic in the PCAP.

3.2.3 RSA Private Key

Now that the beacons are extracted, we inspect the *RSA Public Key* of the beacons using:

```
$ beacon-dump -t raw beacon-8mJm.bin | grep PUBKEY
<Setting index=<BeaconSetting.SETTING_PUBKEY: 7>, type=<SettingsType.TYPE_PTR: 3>,
↳ length=0x100, value=b"\x81\x9f0\r\x06\t*\x86H\x86\xf7r\x01\x01\x01\x05\x00\x03\x81\
↳ x8d\x000\x81\x89\x02\x81\x81\x00\xa78\xcd\xe7_\x1f\xbb\x1c\x18dl7~\x03\x01k\x16+\x12\
↳ xbar\xbd\xfd\xdc6\xb4\xcd.N\x9b\xae\x12 Z\x95\xc2ap\xbf\x90\x81\x05\xad\x7f\xa4\xbb\
↳ xcc\xfa\x862&\x1b\xed\x98p\x9u\x9f2\x07\x94\xe1\xfeI\x95#\xd7\x1f\x08\xa5l\xae\x03\
↳ x15\xbf\xde=l\x8a\x168k\x03\xb7\xa6U\x1a\xa13mP2Z5\x00\xdb'\xd7\x8a\xd8\xfd\x13\xb6\
↳ xa7;\x9f\xb7\xc3\xfbMz\x08\x8e2?\x07a\x86V\xec\xd85\x95\xfa_\x826\x13\x02\x03\x01\x00\
↳ x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
↳ x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
↳ x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
↳ x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
↳ x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
↳ x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00">
```

Our goal is to find out if we can find a matching **RSA Private key** on VirusTotal. When we query VirusTotal for the Public Key bytes we can find that there are some malware samples but also a file called Cobalt Strike 4.3.zip.

FILES 3		Sort by ▾		Filter by ▾	Export ▾	Tools ▾	Help ▾
		Detections	Size	First seen	Last seen	Submitters	
<input type="checkbox"/>	2D7A14857EB68E05D8A03C92E9207A36379BEAC956E67CEFC083EFB0715C9A8 4696.dmp	0 / 56	21.95 MB	2021-11-26 09:36:13	2021-11-26 09:36:13	1	
<input type="checkbox"/>	Cobalt Strike 4.3.zip zip cve-2020-0796 exploit cve-2017-0213	47 / 65	27.97 MB	2021-07-23 09:41:12	2021-07-23 09:41:12	1	
<input type="checkbox"/>	2B7C73D4A827D0F2EE7CE4ACDD4656951F271AE364236AC08FA26EACF2912199 dllhost.exe	0 / 58	77.43 MB	2021-08-15 00:51:57	2021-08-15 00:51:57	1	

Fig. 3: Leaked version of Cobalt Strike 4.3 on VirusTotal (hash redacted)

This is a leaked version of Cobalt Strike containing a file called `.cobaltstrike.beacon_keys`, embedded in this file is a **RSA Private key**. which we can extract using the following Python script `dump_beacon_keys.py`

```
$ file .cobaltstrike.beacon_keys
.cobaltstrike.beacon_keys: Java serialization data, version 5

$ python3 dump_beacon_keys.py
-----BEGIN RSA PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAKc4zedfh7scGGRsN34DAWsWKxK6
cr333Da0zS5Om64SIFqVwmFwv5CBBa1/pLvM+nmGMiYb7Zhw+XXyB5Th/kmVI9cfCKVsrgMVv949
```

(continues on next page)

(continued from previous page)

```

bIoWOGsDt6ZVGqEzbVAyWjUA2yfXitj9E7an05+3w/tNegiOMj8HYYZW7Ng1lfpfgjYTAgMBAAEC
gYBZ63DFTuB4NBZlwc9hQmp71BLbYkbbH/JZtIV0ti5+vx6It2ksDn3kTYzpC+9gUUwLFv9WgMQV
qgJqyvgKti+PMGmMcTJTdd1GpEt3dzhwNzEuScWdxaA0IJZ0NfdMrGcDogHsNDG4YAjg2XP6d1eZ
vHuIYwNycKM4KcCB5suqEQJBA0JdR3jg0eHly2W+ODb11krwbQV0xu0wP3j2veie8tnkuTK3Nfwm
Slx6PSp8ZtABh8PcpRw+91j9/ecFZMHC60kCQQC9HVV200hWnXEdWspC/YCMH3CFxc7SFRgDYK2r
1sVTQU/fTM2bkdaZXDWIZjbLF0b0U7/zQfVsuuZyGMFwdwmbAkBiDxJ1FL8W4pr32i0z8c8A66Hu
mK+j1qfIWOrvqFt/dIudoqwgLNQtt25jxzwwg18yw5Rq5gP0cyLYPwfkv/BxAkAtLhnh5ezr7Hc+
pRcXRAR27vp7aUIia0QAwPavtermTnkxiuE1CWpw97CNHE4uUin7G46RnLExC4T6hgkrzurAkEA
vRVFGcXTmcg49Ha3VIKib83xlNhBnWVqkNyLnAdOBENZUZ479oaPw7Sl+N0SD15TgT25+4P6PKH8
QE6hwC/g5Q==
-----END RSA PRIVATE KEY-----

```

```
$ python3 dump_beacon_keys.py > key.pem
```

3.2.4 Decrypt C2 Traffic

After extracting this key we have the *RSA Private Key* in PEM format that we can use to decrypt beacon sessions by passing it to beacon-pcap using the `-p / --private-key` argument. It accepts both DER and PEM formatted key files.

```

$ beacon-pcap -p key.pem 2021-06-15-Hancitor-with-Ficker-Stealer-and-Cobalt-Strike.pcap -
  ↪ -beacon beacon-8mJm.bin
<Beacon/BeaconMetadata packet_ts=2021-06-15 15:08:55.172675 src_ip=net.ipaddress('10.0.0.
  ↪ 134') src_port=52886 dst_ip=net.ipaddress('5.252.177.17') dst_port=443 raw_http=b'GET /
  ↪ activity HTTP/1.1\r\nAccept: */*\r\nCookie: kR/OTFMhCYQpv09cXl2R7qEespVUFQ/
  ↪ 8YahAbs1b+rEESbSzcAc44R9K1f4zH4GGYxT4dErzNQWimmMW5wQVQSEGFZ36mWc/
  ↪ beoUTQUGVUxcZWxl0t8WB012qC6vsmRSV5uQ0+qxz0Lbz1P/wOkWwbNM0XF9LhVjRrGYSR0Jlrc=\r\nUser-
  ↪ Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727)\r\nHost:
  ↪ 5.252.177.17:443\r\nConnection: Keep-Alive\r\nCache-Control: no-cache\r\n\r\n'
  ↪ magic=48879 size=92 aes_rand=b'\xf9da\xc8\x8b\x07\xe1:\xfa\np\xbc{m\xe0' ansi_
  ↪ cp=58372 oem_cp=46337 bid=693615746 pid=6396 port=0 flag=4 ver_major=10 ver_minor=0
  ↪ ver_build=19042 ptr_x64=0 ptr_gmh=1972243040 ptr_gpa=1972237648 ip=net.ipaddress('134.
  ↪ 5.7.10') info=b'DESKTOP-X9JH6AW\ttabitha.gomez\tsvchost.exe'>
<Beacon/TaskPacket packet_ts=2021-06-15 15:09:56.371968 src_ip=net.ipaddress('5.252.177.
  ↪ 17') src_port=443 dst_ip=net.ipaddress('10.0.0.134') dst_port=52894 raw_http=b'HTTP/1.
  ↪ 1 200 OK\r\nDate: Tue, 15 Jun 2021 15:09:55 GMT\r\nContent-Type: application/octet-
  ↪ stream\r\nContent-Length: 48\r\n\r\nP\xc1\xf1\xa0{3 \xa8\x01}\xfe\xbc1\x8e\xa2\x81\
  ↪ xd7A2\xa3;\xe0\x91\xf5\x90\xdd]\xc5\x88\xa2\x88\x93\x14-\xb4\xbb\x96\xf1\x1c\xd7\r\
  ↪ xa60\xfe\xc5\x9e\xd6' epoch=2021-06-15 15:09:55 total_size=16 command='COMMAND_SLEEP'
  ↪ size=8 data=b'\x00\x00\x00d\x00\x00\x00Z'>

```

We specify a beacon specifically as there are two beacon sessions in this PCAP but they have slightly different urls. If you want to decrypt the other session just pass the other beacon as the parameter using `--beacon`.

3.2.5 Export C2 traffic as records

By default beacon-pcap will output decrypted C2 traffic to stdout as *flow.record* format. You can redirect the records to a file or write them to a file using `-w / --writer` or even pipe it directly to `rdump`.

Example of writing the decrypted C2 records to the file `c2.records.gz`:

```
$ beacon-pcap -w c2.records.gz -p key.pem 2021-06-15-Hancitor-with-Ficker-Stealer-and-
↳ Cobalt-Strike.pcap --beacon beacon-8mJm.bin
```

Hint: By specifying `.gz` as the filename extension the file is automatically gzip compressed by `flow.record`. It supports many other compression algorithms, such as `bz2`, `lz4` and `zstd`. However, they might need additional dependencies.

3.2.6 Dumping records with rdump

Next we can use the `rdump` tool from the `flow.record` package to read and inspect the saved records. By using the `-s / --selector` argument in `rdump` we can filter records and `-f / --filter` to specify an output format string.

For example to list all the *COMMANDS* issued by the Team Server:

```
$ rdump c2.records.gz -s "r.command" -f "{packet_ts} {src_ip}:{src_port} | {command}"
2021-06-15 15:09:56.371968 5.252.177.17:443 | COMMAND_SLEEP
2021-06-15 15:10:12.291611 5.252.177.17:443 | COMMAND_INLINE_EXECUTE_OBJECT
2021-06-15 15:10:30.437461 5.252.177.17:443 | COMMAND_SPAWN_TOKEN_X86
2021-06-15 15:11:10.851089 5.252.177.17:443 | COMMAND_FILE_LIST
2021-06-15 15:11:18.131182 5.252.177.17:443 | COMMAND_FILE_LIST
```

Example to list all the *CALLBACKS* sent by the beacon:

```
$ rdump c2.records.gz -s "r.callback" -f "{packet_ts} {src_ip}:{src_port} | {callback}"
2021-06-15 15:10:12.618050 10.0.0.134:52914 | CALLBACK_PENDING
2021-06-15 15:10:33.171933 10.0.0.134:52933 | CALLBACK_PORTSCAN
2021-06-15 15:10:40.932358 10.0.0.134:52943 | CALLBACK_PORTSCAN
2021-06-15 15:10:50.772303 10.0.0.134:52960 | CALLBACK_PORTSCAN
2021-06-15 15:11:11.251795 10.0.0.134:52983 | CALLBACK_PENDING
```

Notice that the client sent some *portscan* data back, we can inspect the portscan callback data specifically:

```
$ rdump c2.records.gz -s "r.callback == 'CALLBACK_PORTSCAN'" -f "{packet_ts} | {data}"
2021-06-15 15:10:33.171933 | b"(ICMP) Target '10.7.5.2' is alive. [read 8 bytes]\n(ICMP)
↳ Target '10.7.5.7' is alive. [read 8 bytes]\n\xd8\xca`\x05"
2021-06-15 15:10:40.932358 | b"(ICMP) Target '10.7.5.134' is alive. [read 8 bytes]\nF\rEg
↳ "
2021-06-15 15:10:50.772303 | b'10.7.5.7:445 (platform: 500 version: 10.0 name: STORMRUN-
↳ DC domain: STORMRUNCREEK)\n10.7.5.134:445 (platform: 500 version: 10.0 name: DESKTOP-
↳ X9JH6AW domain: STORMRUNCREEK)\nScanner module is complete\n\x00\x00\x00\x00'
```

As you can see it's quite easy and powerful to be able to inspect the beacon traffic stored as records using `rdump`. This is a great way to get a quick overview of the traffic and to extract the relevant data you need for further analysis.

`rdump` has many different output formats, such as `json` by using the `-j / --json` argument:

```
$ rdump c2.records.gz --json
...
{
  "packet_ts": "2021-06-15T15:08:55.172675",
  "src_ip": "10.0.0.134",
  "src_port": 52886,
  "dst_ip": "5.252.177.17",
  "dst_port": 443,
  "raw_http":
  ↪ "R0VUIC9hY3Rpdml0eSBIVFRQLzEuMQ0KQWNjZXB00iAqLyONckNvb2tpZToga1IvT1RGTWhDWVFwdjA5Y1hsMlI3cUVlc3BWWVZR
  ↪ ",
  "magic": 48879,
  "size": 92,
  "aes_rand": "+WRByIsH4Tr6CnC8e2Bt4A==",
  "ansi_cp": 58372,
  "oem_cp": 46337,
  "bid": 693615746,
  "pid": 6396,
  "port": 0,
  "flag": 4,
  "ver_major": 10,
  "ver_minor": 0,
  "ver_build": 19042,
  "ptr_x64": 0,
  "ptr_gmh": 1972243040,
  "ptr_gpa": 1972237648,
  "ip": "134.5.7.10"
}
...
{
  "packet_ts": "2021-06-15T15:09:56.371968",
  "src_ip": "5.252.177.17",
  "src_port": 443,
  "dst_ip": "10.0.0.134",
  "dst_port": 52894,
  "raw_http":
  ↪ "SFRUUC8xLjEgMjAwIE9LDQpEYXRlOiBUdWUsIDE1IEp1biAyMDIxIDE0jA50jU1IEEdNVA0KQ29udGVudC1UeXB1OiBhCHBsaWNh
  ↪ pQwfGgezMgqAF9/rxsjqKB10EyozvqkfWQ3V3FiGCiiJMULbS7lvEc1w2mMP7FntY=",
  "epoch": "2021-06-15T15:09:55.000000",
  "total_size": 16,
  "command": "COMMAND_SLEEP",
  "size": 8,
  "data": "AAAAZAAAAFo="
}
}
```

We recommend to get familiar with the `rdump` tool and the `flow.record` package by going to the documentation here: <https://docs.dissect.tools/en/latest/tools/rdump.html>

See also:

Other useful resources that can help by analysing Cobalt Strike traffic:

- Series: Cobalt Strike: Decrypting Traffic by NVISO.
- Analysing a malware PCAP with IcedID and Cobalt Strike traffic by NETRESEC.

- [Cobalt Strike Analysis and Tutorial: CS Metadata Encryption and Decryption](#) by UNIT42.
- [Open Dataset of Cobalt Strike Beacons](#) by Fox-IT part of NCC Group.

SCRIPTS

There are some example and useful scripts in the `scripts` directory of the repository, but also listed here for convenience.

4.1 example_client.py

Here is an example client that handles some more commands and includes a catch-all handler for unhandled commands. It can be found in the file `scripts/example_client.py` but is also listed here for convenience.

```
#!/usr/bin/env python3
#
# Example beacon client
#
# Run with:
# $ python3 example_client.py --help
#
# Recommended to do a dry run first to see how it will connect using which parameters:
# $ python3 example_client.py <beacon_file> -n
#
# Then run it for real in verbose mode:
# $ python3 example_client.py <beacon_file> -v
#
import textwrap
from io import BytesIO

from dissect.cobaltstrike.client import (
    BeaconCallback,
    BeaconCommand,
    CallbackOutputMessage,
    HttpBeaconClient,
    parse_commandline_options,
)
from dissect.cobaltstrike.utils import p32be, u32be

client = HttpBeaconClient()

@client.handle(None)
def on_empty_task(task):
    client.logger.debug("Received empty task.")
```

(continues on next page)

(continued from previous page)

```

@client.catch_all()
def catch_all(task):
    only = "\n".join(
        [
            ",___",
            "{0,o}",
            "|)``)",
            "O RLY?",
            "",
        ]
    )
    return CallbackOutputMessage(textwrap.indent(only, "\t"))

@client.handle(BeaconCommand.COMMAND_FILE_LIST)
def on_file_list(task):
    # Parse task data for file listing
    with BytesIO(task.data) as data:
        req_no = u32be(data.read(4))
        size = u32be(data.read(4))
        folder = data.read(size).decode()

    # Create file list response buffer
    buffer = "\n".join(
        [
            folder,
            "{type}\t{size}\t{date}\t{name}".format(type="D", size=0, date="04/10 2022",
↳ 13:33:37", name="."),
            "{type}\t{size}\t{date}\t{name}".format(type="D", size=0, date="04/10 2022",
↳ 13:33:37", name=".."),
            "{type}\t{size}\t{date}\t{name}".format(type="D", size=0, date="04/10 2022",
↳ 13:33:37", name="srsly?"),
            "{type}\t{size}\t{date}\t{name}".format(type="F", size=36, date="04/10 2022",
↳ 13:33:37", name="flag.txt"),
        ]
    )

    # <request_number>/buffer/<zero termination>
    buffer = p32be(req_no) + buffer.encode() + p32be(0)
    return BeaconCallback.CALLBACK_PENDING, buffer

@client.handle(BeaconCommand.COMMAND_DOWNLOAD)
def on_download(task):
    # from https://github.com/desaster/kippo
    nowai = "\n".join(
        [
            "___",
            "{o,o}",
            "(__(|",
            "'-''-'",
        ]
    )

```

(continues on next page)

(continued from previous page)

```

        "NO WAI!",
        "",
    ]
)
fid = 100
size = len(nowai)
file_name = b"flag.txt"
client.send_callback(BeaconCallback.CALLBACK_FILE, p32be(fid) + p32be(size) + file_
↪ name + p32be(0))
client.send_callback(BeaconCallback.CALLBACK_FILE_WRITE, p32be(fid) + nowai.encode()
↪ + p32be(0))
client.send_callback(BeaconCallback.CALLBACK_FILE_CLOSE, p32be(fid) + p32be(0))

@client.handle(BeaconCommand.COMMAND_SLEEP)
def on_sleep(task):
    with BytesIO(task.data) as data:
        client.sleeptime = u32be(data.read(4))
        client.jitter = u32be(data.read(4))
        client.logger.info("Set new sleeptime: %u, jitter: %u", client.sleeptime, client.
↪ jitter)

@client.handle(BeaconCommand.COMMAND_PWD)
def on_pwd(task):
    cwd = f"C:\\Users\\{client.user}\\Documents\\"
    return BeaconCallback.CALLBACK_PWD, cwd.encode() + p32be(0)

if __name__ == "__main__":
    args, options = parse_commandline_options(
        defaults=dict(
            user="O RLY?",
            computer="YA RLY",
        )
    )
    client.run(**options)

```

4.2 checksum8-accesslogs.py

```

#!/usr/bin/env python3
#
# Simple script to check the checksum8 of accesslogs
#
import argparse
import collections
import datetime
import re
import sys

```

(continues on next page)

(continued from previous page)

```

from dissect.cobaltstrike import utils

RE_ACCESS_LOG = re.compile(
    r"""
    [^\d]*           # ignore front matter (eg: filename from grep out or
↳ logger name)
    (?P<ip>\d+\.\d+\.\d+\.\d+) # the IP address
    .*              # Ignore other stuff (could be a space, or username in
↳ case of nginx or apache)
    \[(?P<time>.+)\] # the date and time
    \s+             # ignore spaces
    "(?P<request>.*)" # the request
    \s+             # ignore spaces
    (?P<status>[0-9]+) # the status
    \s+             # ignore spaces
    (?P<size>\S+)     # the size
    \s+             # ignore spaces
    "(?P<referrer>.*)" # the referrer
    \s+             # ignore spaces
    "(?P<agent>.*)"   # the user agent
    """,
    re.VERBOSE,
)

def build_parser():
    parser = argparse.ArgumentParser(description="checksum8 accesslogs")
    parser.add_argument("--stats", action="store_true", help="show monthly stats")
    parser.add_argument("-l", "--length", type=int, help="truncate output to this length
↳ ")
    parser.add_argument("-b", "--brief", action="store_true", help="brief output (no
↳ user agent)")
    parser.add_argument("-d", "--datefmt", default=None, help="date format")
    return parser

@utils.catch_sigpipe
def main():
    parser = build_parser()
    args = parser.parse_args()

    stats = collections.Counter()

    print("[reading from stdin..]", file=sys.stderr)
    for line in sys.stdin:
        match = RE_ACCESS_LOG.match(line)
        if not match:
            continue
        ip = match.group("ip")
        apache_stamp = match.group("time")
        request = match.group("request")
        agent = match.group("agent")

```

(continues on next page)

(continued from previous page)

```

dt = datetime.datetime.strptime(apache_stamp, "%d/%b/%Y:%H:%M:%S %z")
method, _, uri = request.partition(" ")
uri, _, version = uri.partition(" ")
if utils.is_stager_x86(uri) or utils.is_stager_x64(uri):
    beacon = "x64" if utils.is_stager_x64(uri) else "x86"
    if args.stats:
        fmt = args.datefmt or "%Y-%m"
        stats[dt.strftime(fmt)] += 1
        continue
    if args.datefmt:
        dt = dt.strftime(args.datefmt)
    if args.brief:
        out = f"{dt} - beacon {beacon} - {method} {uri}"
    else:
        out = f"{dt} - beacon {beacon} - {method} {ip} {uri} - {agent}"

    if args.length and len(out) > args.length:
        out = out[: args.length] + "..."
    print(out)

if args.stats:
    print("date,requests")
    for month, value in sorted(stats.items()):
        print(f"{month},{value}")

if __name__ == "__main__":
    sys.exit(main())

```

4.3 dump_beacon_keys.py

```

#!/usr/bin/env python3
#
# This script dumps the RSA Private Key from `.cobaltstrike.beacon_keys`.
#
# It requires the javaobj module, install it with:
#
#   $ pip install javaobj-py3
#
import base64

import javaobj

key = javaobj.loads(open(".cobaltstrike.beacon_keys", "rb").read())
privkey_der = bytes(c & 0xFF for c in key.array.value.privateKey.encoded)

print("-----BEGIN RSA PRIVATE KEY-----")
print(base64.encodebytes(privkey_der).strip().decode())
print("-----END RSA PRIVATE KEY-----")

```

- Example beacon client – *example_client.py* ^{only?}
- Check webserver logs for *checksum8* requests – *checksum8-accesslogs.py*
- Dump RSA Private Key from a *.cobaltstrike.beacon_keys* file – *dump_beacon_keys.py*

BEACON-ARTIFACT

The command `beacon-artifact` can be used to dump payloads from executables generated by [ArtifactKit](#). Usually the artifact executable is a stageless beacon, but it could also contain stager shellcode.

```
$ beacon-artifact <artifactkit-file> | xxd
```

The `beacon-artifact` tool only dumps the extracted payload (default to stdout). If the extracted payload is a beacon (stageless artifact) and not a stager, you can pipe the output directly to `beacon-dump -` to dump the beacon configuration.

```
$ beacon-artifact <artifactkit-file> | beacon-dump -
```

If the command is not in your path, you can also run the command using the following Python module:

```
$ python -m dissect.cobaltstrike.artifact --help
```

5.1 beacon-artifact - CLI interface

```
beacon-artifact [-h] [-v] [-o OUTPUT] FILE
```

5.1.1 beacon-artifact positional arguments

- **FILE** - FILE to decode (default: None)

5.1.2 beacon-artifact options

- **-h, --help** - show this help message and exit
- **-v, --verbose** - verbosity level (-v for INFO, -vv for DEBUG) (default: 0)
- **-o OUTPUT, --output OUTPUT** - write decoded ArtifactKit payload to FILE (default: -)

BEACON-CLIENT

The command `beacon-client` can be used to connect to a Cobalt Strike Team Server given a beacon config or payload. It will read the beacon settings so it can communicate with the C2 server, and then it will start do check-ins and retrieve Tasks like a real beacon.

Tip: If you enable `-v / --verbose` logging, and you have the `rich` module installed. It will automatically use `rich` to render the console logging which can be easier on the eyes.

The implementation of the client in `beacon-client` is observing only, meaning it does not implement any of the beacon functionality such as executing commands or listing files and does not send any Callback data to the Team Server.

If you want to know how to implement your own custom beacon client that can respond to tasks, please refer to this [tutorial](#).

The `--writer` parameter of `beacon-client` allows you to log the retrieved *beacon tasks* to a file. This can be useful for debugging or logging of tasks that are being sent. The output is written as *flow.record* records and can be dumped using the tool `rdump` which is part of the `flow.record` package and is installed as a dependency.

To ensure you have all the dependencies for `beacon-client` you can use the following pip command:

```
$ pip install -e dissect.cobaltstrike[c2]
```

Here is an example usage of connecting to a Team Server with custom Beacon metadata, we choose a fixed beacon id so we can connect to it again later without creating a new beacon session at the Team Server:

```
$ beacon-client beacon.bin -vi 1234 --user "wing" --computer "safecomputer" -w c2.  
↪records.gz
```

- This will launch the beacon-client using `beacon.bin` as the BeaconConfig.
- The `-v` flag will enable verbose logging. (recommend to see what is going on)
- The `-i` flag will set the Beacon ID to 1234.
- The `--user` and `--computer` arguments are used to set the username and computer name in the Beacon Metadata.
- and `-w` or `--writer` writes decrypted C2 packets such as Tasks and Callback packets to the file `c2.records.gz`.

There are many more options that can be overridden, by default most settings are randomized. To see all the options run it with `--help` and is also documented here: [beacon-client - CLI interface](#).

6.1 Dumping saved records

The contents of `c2.records.gz` can then be dumped using the `rdump` (record dump) tool:

```
$ rdump c2.records.gz
```

For more advanced usage of `rdump` use `--help` or see the documentation for [flow.record](#).

If `beacon-client` is not in your path, you can also run the command using the following Python module:

```
$ python -m dissect.cobaltstrike.client --help
```

6.1.1 beacon-client - CLI interface

```
beacon-client [-h] [-d DOMAIN] [-p PORT] [--sleeptime SLEEPTIME] [--jitter JITTER]
               [-c COMPUTER] [-u USER] [-P PROCESS] [-i BEACON_ID] [-I INTERNAL_IP]
               [--arch {x86,x64}] [--barch {x86,x64}] [--high-integrity] [-n] [-w WRITER]
               [-v]
               [-s] [--no-warning]
               BEACON
```

beacon-client positional arguments

- **BEACON** - beacon to use as configuration (default: None)

beacon-client options

- **-h, --help** - show this help message and exit
- **-n, --dry-run** - show settings and exit
- **--no-warning** - disable connect warning

beacon-client beacon communication

- **-d DOMAIN, --domain DOMAIN** - override the domain configured in the beacon (default: None)
- **-p PORT, --port PORT** - override the port configured in the beacon (default: None)

beacon-client beacon sleep options

- **--sleeptime SLEEPTIME** - override sleeptime settings (in milliseconds) (default: None)
- **--jitter JITTER** - override jitter settings (in percentage) (default: None)

beacon-client beacon metadata

- **-c** COMPUTER, **--computer** COMPUTER - computer name (None = random) (default: None)
- **-u** USER, **--user** USER - user name (None = random) (default: None)
- **-P** PROCESS, **--process** PROCESS - process name (None = random) (default: None)
- **-i** BEACON_ID, **--beacon-id** BEACON_ID - beacon id (None = random) (default: None)
- **-I** INTERNAL_IP, **--internal-ip** INTERNAL_IP - internal ip (None = random) (default: None)

beacon-client beacon metadata flags

- **--arch** ARCH - system architecture (None = random) (default: None)
- **--barch** BARCH - beacon architecture (None = random) (default: None)
- **--high-integrity** - set high integrity flag

beacon-client output options

- **-w** WRITER, **--writer** WRITER - record writer (default: None)
- **-v**, **--verbose** - verbosity level (-v for INFO, -vv for DEBUG) (default: 0)
- **-s**, **--silent** - suppress empty task messages

BEACON-DUMP

You can use the command `beacon-dump` to dump configuration from Cobalt Strike beacon payloads.

If the command is not in your path, you can also use run the command using the following Python module:

```
$ python -m dissect.cobaltstrike.beacon --help
```

7.1 XOR keys

The beacon configuration is usually obfuscated using a single-byte XOR key. `beacon-dump` automatically tries all the default xor keys (0x69 and 0x2e).

In case a beacon uses a non default XOR key you can specify the `-a` or `--all-xor-keys` argument to check all possible single byte XOR keys. Note that this option is not recommended for large payloads such as memory dumps.

You can also use the `-x` or `--xorkey` option to specify a known XOR key or a set of keys by repeating the argument:

```
$ beacon-dump -x 0xAC -x 0xCE -x 0x55 -x 0xED <beacon-file>
```

7.2 Output format

The output format can be specified using the `-f` or `--format` option. The following formats are supported:

- **normal**: output the beacon configuration in a human readable format of key value pairs (default)
- **dumpstruct**: output the beacon settings using `cstruct.dumpstruct`
- **c2profile**: output the beacon configuration as a malleable C2 profile
- **raw**: output the raw beacon configuration

7.2.1 beacon-dump - CLI interface

```
beacon-dump [-h] [-x XORKEY] [-a] [-t {normal,raw,dumpstruct,c2profile}] [-v] FILE
```

beacon-dump positional arguments

- **FILE** - Beacon to dump (default: None)

beacon-dump options

- **-h, --help** - show this help message and exit
- **-x XORKEY, --xorkey XORKEY** - override default xor key(s) (default: -x 0x69 -x 0x2e -x 0x00)
- **-a, --all** - try all other single byte xor keys when default ones fail
- **-t TYPE, --type TYPE** - output format (default: normal)
- **-v, --verbose** - verbosity level (-v for INFO, -vv for DEBUG) (default: 0)

BEACON-PCAP

The command `beacon-pcap` can be used to parse PCAP files containing Cobalt Strike C2 traffic. The AES key of the beacon session or RSA Private key of the Team Server is required to decrypt the traffic.

Tip: If you enable `-v / --verbose` logging, and you have the `rich` module installed. It will automatically use `rich` to render the console logging which can be easier on the eyes.

The beacon config or payload can be specified using the `-b / --beacon` flag, if not specified it tries to find one in the PCAP by checking for any staged beacon payloads. It will always use the first one it finds in the PCAP. If there are multiple staged beacons in the PCAP, you can extract them first using `-e / --extract-beacons` and specify the one you want to use with `--beacon`.

To ensure you have all the dependencies for `beacon-pcap` you can use the following pip command:

```
$ pip install -e dissect.cobaltstrike[pcap]
```

Example usage for if you have the RSA private key:

```
$ beacon-pcap --private-key privkey.der traffic.pcap
```

This will read `traffic.pcap` and use the RSA Private key `privkey.der` for decrypting Beacon Metadata and C2 Packets. As no beacon is specified, it will try to find a staged beacon payload in the PCAP.

By default all the decrypted C2 packets are written as `flow.records` records to `stdout`. The output can be redirected to a file using the `-w / --writer` argument, example:

```
$ beacon-pcap -v -p privkey.der -w beacon-c2.records.gz traffic.pcap
```

This will write the decrypted C2 packets to `beacon-c2.records.gz` instead of `stdout`. The file can then be dumped using the tool `rdump` which is part of the `flow.record` package and is installed as a dependency.

```
$ rdump beacon-c2.records.gz
```

If the command is not in your path, you can also run the command using the following Python module:

```
$ python -m dissect.cobaltstrike.pcap --help
```

8.1 beacon-pcap - CLI interface

```
beacon-pcap [-h] [-f FILTER] [-c C2] [-n NSS_KEYLOG_FILE] [-a AES] [-m HMAC] [-k]
            [-p PRIVATE_KEY] [-b BEACON] [-A] [-v] [-e] [-w WRITER]
            PCAP
```

8.1.1 beacon-pcap positional arguments

- **PCAP** - PCAP to parse (default: None)

8.1.2 beacon-pcap options

- **-h, --help** - show this help message and exit
- **-f FILTER, --filter FILTER** - Wireshark display filter to apply while parsing PCAP (default: None)
- **-c C2, --c2 C2** - Cobalt Strike C2 ip address (default: None)
- **-n NSS_KEYLOG_FILE, --nss-keylog-file NSS_KEYLOG_FILE** - NSS keylog file to use for decrypting SSL traffic (default: None)
- **-a AES, --aes AES** - AES key to use (in hex) (default: None)
- **-m HMAC, --hmac HMAC** - HMAC key to use (in hex) (default: None)
- **-k, --no-hmac-verify** - Disable HMAC signature verification
- **-p PRIVATE_KEY, --private-key PRIVATE_KEY** - Path to RSA private key (default: None)
- **-b BEACON, --beacon BEACON** - Use the BeaconConfig from this Beacon (default: None)
- **-A, --all-metadata** - Dump all metadata and not only unique
- **-v, --verbose** - Increase verbosity (default: 0)
- **-e, --extract-beacons** - Extract found beacons in pcap
- **-w WRITER, --writer WRITER** - Record writer (default: None)

BEACON-XORDECODE

The command `beacon-xordecode` can be used to decode a XorEncoded Cobalt Strike beacon. Not to be confused with the single-byte XOR key that is used to encrypt the Beacon Configuration.

```
$ beacon-xordecode <beacon-file> | xxd
```

If the command is not in your path, you can also use run the command using the following Python module:

```
$ python -m dissect.cobaltstrike.xordecode --help
```

9.1 Nonce offset

A XorEncoded beacon payload consists of the xordecode shellcode stub, the initial *nonce*, the size and then the XorEncoded payload:

```
+-----+-----+-----+-----+
| xordecode shellcode stub | nonce (dword) | payload size (dword) | xorencoded payload |
+-----+-----+-----+-----+
```

To properly decode the XorEncoded payload, the *nonce* offset must be known. The following two different methods are used to determine the *nonce* offset / start of XorEncoded payload:

- Determine *nonce* based on file size, the decoded *size* field is the size of the XorEncoded payload. If it matches it is used as a candidate.
- Determine *nonce* offset based on the end marker of the xordecode shellcode stub.

9.2 MZ header

After the *nonce* candidates have been found it will try to find which of the candidates is the correct one. The MZ header is used to determine the correct candidate. If no MZ header can be found in the payload it will return an error.

You can still use the `-n` or `--nonce-offset` option to manually specify the nonce offset, this will override the automatic nonce and MZ detection.

9.2.1 beacon-xordecode - CLI interface

```
beacon-xordecode [-h] [-n NONCE_OFFSET] [-v] [-o OUTPUT] FILE
```

beacon-xordecode positional arguments

- **FILE** - FILE to decode (default: None)

beacon-xordecode options

- **-h, --help** - show this help message and exit
- **-n NONCE_OFFSET, --nonce-offset NONCE_OFFSET** - Force nonce offset (instead of auto detecting) (default: None)
- **-v, --verbose** - verbosity level (-v for INFO, -vv for DEBUG) (default: 0)
- **-o OUTPUT, --output OUTPUT** - write decoded payload to FILE (default: -)

C2PROFILE-DUMP

The command `c2profile-dump` can be used to parse and dump Malleable C2 profiles. The command is mainly useful for debugging the parsed AST tree. Using the library directly is more useful for extracting information using Python.

```
$ c2profile-dump /path/to/profile.c2
```

To load from a beacon and dump as properties:

```
$ c2profile-dump -b <beacon> -t properties
```

If the command is not in your path, you can also use run the command using the following Python module:

```
$ python -m dissect.cobaltstrike.c2profile --help
```

10.1 c2profile-dump - CLI interface

```
c2profile-dump [-h] [-b] [-a] [-t {pretty,ast,c2profile,properties}] [-v] FILE
```

10.1.1 c2profile-dump positional arguments

- **FILE** - c2 profile or beacon to dump (default: None)

10.1.2 c2profile-dump options

- **-h, --help** - show this help message and exit
- **-b, --beacon** - input is a beacon instead of a .profile file
- **-a, --all** - when using `--beacon`, try all xor keys when default ones fail
- **-t TYPE, --type TYPE** - output format (default: pretty)
- **-v, --verbose** - verbosity level (-v for INFO, -vv for DEBUG) (default: 0)

DISSECT.COBALTSTRIKE

11.1 Submodules

11.1.1 dissect.cobaltstrike.artifact

This module is responsible for dumping payloads from [ArtifactKit](#) generated executables.

Module Contents

Classes

<i>ArtifactKitPayload</i>	Namedtuple containing the ArtifactKit metadata and decoded payload
---------------------------	--

Functions

<i>iter_artifactkit_payloads</i> (→ tor[ArtifactKitPayload])	Itera-	Iterate over found <i>ArtifactKitPayload</i> by scanning <i>fobj</i> for possible ArtifactKit payloads.
<i>main</i> ()		Entrypoint for <i>beacon-artifact</i>

Attributes

<i>logger</i>

`dissect.cobaltstrike.artifact.logger`

class `dissect.cobaltstrike.artifact.ArtifactKitPayload`

Bases: `NamedTuple`

Namedtuple containing the ArtifactKit metadata and decoded payload

offset: `int`

Offset of the ArtifactKit metadata in the file

size: `int`

Size of the payload

xorkey: `bytes`

4-byte random xor mask

hints: `bytes`

Loader hints (GetModuleHandleA, GetProcAddress)

payload: `bytes`

Decoded ArtifactKit payload

`dissect.cobaltstrike.artifact.iter_artifactkit_payloads(fobj: BinaryIO, start_offset: int | None = 0, maxrange: int | None = None) → Iterator[ArtifactKitPayload]`

Iterate over found [ArtifactKitPayload](#) by scanning *fobj* for possible ArtifactKit payloads.

Side effects: file position due to seeking

Note: No additional checks are done on the ArtifactKit payloads to ensure that what is found is actually correct.

Parameters

- **fobj** – file-like object
- **start_offset** – starting offset to search for ArtifactKit payloads, if *None* it will search from current offset. (default: 0)
- **maxrange** – maximum file offset to limit search to, if *None* it will search the entire file (default: *None*)

Yields

[ArtifactKitPayload](#)

`dissect.cobaltstrike.artifact.main()`

Entrypoint for *beacon-artifact*

11.1.2 dissect.cobaltstrike.beacon

This module is responsible for extracting and parsing configuration from Cobalt Strike beacon payloads.

Module Contents

Classes

[BeaconConfig](#)

A [BeaconConfig](#) object represents a single Beacon configuration

Functions

<code>find_beacon_config_bytes(→ Iterator[bytes])</code>	Find and yield (possible) Cobalt Strike configuration bytes from file <i>fh</i> using <i>xorkey</i> (eg: b"x69").
<code>iter_beacon_config_blocks(→ Iterator[Tuple[bytes, dict]])</code>	Yield tuple with found Beacon <i>config_block_bytes</i> from file <i>fobj</i> and <i>extra_info</i> dict
<code>make_byte_list(→ List[bytes])</code>	Return all single-byte bytes as an ordered list, excluding <i>exclude</i> bytes.
<code>iter_settings(→ Iterator[Setting])</code>	Returns an iterator yielding <i>Setting</i> objects by reading data from <i>fobj</i>
<code>grouper(iterable, n[, fillvalue])</code>	Collect data into fixed-length chunks or blocks
<code>parse_recover_binary(→ List[Tuple[str, Union[int, bool]]])</code>	Parse <code>SETTING_C2_RECOVER</code> (<i>.http-get.server.output</i>) data
<code>parse_transform_binary(→ List[Tuple[str, Union[str, ...]])</code>	Parse <code>SETTING_C2_{REQUEST,POSTREQ}</code> (<i>.http-{get,post}.client</i>) data
<code>parse_execute_list(→ List[str])</code>	Parse <code>SETTING_PROCIJN_EXECUTE</code> (<i>.process-inject.execute</i>) data
<code>parse_process_injection_transform_steps(→ list)</code>	Parse <code>SETTING_PROCIJN_TRANSFORM_X{86,64}</code> (<i>.process-inject.transform-x{86,64}</i>) data
<code>parse_gargle(→ list)</code>	Parse <code>SETTING_GARGLE_SECTIONS</code> (<i>.stage.{sleep_mask,obfuscate,userwx}</i>) data
<code>parse_pivot_frame(→ bytes)</code>	Parse <code>SETTING_{TCP,SMB}_FRAME_HEADER</code> (<i>.{tcp,smb}_frame_header</i>) data
<code>sha256sum_pubkey(→ str)</code>	Return the SHA-256 digest of <i>der_data</i>
<code>null_terminated_bytes(→ bytes)</code>	Return null terminated <i>data</i> as bytes.
<code>null_terminated_str(→ str)</code>	Return null terminated <i>data</i> as string. Non ascii characters are ignored.
<code>build_parser()</code>	
<code>main()</code>	Entrypoint for beacon-dump.

Attributes

<i>logger</i>	
<i>CS_DEF</i>	
<i>cs_struct</i>	
<i>TransformStep</i>	
<i>BeaconSetting</i>	
<i>DeprecatedBeaconSetting</i>	
<i>SettingsType</i>	
<i>Setting</i>	
<i>BeaconProtocol</i>	
<i>CryptoScheme</i>	
<i>ProxyServer</i>	
<i>InjectAllocator</i>	
<i>InjectExecutor</i>	
<i>DEFAULT_XOR_KEYS</i>	Default XOR keys used by Cobalt Strike for obfuscating Beacon config bytes
<i>SETTING_TO_PRETTYFUNC</i>	BeaconSetting enum to pretty function mapping

`dissect.cobaltstrike.beacon.logger`

`dissect.cobaltstrike.beacon.CS_DEF = Multiline-String`

```
"""
enum BeaconSetting: uint16 {
    SETTING_PROTOCOL = 1,
    SETTING_PORT = 2,
    SETTING_SLEEPTIME = 3,
    SETTING_MAXGET = 4,
    SETTING_JITTER = 5,
    SETTING_MAXDNS = 6,
    SETTING_PUBKEY = 7,
    SETTING_DOMAINS = 8,
    SETTING_USERAGENT = 9,
    SETTING_SUBMITURI = 10,
    SETTING_C2_RECOVER = 11,
    SETTING_C2_REQUEST = 12,
    SETTING_C2_POSTREQ = 13,
    SETTING_SPAWNTO = 14,          // releasenotes.txt
}
```

(continues on next page)

(continued from previous page)

```

// CobaltStrike version >= 3.4 (27 Jul, 2016)
SETTING_PIPENAME = 15,

SETTING_KILLDATE_YEAR = 16,           // Deprecated since Cobalt Strike 4.
↪7  SETTING_BOF_ALLOCATOR = 16,       // Introduced in Cobalt Strike 4.7

SETTING_KILLDATE_MONTH = 17,         // Deprecated since Cobalt Strike 4.
↪8  SETTING_SYSCALL_METHOD = 17,      // Introduced in Cobalt Strike 4.8

SETTING_KILLDATE_DAY = 18,
SETTING_DNS_IDLE = 19,
SETTING_DNS_SLEEP = 20,

// CobaltStrike version >= 3.5 (22 Sept, 2016)
SETTING_SSH_HOST = 21,
SETTING_SSH_PORT = 22,
SETTING_SSH_USERNAME = 23,
SETTING_SSH_PASSWORD = 24,
SETTING_SSH_KEY = 25,
SETTING_C2_VERB_GET = 26,
SETTING_C2_VERB_POST = 27,
SETTING_C2_CHUNK_POST = 28,
SETTING_SPAWNTO_X86 = 29,
SETTING_SPAWNTO_X64 = 30,

// CobaltStrike version >= 3.6 (8 Dec, 2016)
SETTING_CRYPTO_SCHEME = 31,

// CobaltStrike version >= 3.7 (15 Mar, 2016)
SETTING_PROXY_CONFIG = 32,
SETTING_PROXY_USER = 33,
SETTING_PROXY_PASSWORD = 34,
SETTING_PROXY_BEHAVIOR = 35,

// CobaltStrike version >= 3.8 (23 May 2017)
// DEPRECATED_SETTING_INJECT_OPTIONS = 36,

// Renamed from DEPRECATED_SETTING_INJECT_OPTIONS in CobaltStrike 4.5
SETTING_WATERMARKHASH = 36,

// CobaltStrike version >= 3.9 (Sept 26, 2017)
SETTING_WATERMARK = 37,

// CobaltStrike version >= 3.11 (April 9, 2018)
SETTING_CLEANUP = 38,

// CobaltStrike version >= 3.11 (May 24, 2018)
SETTING_CFG_CAUTION = 39,

```

(continues on next page)

(continued from previous page)

```

// CobaltStrike version >= 3.12 (Sept 6, 2018)
SETTING_KILLDATE = 40,
SETTING_GARGLE_NOOK = 41,          // https://www.youtube.com/watch?
↪v=nLTgWdXrx3U
SETTING_GARGLE_SECTIONS = 42,
SETTING_PROCINJ_PERMS_I = 43,
SETTING_PROCINJ_PERMS = 44,
SETTING_PROCINJ_MINALLOC = 45,
SETTING_PROCINJ_TRANSFORM_X86 = 46,
SETTING_PROCINJ_TRANSFORM_X64 = 47,

SETTING_PROCINJ_ALLOWED = 48,      // Deprecated since Cobalt_
↪Strike 4.7
SETTING_PROCINJ_BOF_REUSE_MEM = 48, // Introduced in Cobalt Strike_
↪4.7

// CobaltStrike version >= 3.13 (Jan 2, 2019)
SETTING_BINDHOST = 49,

// CobaltStrike version >= 3.14 (May 4, 2019)
SETTING_HTTP_NO_COOKIES = 50,
SETTING_PROCINJ_EXECUTE = 51,
SETTING_PROCINJ_ALLOCATOR = 52,
SETTING_PROCINJ_STUB = 53,        // .self = MD5(cobaltstrike.jar)

// CobaltStrike version >= 4.0 (Dec 5, 2019)
SETTING_HOST_HEADER = 54,
SETTING_EXIT_FUNK = 55,

// CobaltStrike version >= 4.1 (June 25, 2020)
SETTING_SSH_BANNER = 56,
SETTING_SMB_FRAME_HEADER = 57,
SETTING_TCP_FRAME_HEADER = 58,

// CobaltStrike version >= 4.2 (Nov 6, 2020)
SETTING_HEADERS_REMOVE = 59,

// CobaltStrike version >= 4.3 (Mar 3, 2021)
SETTING_DNS_BEACON_BEACON = 60,
SETTING_DNS_BEACON_GET_A = 61,
SETTING_DNS_BEACON_GET_AAAA = 62,
SETTING_DNS_BEACON_GET_TXT = 63,
SETTING_DNS_BEACON_PUT_METADATA = 64,
SETTING_DNS_BEACON_PUT_OUTPUT = 65,
SETTING_DNSRESOLVER = 66,
SETTING_DOMAIN_STRATEGY = 67,
SETTING_DOMAIN_STRATEGY_SECONDS = 68,
SETTING_DOMAIN_STRATEGY_FAIL_X = 69,
SETTING_DOMAIN_STRATEGY_FAIL_SECONDS = 70,

// CobaltStrike version >= 4.5 (Dec 14, 2021)
SETTING_MAX_RETRY_STRATEGY_ATTEMPTS = 71,

```

(continues on next page)

(continued from previous page)

```

    SETTING_MAX_RETRY_STRATEGY_INCREASE = 72,
    SETTING_MAX_RETRY_STRATEGY_DURATION = 73,

    // CobaltStrike version >= 4.7 (Aug 17, 2022)
    SETTING_MASKED_WATERMARK = 74,
};

enum DeprecatedBeaconSetting: uint16 {
    SETTING_KILLDATE_YEAR = 16,
    SETTING_INJECT_OPTIONS = 36,
};

enum TransformStep: uint32 {
    APPEND = 1,
    PREPEND = 2,
    BASE64 = 3,
    PRINT = 4,
    PARAMETER = 5,
    HEADER = 6,
    BUILD = 7,
    NETBIOS = 8,
    _PARAMETER = 9,
    _HEADER = 10,
    NETBIOSU = 11,
    URI_APPEND = 12,
    BASE64URL = 13,
    STRREP = 14,
    MASK = 15,
    // CobaltStrike version >= 4.0 (Dec 5, 2019)
    _HOSTHEADER = 16,
};

enum SettingsType: uint16 {
    TYPE_NONE = 0,
    TYPE_SHORT = 1,
    TYPE_INT = 2,
    TYPE_PTR = 3,
};

struct Setting {
    BeaconSetting index;    // uint16
    SettingsType type;      // uint16
    uint16 length;         // uint16
    char value[length];
};

flag BeaconProtocol {
    http = 0,
    dns = 1,
    smb = 2,
    tcp = 4,
    https = 8,

```

(continues on next page)

(continued from previous page)

```
    bind = 16
};

flag ProxyServer {
    MANUAL = 0,
    DIRECT = 1,
    PRECONFIG = 2,
    MANUAL_CREDS = 4
};

enum CryptoScheme: uint16 {
    CRYPTO_LICENSED_PRODUCT = 0,
    CRYPTO_TRIAL_PRODUCT = 1
};

enum InjectAllocator: uint8 {
    VirtualAllocEx = 0,
    NtMapViewOfSection = 1,
};

enum InjectExecutor: uint8 {
    CreateThread = 1,
    SetThreadContext = 2,
    CreateRemoteThread = 3,
    RtlCreateUserThread = 4,
    NtQueueApcThread = 5,
    CreateThread_ = 6,
    CreateRemoteThread_ = 7,
    NtQueueApcThread_s = 8
};
""""
```

dissect.cobaltstrike.beacon.cs_struct

dissect.cobaltstrike.beacon.TransformStep

dissect.cobaltstrike.beacon.BeaconSetting

dissect.cobaltstrike.beacon.DeprecatedBeaconSetting

dissect.cobaltstrike.beacon.SettingsType

dissect.cobaltstrike.beacon.Setting

dissect.cobaltstrike.beacon.BeaconProtocol

dissect.cobaltstrike.beacon.CryptoScheme

dissect.cobaltstrike.beacon.ProxyServer

dissect.cobaltstrike.beacon.InjectAllocator

dissect.cobaltstrike.beacon.InjectExecutor

```
dissect.cobaltstrike.beacon.DEFAULT_XOR_KEYS: List[bytes] = [b'i', b'.', b'\x00']
```

Default XOR keys used by Cobalt Strike for obfuscating Beacon config bytes

```
dissect.cobaltstrike.beacon.find_beacon_config_bytes(fh: BinaryIO, xorkey: bytes) → Iterator[bytes]
```

Find and yield (possible) Cobalt Strike configuration bytes from file *fh* using *xorkey* (eg: b"x69").

This is done by scraping the file *fh* for XOR encoded configuration blocks. A beacon configuration block always (unless modified) starts with:

```
Setting(index=SETTING_PROTOCOL, type=TYPE_SHORT, length=0x2)
```

```
# which translates to the following bytes
```

```
b"\x00\x01\x00\x01\x00\x02\x00"
```

These bytes are used in conjunction with the XOR key for finding the (potential) start of a configuration block.

Parameters

- **fh** – file object
- **xorkey** – XOR key (as bytes)

Yields

Beacon configuration bytes (4096 bytes), in deobfuscated (un-XOR'd) form.

```
dissect.cobaltstrike.beacon.iter_beacon_config_blocks(fobj: BinaryIO, xor_keys=None,
                                                       xordecode=True, all_xor_keys=False) →
                                                       Iterator[Tuple[bytes, dict]]
```

Yield tuple with found Beacon *config_block_bytes* from file *fobj* and *extra_info* dict

It always start seeking from the beginning of *fobj*. Side effects: file handle position due to seeking

The *extra_info* dictionary holds some metadata such as if the *fobj* was xorencoded and which xorkey was used.

Parameters

- **xor_keys** – list XOR keys (as bytes), defaults to: [DEFAULT_XOR_KEYS](#) if not specified.
- **xordecode** – If True it will also try to *XorDecode* the file object.
- **all_xor_keys** – Try ALL single-byte XOR keys if no beacon config is found using the default keys.

Yields

Tuple as (config_block_bytes, extra_info_dict) – *extra_info* dict contains:
{"xorkey": bytes, "xorencoded": bool}

```
dissect.cobaltstrike.beacon.make_byte_list(exclude: List[bytes] = None) → List[bytes]
```

Return all single-byte bytes as an ordered list, excluding *exclude* bytes.

```
dissect.cobaltstrike.beacon.iter_settings(fobj: bytes | BinaryIO) → Iterator[Setting]
```

Returns an iterator yielding [Setting](#) objects by reading data from *fobj*

The file position will be at the end of the Beacon config after parsing is done. This can be used to determine the exact size of the Beacon configuration block.

Some edge cases are also handled:

- User-Agent string that exceeds the Setting length.
- Deprecated setting SETTING_INJECT_OPTIONS

Parameters

fobj – bytes or file-like object with Beacon configuration data

Yields

Setting objects

`dissect.cobaltstrike.beacon.grouper(iterable, n, fillvalue=None)`

Collect data into fixed-length chunks or blocks

`dissect.cobaltstrike.beacon.parse_recover_binary(program: bytes) → List[Tuple[str, int | bool]]`

Parse SETTING_C2_RECOVER (*.http-get.server.output*) data

`dissect.cobaltstrike.beacon.parse_transform_binary(program: bytes, build: str = 'metadata') → List[Tuple[str, str | bytes | bool]]`

Parse SETTING_C2_{REQUEST,POSTREQ} (*.http-{get,post}.client*) data

`dissect.cobaltstrike.beacon.parse_execute_list(data: bytes) → List[str]`

Parse SETTING_PROCIJ_EXECUTE (*.process-inject.execute*) data

`dissect.cobaltstrike.beacon.parse_process_injection_transform_steps(data: bytes) → list`

Parse SETTING_PROCIJ_TRANSFORM_X{86,64} (*.process-inject.transform-x{86,64}*) data

`dissect.cobaltstrike.beacon.parse_gargle(data: bytes) → list`

Parse SETTING_GARGLE_SECTIONS (*.stage.{sleep_mask,obfuscate,userwx}*) data

`dissect.cobaltstrike.beacon.parse_pivot_frame(data: bytes) → bytes`

Parse SETTING_{TCP,SMB}_FRAME_HEADER (*.{tcp,smb}_frame_header*) data

`dissect.cobaltstrike.beacon.sha256sum_pubkey(der_data: bytes) → str`

Return the SHA-256 digest of *der_data*

`dissect.cobaltstrike.beacon.null_terminated_bytes(data: bytes) → bytes`

Return null terminated *data* as bytes.

```
>>> null_terminated_bytes(b"Hello World\x00\x00Foobar\x00\x00")
b'Hello World'
>>> null_terminated_bytes(b"foo\xffbar\x00\x00\x00baz\x00")
b'foo\xffbar'
```

`dissect.cobaltstrike.beacon.null_terminated_str(data: bytes) → str`

Return null terminated *data* as string. Non ascii characters are ignored.

```
>>> null_terminated_str(b"Hello World\x00\x00foo bar\x00\x00")
'Hello World'
>>> null_terminated_str(b"Goodbye\xffPlanet\x00\x00")
'GoodbyePlanet'
```

`dissect.cobaltstrike.beacon.SETTING_TO_PRETTYFUNC: Dict[BeaconSetting, Callable]`

BeaconSetting enum to pretty function mapping

class `dissect.cobaltstrike.beacon.BeaconConfig(config_block: bytes)`

A *BeaconConfig* object represents a single Beacon configuration

It holds configuration data, parsed settings and other metadata of a Cobalt Strike Beacon and provides useful methods and properties for accessing the Beacon settings. It does *not* contain the Beacon payload data itself.

It can be directly instantiated using configuration data. Otherwise, use the following constructors:

- `BeaconConfig.from_file()`
- `BeaconConfig.from_path()`
- `BeaconConfig.from_bytes()`

The **from_** constructors automatically tries to extract the configuration data (first candidate only) and also handles *xorencoded* payloads and XOR decoding of obfuscated configuration blocks that is common with Cobalt Strike.

```
property setting_enums: list
```

List of BeaconSetting *enum* values in the order of appearance within the Beacon configuration. Example value:

```
[1, 2, 3, 4, 5, 7, ..., 45, 46, 47, 53, 51, 52]
```

```
property max_setting_enum: int
```

The maximum BeaconSetting *enum* value present in the Beacon configuration.

```
property raw_settings: Mapping[str, Any]
```

Read-only Beacon settings mapping with raw values, indexed by *BeaconSetting* name.

The raw bytes of *TYPE_SHORT* and *TYPE_INT* values are converted to int. Example value:

[illegible]

```
property raw_settings_by_index: Mapping[int, Any]
```

Read-only Beacon settings mapping with raw values, indexed by *BeaconSetting* constant.

The raw bytes of *TYPE_SHORT* and *TYPE_INT* values are converted to int. Example value:

```
mappingproxy({
  1: 8,
  2: 443,
  3: 60000,
  ...
  27: b'POST\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00',
  53: b'\x0c\xe2\xf5TD\xe4y5\x16\xb5\xaf\xe9g\xbe\x92U',
})
```

property settings: Mapping[str, Any]

Read-only Beacon settings mapping with human readable values, indexed by *BeaconSetting* name. Example value:

```
mappingproxy({'SETTING_PROTOCOL': 8,
              'SETTING_PORT': 443,
              'SETTING_SLEEPTIME': 60000,
```

(continues on next page)

(continued from previous page)

```
...
'SETTING_C2_VERB_POST': 'POST',
'SETTING_PROCINJ_STUB': '0ce2f55444e4793516b5afe967be9255',
})
```

property settings_by_index: Mapping[int, Any]

Read-only Beacon settings mapping with human readable values, indexed by *BeaconSetting* constant. Example value:

```
mappingproxy({
  1: 8,
  2: 443,
  3: 60000,
  ...
  27: 'POST',
  53: '0ce2f55444e4793516b5afe967be9255',
})
```

property domain_uri_pairs: List[Tuple[str, str]]

List of configured (*domain*, *uri*) pairs in the Beacon. Example value:

```
[
  ('c1.example.com', '/__utm.gif'),
  ('c2.example.com', '/en_US/all.js'),
]
```

property uris: List[str]

List of configured Beacon URIs. Example value:

```
['/__utm.gif', '/en_US/all.js']
```

property domains: List[str]

List of configured Beacon domains. Example value:

```
['c1.example.com', 'c2.example.com']
```

property submit_uri: str | None

The submit URI that the beacon uses for sending callback data. Example value:

```
['/submit.php']
```

property killdate: str | None

Normalized kill date as YYYY-mm-dd string or None if not defined in Beacon.

Note: The reason why the return type is a `str` instead of a `datetime.date` object is that the configured *killdate* in the Beacon can be arbitrary. e.g. 9999-99-99

property protocol: str | None

The protocol the Beacon uses for communication, e.g. "http", "dns". None if unknown.

property port: int | None

The port the Beacon uses for communication, e.g. 80, 443. None if not defined in config.

property watermark: int | None

Beacon watermark (also known as customer or authorization id).

property is_trial: bool

True if Beacon is a trial version (CRYPTO_TRIAL_PRODUCT). Otherwise, False.

property version: [dissect.cobaltstrike.version.BeaconVersion](#)

Deduced version of Cobalt Strike as [BeaconVersion](#) object.

The version is deduced from the Beacon's [pe_export_stamp](#) when available, otherwise from [max_setting_enum](#).

property public_key: bytes

The RSA public key used by the Beacon in DER format.

property sleeptime: int | None

The sleep time in milliseconds the Beacon uses between communication attempts.

property jitter: int | None

The jitter in milliseconds the Beacon uses between communication attempts.

config_block: bytes

Raw beacon configuration block bytes

settings_tuple

Tuple containing the *Setting* objects parsed from *config_block*

xorkey: bytes | None

XOR key that was used to obfuscate the configuration block, None if unknown.

xorencoded: bool = False

True if the beacon was xorencoded, otherwise False

pe_export_stamp: int | None

PE export timestamp, None if unknown.

pe_compile_stamp: int | None

PE compile timestamp, None if unknown.

architecture: str | None

PE architecture, "x86" or "x64" and None if unknown.

classmethod from_file(*fobj: BinaryIO*, *xor_keys: List[bytes] = None*, *all_xor_keys: bool = False*) → [BeaconConfig](#)

Create a [BeaconConfig](#) from file object, or raises ValueError if no beacon config is found.

Parameters

- **fobj** – file-like object
- **xor_keys** – override the default XOR keys (as bytes) when specified. Default None.
- **all_xor_keys** – if True, it will try ALL single-byte XOR keys if the defaults don't work

Returns

[BeaconConfig](#)

Raises

ValueError – If no valid beacon configuration was found

classmethod **from_path**(*path: str | os.PathLike, xor_keys: List[bytes] = None, all_xor_keys: bool = False*) → *BeaconConfig*

Create a *BeaconConfig* from path, or raises *ValueError* if no beacon config is found.

Parameters

- **path** – path to file on disk
- **xor_keys** – override the default *XOR* keys (as bytes) when specified. Default *None*.
- **all_xor_keys** – if *True* it will try ALL single-byte *XOR* keys if the defaults don't work

Returns

BeaconConfig

Raises

ValueError – If no valid beacon configuration was found

classmethod **from_bytes**(*data: bytes, xor_keys: List[bytes] = None, all_xor_keys: bool = False*) → *BeaconConfig*

Create a *BeaconConfig* from bytes, or raises *ValueError* if no beacon config is found.

Parameters

- **data** – configuration bytes
- **xor_keys** – override the default *XOR* keys when specified. Default *None*.
- **all_xor_keys** – if *True* it will try ALL single-byte *XOR* keys if the defaults don't work

Returns

BeaconConfig

Raises

ValueError – If no valid beacon configuration was found

__repr__() → str

Return repr(self).

settings_map(*index_type='enum', pretty=False, parse=True*) → *types.MappingProxyType*

Return a read-only settings mapping indexed by given *index_type*.

Parameters

- **index_type** – index type of the dictionary, can be one of:
 - *name*: indexed by *BeaconSetting* name (str)
 - *const*: indexed by *BeaconSetting* constant (int)
 - *enum*: indexed by *BeaconSetting* enum (enum object).
- **pretty** – if *True*, apply pretty functions on the values.
- **parse** – if *True*, the raw bytes of *TYPE_SHORT* and *TYPE_INT* values are converted to int.

Returns

OrderedDict

`dissect.cobaltstrike.beacon.build_parser()`

`dissect.cobaltstrike.beacon.main()`

Entrypoint for beacon-dump.

11.1.3 dissect.cobaltstrike.c2

This module is responsible for working with Cobalt Strike C2 traffic.

Module Contents

Classes

<i>EncryptedPacket</i>	Container to hold ciphertext and HMAC signature.
<i>C2Data</i>	Container for holding C2 data that is used for transform and recover steps.
<i>ServerC2Data</i>	Container for holding recovered server-side C2Data.
<i>ClientC2Data</i>	Container for holding recovered client-side C2Data.
<i>HttpRequest</i>	HTTP Request container.
<i>HttpResponse</i>	HTTP Response container.
<i>BeaconKeys</i>	Helper container to hold beacon session keys (AES + HMAC).
<i>HttpDataTransform</i>	Transform and recover Cobalt Strike HTTP C2 data using transformation steps.
<i>C2Http</i>	Class for decrypting and encrypting Cobalt Strike HTTP C2 traffic.

Functions

<i>enable_reprlib_c2()</i>	Enables reprlib <code>__repr__</code> for most of the namedtuple classes in this module.
<i>c2packet_to_record</i> (→ flow.record.Record)	Convert <i>c2packet</i> to a flow.record.
<i>parse_raw_http</i> (→ Union[HttpRequest, HttpResponse])	Parse a raw HTTP request/response bytes and returns a <i>HttpRequest</i> or <i>HttpResponse</i> accordingly.
<i>decrypt_metadata</i> (...)	Decrypt <i>encrypted_metadata</i> using RSA <i>private_key</i> .
<i>encrypt_metadata</i> (→ bytes)	Encrypt <i>metadata</i> using RSA <i>public_key</i> .
<i>derive_aes_hmac_keys</i> (→ Tuple[bytes, bytes])	Derive the AES and HMAC keys from the <i>aes_random</i> bytes.
<i>pad</i> (→ bytes)	Mimics the padding behaviour in Cobalt Strike (which is to fill it with b'A').
<i>encrypt_data</i> (→ bytes)	AES encrypt <i>data</i> with given <i>aes_key</i> and <i>iv</i> .
<i>decrypt_data</i> (→ bytes)	AES decrypt the <i>data</i> with given <i>aes_key</i> and <i>iv</i> and return the decrypted bytes.
<i>decrypt_packet</i> (→ bytes)	Decrypt <i>EncryptedPacket</i> <i>packet</i> and return the decrypted plaintext bytes.
<i>encrypt_packet</i> (→ EncryptedPacket)	Encrypt <i>plaintext</i> bytes and return a <i>EncryptedPacket</i> .

Attributes

<i>TransformStep</i>	Type TransformStep.
<i>C2Packet</i>	Type that is either a BeaconMetadata, a TaskPacket or a CallbackPacket.
<i>logger</i>	

`dissect.cobaltstrike.c2.TransformStep`

Type TransformStep.

`dissect.cobaltstrike.c2.C2Packet`

Type that is either a BeaconMetadata, a TaskPacket or a CallbackPacket.

`dissect.cobaltstrike.c2.logger`

class `dissect.cobaltstrike.c2.EncryptedPacket`

Bases: `NamedTuple`

Container to hold ciphertext and HMAC signature.

ciphertext: `bytes`

signature: `bytes`

dumps()

Return the EncryptedPacket as a bytes object with a size frame header.

| size | ciphertext | signature |

raise_for_signature(*hmac_key: bytes*)

Parameters

hmac_key – HMAC key to use for signature verification

Raises

ValueError – if signature of the ciphertext is incorrect.

class `dissect.cobaltstrike.c2.C2Data`

Bases: `NamedTuple`

Container for holding C2 data that is used for transform and recover steps.

output: `bytes | None`

metadata: `bytes | None`

id: `bytes | None`

class `dissect.cobaltstrike.c2.ServerC2Data`

Bases: `C2Data`

Container for holding recovered server-side C2Data.

iter_encrypted_packets() → `Iterator[EncryptedPacket]`

Iterate over EncryptedPacket, parsed from server-side *c2data.output* data.

For server-side data this is always one packet.

class dissect.cobaltstrike.c2.**ClientC2Data**

Bases: *C2Data*

Container for holding recovered client-side C2Data.

iter_encrypted_packets() → Iterator[*EncryptedPacket*]

Iterate over EncryptedPacket, parsed from client-side *c2data.output* data.

For client-side data this could be one or more packets.

class dissect.cobaltstrike.c2.**HttpRequest**

Bases: NamedTuple

HTTP Request container.

method: bytes

uri: bytes

params: Dict[bytes, bytes]

headers: Dict[bytes, bytes]

body: bytes

class dissect.cobaltstrike.c2.**HttpResponse**

Bases: NamedTuple

HTTP Response container.

status: int

headers: Dict[bytes, bytes]

reason: bytes

body: bytes

request: *HttpRequest* | None

class dissect.cobaltstrike.c2.**BeaconKeys**

Bases: NamedTuple

Helper container to hold beacon session keys (AES + HMAC).

DEFAULT_AES_IV = b'abcdefghijklmnp'

aes_key: bytes | None

hmac_key: bytes | None

iv: bytes

classmethod **from_aes_rand**(*aes_rand*: bytes, *iv*: bytes = *DEFAULT_AES_IV*) → *BeaconKeys*

Create a *BeaconKeys* instance from AES random bytes.

classmethod **from_beacon_metadata**(*metadata*: dissect.cobaltstrike.c2.BeaconMetadata, *iv*: bytes = *DEFAULT_AES_IV*) → *BeaconKeys*

Create a *BeaconKeys* instance from BeaconMetadata.

`dissect.cobaltstrike.c2.enable_reprlib_c2()`

Enables reprlib `__repr__` for most of the namedtuple classes in this module.

`dissect.cobaltstrike.c2.c2packet_to_record(c2packet: C2Packet) → flow.record.Record`

Convert `c2packet` to a `flow.record`.

`dissect.cobaltstrike.c2.parse_raw_http(data: bytes) → HttpRequest | HttpResponse`

Parse a raw HTTP request/response bytes and returns a `HttpRequest` or `HttpResponse` accordingly.

Parameters

data – raw HTTP request or response data bytes.

Returns

Either a `HttpRequest` or `HttpResponse` object based on the data.

Raises

ValueError – if it cannot be parsed as `HttpRequest` or `HttpResponse`.

`class dissect.cobaltstrike.c2.HttpDataTransform(steps: List[TransformStep], reverse: bool = False, build: str = None)`

Transform and recover Cobalt Strike HTTP C2 data using transformation steps.

`transform(c2data: C2Data, request: HttpRequest | None = None) → HttpRequest`

Transform `c2data` information into a `HttpRequest` namedtuple.

Parameters

- **c2data** – `C2Data` named tuple that needs to be transformed
- **request** – Optional initial HTTP request data

Returns

Transformed HTTP request data

Return type

`HttpRequest`

`recover(http: HttpRequest) → ClientC2Data`

`recover(http: HttpResponse) → ServerC2Data`

Recovers the transformed data in `http` object and returns a `C2Data` namedtuple.

Parameters

http – a `HttpRequest` or `HttpResponse` namedtuple

Returns

Either a `ClientC2Data` or `ServerC2Data` namedtuple based on the `http` data.

`class dissect.cobaltstrike.c2.C2Http(bconfig: dissect.cobaltstrike.beacon.BeaconConfig, aes_key: bytes | None = None, hmac_key: bytes | None = None, aes_rand: bytes | None = None, rsa_private_key: Crypto.PublicKey.RSA.RsaKey | None = None, verify_hmac=True)`

Class for decrypting and encrypting Cobalt Strike HTTP C2 traffic.

It requires to be initialized with a `BeaconConfig` and one of the following *key* material:

- *aes_key* and optionally *hmac_key*
- *aes_rand*
- *rsa_private_key* (most preferred when available)

get_transform_for_http(*http*: *HttpRequest* | *HttpResponse* | *bytes*) → *HttpDataTransform*

Return the correct *HttpDataTransform* instance for given *http*.

Parameters

http – either a *HttpRequest* or *HttpResponse* object or raw HTTP bytes.

Returns

The correct *HttpDataTransform* instance for given *http*.

Return type

HttpDataTransform

Raises

ValueError – if no correct transform can be found for given *http* object.

iter_recover_http(*http*: *bytes* | *HttpRequest* | *HttpResponse*, *keys*: *BeaconKeys* | *None* = *None*) → *Iterator*[*C2Packet*]

Yield decrypted *C2Packet* objects from given *http* object.

You can pass your own set of *BeaconKeys* *keys* to use for decryption instead of the default initialized ones. This can be useful if you are processing multiple Beacon sessions and do some sort of session tracking outside this class.

Parameters

- **http** – A *HttpRequest* or *HttpResponse* object, or raw HTTP request or response bytes.
- **keys** – Optional *BeaconKeys* to use for decryption instead of current default keys.

Yields

C2Packet – A *C2Packet* object for each decrypted packet found in the HTTP request or response.

`dissect.cobaltstrike.c2.decrypt_metadata`(*encrypted_metadata*: *bytes*, *private_key*: *Crypto.PublicKey.RSA.RsaKey*) → *dissect.cobaltstrike.c_c2.BeaconMetadata*

Decrypt *encrypted_metadata* using RSA *private_key*.

Parameters

- **encrypted_metadata** – the encrypted metadata bytes
- **private_key** – the RSA private key used for decryption

Returns

The decrypted metadata.

Return type

BeaconMetadata

Raises

ValueError – if RSA failed to decrypt or metadata magic is invalid

`dissect.cobaltstrike.c2.encrypt_metadata`(*metadata*: *dissect.cobaltstrike.c_c2.BeaconMetadata*, *public_key*: *Crypto.PublicKey.RSA.RsaKey*) → *bytes*

Encrypt *metadata* using RSA *public_key*.

Parameters

- **metadata** – *BeaconMetadata* object to encrypt
- **public_key** – the RSA public key used for encryption

Returns

The encrypted metadata as bytes

`dissect.cobaltstrike.c2.derive_aes_hmac_keys(aes_random: bytes) → Tuple[bytes, bytes]`

Derive the AES and HMAC keys from the *aes_random* bytes.

Parameters

aes_random – the bytes to derive the keys from

Returns

Tuple of (aes_key, hmac_key)

`dissect.cobaltstrike.c2.pad(data: bytes, block_size: int = AES.block_size) → bytes`

Mimics the padding behaviour in Cobalt Strike (which is to fill it with b'A').

Parameters

- **data** – the data to pad
- **block_size** – the block size to use for padding

Returns

The padded data

`dissect.cobaltstrike.c2.encrypt_data(data: bytes, aes_key: bytes, iv: bytes) → bytes`

AES encrypt *data* with given *aes_key* and *iv*.

Parameters

- **data** – the data to encrypt
- **aes_key** – the AES key to use
- **iv** – the initialization vector to use

Returns

The encrypted data as bytes

`dissect.cobaltstrike.c2.decrypt_data(data: bytes, aes_key: bytes, iv: bytes) → bytes`

AES decrypt the *data* with given *aes_key* and *iv* and return the decrypted bytes.

Parameters

- **data** – the encrypted data
- **aes_key** – the AES key to use for decryption
- **iv** – the AES IV to use for decryption

Returns

The decrypted data as bytes

`dissect.cobaltstrike.c2.decrypt_packet(packet: EncryptedPacket, aes_key: bytes, hmac_key: bytes | None = None, iv: bytes = BeaconKeys.DEFAULT_AES_IV, verify: bool = True) → bytes`

Decrypt [EncryptedPacket](#) *packet* and return the decrypted plaintext bytes.

If *hmac_key* is defined, the signature of the ciphertext is verified first before decrypting.

Parameters

- **packet** – the [EncryptedPacket](#) to decrypt
- **aes_key** – the AES key to use for decryption
- **hmac_key** – the HMAC key to use for signature verification

- **iv** – the AES IV to use for decryption
- **verify** – whether to verify the HMAC signature of the ciphertext

Returns

The decrypted plaintext bytes

`dissect.cobaltstrike.c2.encrypt_packet` (*plaintext: bytes, aes_key: bytes, hmac_key: bytes, iv: bytes = BeaconKeys.DEFAULT_AES_IV*) → *EncryptedPacket*

Encrypt *plaintext* bytes and return a *EncryptedPacket*.

Parameters

- **plaintext** – the plaintext bytes to encrypt
- **aes_key** – the AES key to use for encryption
- **hmac_key** – the HMAC key to use for signature generation
- **iv** – the AES IV to use for encryption

Returns

The *EncryptedPacket* containing the ciphertext and HMAC signature

11.1.4 dissect.cobaltstrike.c2profile

This module is responsible for parsing and generating Cobalt Strike Malleable C2 profiles. It uses the *lark-parser* library for parsing the syntax using the `c2profile.lark` grammar file.

Module Contents

Classes

<i>StringIterator</i>	Helper class for iterating over characters in a string
<i>ConfigBlock</i>	Base class for configuration blocks
<i>HttpOptionsBlock</i>	<i>.http-{stager,get,post}.{client,server}</i> block
<i>DataTransformBlock</i>	<i>data_transform</i> block
<i>HttpStagerBlock</i>	<i>.http-stager</i> block
<i>HttpConfigBlock</i>	<i>.http-config</i> block
<i>StageBlock</i>	<i>.stage</i> block
<i>StageTransformBlock</i>	<i>.stage.transform-x86</i> and <i>.stage.transform-x64</i> block
<i>ProcessInjectBlock</i>	<i>.process-inject</i> block
<i>HttpGetBlock</i>	<i>.http-get</i> block
<i>HttpPostBlock</i>	<i>.http-post</i> block
<i>PostExBlock</i>	<i>.post-ex</i> block
<i>DnsBeaconBlock</i>	<i>.dns-beacon</i> block
<i>ExecuteOptionsBlock</i>	<i>.process-inject.execute</i> block
<i>C2Profile</i>	A <i>C2Profile</i> object represents a parsed Malleable C2 Profile

Functions

<code>value_to_string(→ str)</code>	Converts value to it's STRING Token value
<code>string_token_to_bytes(→ Union[lark.Token, bytes])</code>	Convert a STRING Token value to it's native Python bytes value.
<code>build_parser()</code>	
<code>main()</code>	Entrypoint for c2profile-dump.

Attributes

<code>logger</code>
<code>c2profile_parser</code>

`dissect.cobaltstrike.c2profile.logger`

`dissect.cobaltstrike.c2profile.c2profile_parser`

`dissect.cobaltstrike.c2profile.value_to_string(value: str | bytes) → str`

Converts value to it's STRING Token value

`dissect.cobaltstrike.c2profile.string_token_to_bytes(token: lark.Token) → lark.Token | bytes`

Convert a STRING Token value to it's native Python bytes value.

If the input is not of Token.type STRING it will return the original Token.

class `dissect.cobaltstrike.c2profile.StringIterator(string: str)`

Helper class for iterating over characters in a string

has_next(count: int = 1) → bool

next(count: int) → List[str]

__iter__()

__next__()

class `dissect.cobaltstrike.c2profile.ConfigBlock(**kwargs)`

Base class for configuration blocks

__name__ = 'ConfigBlock'

init_kwargs(**kwargs)

set_config_block(option, config_block)

set_non_empty_config_block(option, config_block)

set_option(option, value)

_pair(option, value)

```

    _enable(option, value)

    _header(option, value)

    _parameter(option, value)

class dissect.cobaltstrike.c2profile.HttpOptionsBlock(**kwargs)
    Bases: ConfigBlock
    .http-{stager,get,post}.{client,server} block
    __name__ = 'http_options'

    header

    parameter

class dissect.cobaltstrike.c2profile.DataTransformBlock(steps=None)
    Bases: ConfigBlock
    data_transform block
    property tree
    __name__ = 'DataTransformBlock'

    add_step(option, value)

    add_termination(option, value)

class dissect.cobaltstrike.c2profile.HttpStagerBlock(**kwargs)
    Bases: ConfigBlock
    .http-stager block
    __name__ = 'http_stager'

class dissect.cobaltstrike.c2profile.HttpConfigBlock(**kwargs)
    Bases: ConfigBlock
    .http-config block
    __name__ = 'http_config'

    header

class dissect.cobaltstrike.c2profile.StageBlock(**kwargs)
    Bases: ConfigBlock
    .stage block
    __name__ = 'stage'

class dissect.cobaltstrike.c2profile.StageTransformBlock(**kwargs)
    Bases: ConfigBlock
    .stage.transform-x86 and .stage.transform-x64 block
    __name__ = 'StageTransformBlock'

    strep

```

```
class dissect.cobaltstrike.c2profile.ProcessInjectBlock(**kwargs)
    Bases: ConfigBlock
    .process-inject block
    __name__ = 'process_inject'

class dissect.cobaltstrike.c2profile.HttpGetBlock(**kwargs)
    Bases: ConfigBlock
    .http-get block
    __name__ = 'http_get'

class dissect.cobaltstrike.c2profile.HttpPostBlock(**kwargs)
    Bases: ConfigBlock
    .http-post block
    __name__ = 'http_post'

class dissect.cobaltstrike.c2profile.PostExBlock(**kwargs)
    Bases: ConfigBlock
    .post-ex block
    __name__ = 'post_ex'

class dissect.cobaltstrike.c2profile.DnsBeaconBlock(**kwargs)
    Bases: ConfigBlock
    .dns-beacon block
    __name__ = 'dns_beacon'

class dissect.cobaltstrike.c2profile.ExecuteOptionsBlock(**kwargs)
    Bases: ConfigBlock
    .process-inject.execute block
    __name__ = 'ExecuteOptionsBlock'

    createthread_special
    createremotethread_special
    createthread
    createremotethread
    ntqueueapcthread
    ntqueueapcthread_s
    rtlcreateuserthread
    setthreadcontext

    classmethod from_execute_list(execute_list=None)
```

class `dissect.cobaltstrike.c2profile.C2Profile(**kwargs)`
 Bases: [*ConfigBlock*](#)
 A [*C2Profile*](#) object represents a parsed Malleable C2 Profile
 Besides loading C2 Profiles, it also provides methods for building a C2 Profile from scratch.

property properties
 C2 Profile settings as dictionary, alias for [*as_dict\(\)*](#)

__name__ = 'start'

set_option(option, value)
 Sets a global option in the AST tree. E.g: `set_option("jitter", "6000")`

classmethod from_path(path: str | os.PathLike) → C2Profile
 Construct a [*C2Profile*](#) from given path (path to a malleable C2 profile)

classmethod from_text(source: str) → C2Profile
 Construct a [*C2Profile*](#) from text (malleable C2 profile syntax)

classmethod from_beacon_config(config: dissect.cobaltstrike.beacon.BeaconConfig) → C2Profile
 Construct a [*C2Profile*](#) from a [*BeaconConfig*](#)

__str__() → str
 Return str(self).

as_text() → str
 Return the C2 Profile settings as text (malleable C2 profile syntax).

as_dict() → dict
 Return the C2 Profile settings as a dictionary

`dissect.cobaltstrike.c2profile.build_parser()`

`dissect.cobaltstrike.c2profile.main()`
 Entrypoint for c2profile-dump.

11.1.5 dissect.cobaltstrike.c_c2

Structure definitions and classes for dealing with Cobalt Strike C2 packets. Mainly used by [*dissect.cobaltstrike.c2*](#).

Module Contents

Classes

<i>BeaconCommand</i>	Enum where members are also (and must be) ints
<i>BeaconCallback</i>	Enum where members are also (and must be) ints
<i>BeaconMetadata</i>	Holds parsed structure data.
<i>CallbackPacket</i>	Holds parsed structure data.
<i>TaskPacket</i>	Holds parsed structure data.

Functions

`typedef_for_enum(→ str)`Return C compatible typedef string for *enum_class*.

Attributes

`C2_DEF``c2struct`

class dissect.cobaltstrike.c_c2.BeaconCommand

Bases: enum.IntEnum

Enum where members are also (and must be) ints

COMMAND_SPAWN = 1**COMMAND_SHELL** = 2**COMMAND_DIE** = 3**COMMAND_SLEEP** = 4**COMMAND_CD** = 5**COMMAND_KEYLOG_START** = 6**COMMAND_NOOP** = 6**COMMAND_KEYLOG_STOP** = 7**COMMAND_CHECKIN** = 8**COMMAND_INJECT_PID** = 9**COMMAND_UPLOAD** = 10**COMMAND_DOWNLOAD** = 11**COMMAND_EXECUTE** = 12**COMMAND_SPAWN_PROC_X86** = 13**COMMAND_CONNECT** = 14**COMMAND_SEND** = 15**COMMAND_CLOSE** = 16**COMMAND_LISTEN** = 17**COMMAND_INJECT_PING** = 18

COMMAND_CANCEL_DOWNLOAD = 19
COMMAND_PIPE_ROUTE = 22
COMMAND_PIPE_CLOSE = 23
COMMAND_PIPE_REOPEN = 24
COMMAND_TOKEN_GETUID = 27
COMMAND_TOKEN_REV2SELF = 28
COMMAND_TIMESTAMP = 29
COMMAND_STEAL_TOKEN = 31
COMMAND_PS_LIST = 32
COMMAND_PS_KILL = 33
COMMAND_PSH_IMPORT = 37
COMMAND_RUNAS = 38
COMMAND_PWD = 39
COMMAND_JOB_REGISTER = 40
COMMAND_JOBS = 41
COMMAND_JOB_KILL = 42
COMMAND_INJECTX64_PID = 43
COMMAND_SPAWNX64 = 44
COMMAND_INJECT_PID_PING = 45
COMMAND_INJECTX64_PID_PING = 46
COMMAND_PAUSE = 47
COMMAND_LOGINUSER = 49
COMMAND_LSOCKET_BIND = 50
COMMAND_LSOCKET_CLOSE = 51
COMMAND_STAGE_PAYLOAD = 52
COMMAND_FILE_LIST = 53
COMMAND_FILE_MKDIR = 54
COMMAND_FILE_DRIVES = 55
COMMAND_FILE_RM = 56
COMMAND_STAGE_PAYLOAD_SMB = 57
COMMAND_WEBSERVER_LOCAL = 59

```
COMMAND_ELEVATE_PRE = 60
COMMAND_ELEVATE_POST = 61
COMMAND_JOB_REGISTER_IMPERSONATE = 62
COMMAND_SPAWN_POWERSHELLX86 = 63
COMMAND_SPAWN_POWERSHELLX64 = 64
COMMAND_INJECT_POWERSHELLX86_PID = 65
COMMAND_INJECT_POWERSHELLX64_PID = 66
COMMAND_UPLOAD_CONTINUE = 67
COMMAND_PIPE_OPEN_EXPLICIT = 68
COMMAND_SPAWN_PROC_X64 = 69
COMMAND_JOB_SPAWN_X86 = 70
COMMAND_JOB_SPAWN_X64 = 71
COMMAND_SETENV = 72
COMMAND_FILE_COPY = 73
COMMAND_FILE_MOVE = 74
COMMAND_PPID = 75
COMMAND_RUN_UNDER_PID = 76
COMMAND_GETPRIVS = 77
COMMAND_EXECUTE_JOB = 78
COMMAND_PSH_HOST_TCP = 79
COMMAND_DLL_LOAD = 80
COMMAND_REG_QUERY = 81
COMMAND_LSOCKET_TCPPIVOT = 82
COMMAND_ARGUE_ADD = 83
COMMAND_ARGUE_REMOVE = 84
COMMAND_ARGUE_LIST = 85
COMMAND_TCP_CONNECT = 86
COMMAND_JOB_SPAWN_TOKEN_X86 = 87
COMMAND_JOB_SPAWN_TOKEN_X64 = 88
COMMAND_SPAWN_TOKEN_X86 = 89
COMMAND_SPAWN_TOKEN_X64 = 90
```



```
COMMAND_INJECTX64_PING = 91
COMMAND_BLOCKDLLS = 92
COMMAND_SPAWNAS_X86 = 93
COMMAND_SPAWNAS_X64 = 94
COMMAND_INLINE_EXECUTE = 95
COMMAND_RUN_INJECT_X86 = 96
COMMAND_RUN_INJECT_X64 = 97
COMMAND_SPAWNU_X86 = 98
COMMAND_SPAWNU_X64 = 99
COMMAND_INLINE_EXECUTE_OBJECT = 100
COMMAND_JOB_REGISTER_MSGMODE = 101
COMMAND_LSOCKET_BIND_LOCALHOST = 102
```

```
class dissect.cobaltstrike.c_c2.BeaconCallback
```

```
    Bases: enum.IntEnum
```

```
    Enum where members are also (and must be) ints
```

```
    CALLBACK_OUTPUT = 0
    CALLBACK_KEYSTROKES = 1
    CALLBACK_FILE = 2
    CALLBACK_SCREENSHOT = 3
    CALLBACK_CLOSE = 4
    CALLBACK_READ = 5
    CALLBACK_CONNECT = 6
    CALLBACK_PING = 7
    CALLBACK_FILE_WRITE = 8
    CALLBACK_FILE_CLOSE = 9
    CALLBACK_PIPE_OPEN = 10
    CALLBACK_PIPE_CLOSE = 11
    CALLBACK_PIPE_READ = 12
    CALLBACK_POST_ERROR = 13
    CALLBACK_PIPE_PING = 14
    CALLBACK_TOKEN_STOLEN = 15
```

```
CALLBACK_TOKEN_GETUID = 16
CALLBACK_PROCESS_LIST = 17
CALLBACK_POST_REPLAY_ERROR = 18
CALLBACK_PWD = 19
CALLBACK_JOBS = 20
CALLBACK_HASHDUMP = 21
CALLBACK_PENDING = 22
CALLBACK_ACCEPT = 23
CALLBACK_NETVIEW = 24
CALLBACK_PORTSCAN = 25
CALLBACK_DEAD = 26
CALLBACK_SSH_STATUS = 27
CALLBACK_CHUNK_ALLOCATE = 28
CALLBACK_CHUNK_SEND = 29
CALLBACK_OUTPUT_OEM = 30
CALLBACK_ERROR = 31
CALLBACK_OUTPUT_UTF8 = 32
```

```
dissect.cobaltstrike.c_c2.C2_DEF = Multiline-String
```

```
"""
// Callback data from: Beacon -> Team Server
typedef struct CallbackPacket {
    uint32 counter;
    uint32 size;
    BeaconCallback callback;
    char data[size];
};

// Task from: Team Server -> Beacon
typedef struct TaskPacket {
    uint32 epoch;
    uint32 total_size;
    BeaconCommand command;
    uint32 size;
    char data[size];
};

struct BeaconMetadata {
    uint32 magic;
    uint32 size;
    char aes_rand[16];
}
```

(continues on next page)

(continued from previous page)

```

uint16 ansi_cp;      // GetACP
uint16 oem_cp;       // GetOEMCP
uint32 bid;
uint32 pid;
uint16 port;
uint8 flag;
uint8 ver_major;
uint8 ver_minor;
uint16 ver_build;
uint32 ptr_x64;      // for x64 addressing
uint32 ptr_gmh;      // GetModuleHandle
uint32 ptr_gpa;      // GetProcAddress
uint32 ip;
char info[size - 51];
};
"""

```

`dissect.cobaltstrike.c_c2.c2struct`

`dissect.cobaltstrike.c_c2.typedef_for_enum`(*enum_class*: *enum.IntEnum*, *int_type*: *str* = 'uint32') → *str*
 Return C compatible typedef string for *enum_class*.

class `dissect.cobaltstrike.c_c2.BeaconMetadata`(*args, **kwargs)

Bases: `dissect.cstruct.Instance`

Holds parsed structure data.

magic: `int`

size: `int`

aes_rand: `bytes`

ansi_cp: `int`

oem_cp: `int`

bid: `int`

pid: `int`

port: `int`

flag: `int`

ver_major: `int`

ver_minor: `int`

ver_build: `int`

ptr_x64: `int`

ptr_gmh: `int`

ptr_gpa: `int`

```
    ip: int
    info: bytes
    __eq__(other)
        Return self==value.
    __hash__()
        Return hash(self).
class dissect.cobaltstrike.c_c2.CallbackPacket(*args, **kwargs)
    Bases: dissect.cstruct.Instance
    Holds parsed structure data.
    counter: int
    size: int
    callback: BeaconCallback
    data: bytes
    __eq__(other)
        Return self==value.
    __hash__()
        Return hash(self).
class dissect.cobaltstrike.c_c2.TaskPacket(*args, **kwargs)
    Bases: dissect.cstruct.Instance
    Holds parsed structure data.
    epoch: int
    total_size: int
    command: BeaconCommand
    size: int
    data: bytes
    __eq__(other)
        Return self==value.
    __hash__()
        Return hash(self).
```

11.1.6 dissect.cobaltstrike.client

Beacon client that can actively connect to a Cobalt Strike Team Server.

Danger: The client actively connects to a Cobalt Strike Team Server, caution should be taken when using this. A default client will perform check-ins and only log the tasks it receives unless implemented otherwise.

Module Contents

Classes

<i>HttpBeaconClient</i>	A Beacon Client that can communicate with a Cobalt Strike Team Server over HTTP.
-------------------------	--

Functions

<i>random_computer_name</i> (→ str)	Returns a random Windows like computer name, if <i>username</i> is set it can also return <USERNAME>-PC
<i>random_username_name</i> (→ str)	Returns a random username in the form of john.smith or John Smith.
<i>random_windows_ver</i> (→ Tuple[int, int, int])	Return a random Windows version in the form of the tuple (major, minor, build).
<i>random_process_name</i> (→ str)	Return a random process name.
<i>random_internal_ip</i> (→ ipaddress.IPv4Address)	Return a random internal RFC1918 IP address.
<i>log_task</i> (task)	
<i>CallbackError</i> (→ Tuple[int, bytes])	
<i>CallbackDebugMessage</i> (→ Tuple[int, bytes])	This will output '[-] DEBUG: <message>' to the Team Server console.
<i>CallbackOutputMessage</i> (→ Tuple[int, bytes])	This will output '[+] received output: <message>' to the Team Server console.
<i>build_parser</i> (→ argparse.ArgumentParser)	Return the default ArgumentParser for the beacon client.
<i>parse_commandline_options</i> (→ Tuple[argparse.Namespace, ...])	Helper function to parse commandline options and return a tuple of (args, options).
<i>main</i> ()	

Attributes

<i>logger</i>
<i>FIRST_NAMES</i>
<i>LAST_NAMES</i>
<i>PROCESS_NAMES</i>
<i>COMPUTERNAME_TEMPLATES</i>

dissect.cobaltstrike.client.logger

```
dissect.cobaltstrike.client.FIRST_NAMES = ['Michael', 'James', 'John', 'Robert', 'David',  
'William', 'Mary', 'Christopher', 'Joseph',...]
```

```
dissect.cobaltstrike.client.LAST_NAMES = ['SMITH', 'JOHNSON', 'WILLIAMS', 'BROWN',  
'JONES', 'GARCIA', 'RODRIGUEZ', 'MILLER', 'MARTINEZ',...]
```

```
dissect.cobaltstrike.client.PROCESS_NAMES = ['rundll32.exe', 'dllhost.exe',  
'gpupdate.exe', 'svchost.exe', 'mstsc.exe', 'WerFault.exe',...]
```

```
dissect.cobaltstrike.client.COMPUTERNAME_TEMPLATES
```

```
dissect.cobaltstrike.client.random_computer_name(username: str | None = None) → str  
Returns a random Windows like computer name, if username is set it can also return <USERNAME>-PC
```

```
dissect.cobaltstrike.client.random_username_name() → str  
Returns a random username in the form of john.smith or John Smith.
```

```
dissect.cobaltstrike.client.random_windows_ver() → Tuple[int, int, int]  
Return a random Windows version in the form of the tuple (major, minor, build).
```

```
dissect.cobaltstrike.client.random_process_name() → str  
Return a random process name.
```

```
dissect.cobaltstrike.client.random_internal_ip() → ipaddress.IPv4Address  
Return a random internal RFC1918 IP address.
```

```
dissect.cobaltstrike.client.log_task(task)
```

```
dissect.cobaltstrike.client.CallbackError(code: int, n1: int, n2: int, message: str) → Tuple[int, bytes]
```

```
dissect.cobaltstrike.client.CallbackDebugMessage(message: str) → Tuple[int, bytes]  
This will output '[-] DEBUG: <message>' to the Team Server console.
```

```
dissect.cobaltstrike.client.CallbackOutputMessage(message: str) → Tuple[int, bytes]  
This will output '[+] received output: <message>' to the Team Server console.
```

```
class dissect.cobaltstrike.client.HttpBeaconClient
```

A Beacon Client that can communicate with a Cobalt Strike Team Server over HTTP.

```
run(bconfig: dissect.cobaltstrike.c2.BeaconConfig, dry_run=False, scheme=None, domain=None,  
    port=None, beacon_id=None, pid=None, computer=None, user=None, process=None,  
    internal_ip=None, arch=None, barch=None, ansi_cp=58372, oem_cp=46337, high_integrity=False,  
    sleeptime=None, jitter=None, user_agent=None, host_header=None, verbose=None, silent=None,  
    writer=None)
```

Run the Beacon Client.

```
_initial_get_request() → dissect.cobaltstrike.c2.HttpRequest  
Return the initial HttpRequest object for retrieving tasks from the Team Server.
```

```
_initial_post_request() → dissect.cobaltstrike.c2.HttpRequest  
Return the initial HttpRequest object for sending callback data to the Team Server.
```

```
get_sleep_time() → float  
Return the sleep time with jitter for the beacon loop.
```

register_task(*command_id: None | int, func*)

Register a task handler for a given command ID.

Parameters

- **command_id** – The command ID to register the handler for. *None* is handler for empty tasks. -1 is a catch-all handler.
- **func** – The function to call when a task with the given command ID is received.

get_task() → *dissect.cobaltstrike.c2.TaskPacket | None*

Get a task from the Team Server.

send_callback(*callback_id: int, data: bytes*)

Send callback data to the Team Server.

handle(*command: None | int | dissect.cobaltstrike.c2.BeaconCommand*)

decorator to register a handler for *command*, if *None* it registers a handler for empty tasks

catch_all()

decorator to handle all *unhandled* commands.

print_settings()

get_handlers(*command_id: int | None*) → *List[Callable]*

Get a list of handlers for a given command ID.

_beacon_loop()

dissect.cobaltstrike.client.**build_parser**() → *argparse.ArgumentParser*

Return the default ArgumentParser for the beacon client.

dissect.cobaltstrike.client.**parse_commandline_options**(*parser=None, defaults=None*) →
Tuple[argparse.Namespace, Dict[str, Any]]

Helper function to parse commandline options and return a tuple of (args, options).

This method is useful for creating default commandline options for a Beacon client. The returned options can be passed to *HttpBeaconClient.run()* as follows:

```
from dissect.cobaltstrike.client import HttpBeaconClient, parse_commandline_options

beacon = HttpBeaconClient()

args, options = parse_commandline_options(defaults={
    "beacon_id": 1234,
    "computer": "dissect",
    "user": "cobaltstrike",
    "process": "calc.exe",
})

beacon.run(**options)
```

If *parser* is not defined it will use the default argparse parser created by *build_parser()*. The *defaults* dictionary can be used to override the default argparse settings.

Parameters

- **parser** – an instance of *argparse.ArgumentParser*, if *None* it will use the parser created by *client.build_parser()*.

- **defaults** – A dictionary to override the default settings for the argument parser. Unknown keys will be ignored.

Returns

Tuple of (args, options) where *args* is the parsed arguments from the commandline and *options* is a dictionary of options that can be passed to [HttpBeaconClient.run\(\)](#).

`dissect.cobaltstrike.client.main()`

11.1.7 dissect.cobaltstrike.pcap

Module Contents

Classes

<i>BeaconCapture</i>	A class representing a beacon capture file.
--------------------------------------	---

Functions

<i>packet_to_record</i> (→ flow.record.Record)	Convert pcap <i>packet</i> to a flow.record.
<i>c2packet_to_record</i> (→ flow.record.Record)	Convert <i>c2packet</i> to a flow.record.
<i>raw_http_from_packet</i> (→ bytes)	Return the extracted raw HTTP bytes from <i>packet</i> .
<i>main</i> ()	

Attributes

<i>logger</i>	
<i>PacketRecord</i>	Record Descriptor for basic PCAP packet information

`dissect.cobaltstrike.pcap.logger`

`dissect.cobaltstrike.pcap.PacketRecord`

Record Descriptor for basic PCAP packet information

`dissect.cobaltstrike.pcap.packet_to_record(packet: pyshark.packet.packet.Packet) → flow.record.Record`

Convert pcap *packet* to a flow.record.

`dissect.cobaltstrike.pcap.c2packet_to_record(c2packet: dissect.cobaltstrike.c2.C2Packet) → flow.record.Record`

Convert *c2packet* to a flow.record.

`dissect.cobaltstrike.pcap.raw_http_from_packet(packet: pyshark.packet.packet.Packet) → bytes`

Return the extracted raw HTTP bytes from *packet*.


```
class dissect.cobaltstrike.pcap.BeaconCapture(pcap: str, bconfig:
    dissect.cobaltstrike.beacon.BeaconConfig | None =
    None, aes_key: bytes | None = None, hmac_key: bytes |
    None = None, rsa_private_key:
    Crypto.PublicKey.RSA.RsaKey | None = None,
    verify_hmac: bool = True, all_metadata: bool = False,
    extract_beacons: bool = False)
```

A class representing a beacon capture file.

Parameters

- **pcap** – A PCAP file containing Cobalt Strike traffic
- **nss** – NSSKEYLOGFILE containing the client random and masterkey in NSS format
- **aes_key** – AES key used in the beacon session
- **hmac_key** – hmac key used in the beacon session (optional)
- **c2** – IP address of the Cobalt Strike C2 server
- **config** – A Cobalt Strike *BeaconConfig* configuration
- **filter** – A Wireshark display filter used for filtering the pcap

__iter__() → Iterator[Tuple[pyshark.packet.packet.Packet, dissect.cobaltstrike.c2.C2Packet]]

Alias for *BeaconCapture.iter_parse_pcap()*.

iter_parse_pcap(pcap: str, all_metadata: bool | None = None, nss_keylog_file: str | None = None, c2_ip: str | None = None, display_filter: str = 'http', extract_beacons: bool = False) → Iterator[Tuple[pyshark.packet.packet.Packet, dissect.cobaltstrike.c2.C2Packet]]

Yields (packet, c2packet) for every decrypted http C2 packet in the PCAP.

Parameters

- **pcap** – path to PCAP file
- **all_metadata** – If True it will yield all decrypted BeaconMetadata. Otherwise, yield only the metadata that has not been seen yet. Useful if you want to ignore subsequent check-ins.
- **nss_keylog_file** – path to a SSLKEY_LOG file for decrypting TLS traffic in the pcap.
- **c2_ip** – IP address of the C2, if defined it will be used to filter packets and speed up processing.
- **display_filter** – A wireshark display filter to apply to the pcap. It's recommended to use at least `http` (default).

Yields

Tuple of (packet, c2packet)

find_staged_beacon(response: dissect.cobaltstrike.c2.HttpResponse) → dissect.cobaltstrike.beacon.BeaconConfig | None

Returns a *BeaconConfig* if found in the HTTP response body. If the response has an associated request it will check if the request is a stager uri first.

Parameters

response – The HttpResponse object to check for Stager URI and Beacon payload.

Returns

The beacon config if found, otherwise *None*.

Return type*BeaconConfig*

dissect.cobaltstrike.pcap.main()

11.1.8 dissect.cobaltstrike.pe

This module contains helper functions for parsing PE files, mainly for extracting Beacon specific PE artifacts.

Module Contents

Functions

<i>find_mz_offset</i> (→ Optional[int])	Find and return the start offset of a valid IMAGE_DOS_HEADER or None if it cannot be found.
<i>find_compile_stamps</i> (→ Tuple[Optional[int], Optional[int]])	Find and return a tuple with the <i>PE compile</i> and <i>PE export</i> timestamps.
<i>find_magic_mz</i> (→ Optional[bytes])	Find and returns the MZ header bytes or None if cannot be found
<i>find_magic_pe</i> (→ Optional[bytes])	Find and returns the PE header (magic_pe) bytes or None if cannot be found
<i>find_stage_prepend_append</i> (→ Tuple[Optional[bytes], ...])	Find and return the stage prepend and append bytes as a tuple.
<i>find_architecture</i> (→ Optional[str])	Find and return the PE image architecture, either "x86" or "x64" or None if not found.

Attributes

<i>logger</i>
<i>PE_DEF</i>
<i>pestruct</i>
<i>DOSHEADER_X64</i>
<i>DOSHEADER_X86</i>

dissect.cobaltstrike.pe.logger

dissect.cobaltstrike.pe.PE_DEF = Multiline-String

```
"""
#define IMAGE_FILE_MACHINE_AMD64    0x8664
#define IMAGE_FILE_MACHINE_I386     0x014c
#define IMAGE_FILE_MACHINE_IA64     0x0200
```

(continues on next page)

(continued from previous page)

```

#define IMAGE_NUMBEROF_DIRECTORY_ENTRIES 16
#define IMAGE_SIZEOF_SHORT_NAME 8

#define IMAGE_DIRECTORY_ENTRY_EXPORT 0
#define IMAGE_DIRECTORY_ENTRY_IMPORT 1
#define IMAGE_DIRECTORY_ENTRY_RESOURCE 2

typedef struct _IMAGE_DOS_HEADER
{
    WORD e_magic;
    WORD e_cblp;
    WORD e_cp;
    WORD e_crlc;
    WORD e_cparhdr;
    WORD e_minalloc;
    WORD e_maxalloc;
    WORD e_ss;
    WORD e_sp;
    WORD e_csum;
    WORD e_ip;
    WORD e_cs;
    WORD e_lfarlc;
    WORD e_ovno;
    WORD e_res[4];
    WORD e_oemid;
    WORD e_oeminfo;
    WORD e_res2[10];
    LONG e_lfanew;
} IMAGE_DOS_HEADER;

typedef struct _IMAGE_FILE_HEADER {
    WORD Machine;
    WORD NumberOfSections;
    DWORD TimeDateStamp;
    DWORD PointerToSymbolTable;
    DWORD NumberOfSymbols;
    WORD SizeOfOptionalHeader;
    WORD Characteristics;
} IMAGE_FILE_HEADER;

typedef struct _IMAGE_DATA_DIRECTORY {
    ULONG VirtualAddress;
    ULONG Size;
} IMAGE_DATA_DIRECTORY;

typedef struct _IMAGE_OPTIONAL_HEADER {
    WORD Magic;
    BYTE MajorLinkerVersion;
    BYTE MinorLinkerVersion;
    DWORD SizeOfCode;
    DWORD SizeOfInitializedData;
    DWORD SizeOfUninitializedData;

```

(continues on next page)

(continued from previous page)

```

    DWORD    AddressOfEntryPoint;
    DWORD    BaseOfCode;
    DWORD    BaseOfData;
    DWORD    ImageBase;
    DWORD    SectionAlignment;
    DWORD    FileAlignment;
    WORD     MajorOperatingSystemVersion;
    WORD     MinorOperatingSystemVersion;
    WORD     MajorImageVersion;
    WORD     MinorImageVersion;
    WORD     MajorSubsystemVersion;
    WORD     MinorSubsystemVersion;
    DWORD    Win32VersionValue;
    DWORD    SizeOfImage;
    DWORD    SizeOfHeaders;
    DWORD    CheckSum;
    WORD     Subsystem;
    WORD     DllCharacteristics;
    DWORD    SizeOfStackReserve;
    DWORD    SizeOfStackCommit;
    DWORD    SizeOfHeapReserve;
    DWORD    SizeOfHeapCommit;
    DWORD    LoaderFlags;
    DWORD    NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER;

typedef struct _IMAGE_OPTIONAL_HEADER64 {
    WORD     Magic;
    BYTE     MajorLinkerVersion;
    BYTE     MinorLinkerVersion;
    DWORD    SizeOfCode;
    DWORD    SizeOfInitializedData;
    DWORD    SizeOfUninitializedData;
    DWORD    AddressOfEntryPoint;
    DWORD    BaseOfCode;
    ULONGLONG ImageBase;
    DWORD    SectionAlignment;
    DWORD    FileAlignment;
    WORD     MajorOperatingSystemVersion;
    WORD     MinorOperatingSystemVersion;
    WORD     MajorImageVersion;
    WORD     MinorImageVersion;
    WORD     MajorSubsystemVersion;
    WORD     MinorSubsystemVersion;
    DWORD    Win32VersionValue;
    DWORD    SizeOfImage;
    DWORD    SizeOfHeaders;
    DWORD    CheckSum;
    WORD     Subsystem;
    WORD     DllCharacteristics;
    ULONGLONG SizeOfStackReserve;

```

(continues on next page)

(continued from previous page)

```

        ULONGLONG    SizeOfStackCommit;
        ULONGLONG    SizeOfHeapReserve;
        ULONGLONG    SizeOfHeapCommit;
        DWORD        LoaderFlags;
        DWORD        NumberOfRvaAndSizes;
        IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
    } IMAGE_OPTIONAL_HEADER64;

typedef struct _IMAGE_SECTION_HEADER {
    char    Name[IMAGE_SIZEOF_SHORT_NAME];
    ULONG    VirtualSize;
    ULONG    VirtualAddress;
    ULONG    SizeOfRawData;
    ULONG    PointerToRawData;
    ULONG    PointerToRelocations;
    ULONG    PointerToLinenumbers;
    USHORT    NumberOfRelocations;
    USHORT    NumberOfLinenumbers;
    ULONG    Characteristics;
} IMAGE_SECTION_HEADER;

typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        ULONG    Characteristics;
        ULONG    OriginalFirstThunk;
    } u;
    ULONG    TimeDateStamp;
    ULONG    ForwarderChain;
    ULONG    Name;
    ULONG    FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;

typedef struct _IMAGE_EXPORT_DIRECTORY {
    ULONG    Characteristics;
    ULONG    TimeDateStamp;
    USHORT    MajorVersion;
    USHORT    MinorVersion;
    ULONG    Name;
    ULONG    Base;
    ULONG    NumberOfFunctions;
    ULONG    NumberOfNames;
    ULONG    AddressOfFunctions;    // RVA from base of image
    ULONG    AddressOfNames;        // RVA from base of image
    ULONG    AddressOfNameOrdinals; // RVA from base of image
} IMAGE_EXPORT_DIRECTORY;

```

dissect.cobaltstrike.pe.peststruct

dissect.cobaltstrike.pe.DOSHEADER_X64

dissect.cobaltstrike.pe.DOSHEADER_X86

`dissect.cobaltstrike.pe.find_mz_offset(fh: BinaryIO, start_offset: int = 0, maxrange: int = 1024) → int | None`

Find and return the start offset of a valid IMAGE_DOS_HEADER or `None` if it cannot be found.

It uses `IMAGE_DOS_HEADER.e_lfanew` and `IMAGE_FILE_HEADER.Machine` as a constraint.

Side effects: file handle position due to seeking

Parameters

- **fh** – file like object
- **start_offset** – offset to start searching from, `None` indicates from current file position
- **maxrange** – how far to search for into the file object

Returns

offset of the start of IMAGE_DOS_HEADER in the file object or `None` if it's not found

`dissect.cobaltstrike.pe.find_compile_stamps(fh: BinaryIO, start_offset: int = 0, maxrange: int = 1024) → Tuple[int | None, int | None]`

Find and return a tuple with the *PE compile* and *PE export* timestamps.

If one or more *TimeDateStamps* are not found it will be returned as `None` in the tuple.

Side effects: file handle position due to seeking

Parameters

- **fh** – file like object
- **start_offset** – offset to start searching from, `None` indicates from current file position
- **maxrange** – how far to search for into the file object

Returns

Tuple with `(IMAGE_FILE_HEADER.TimeDateStamp, IMAGE_EXPORT_DIRECTORY.TimeDateStamp)`. Either tuple values can be `None` if it's not found.

`dissect.cobaltstrike.pe.find_magic_mz(fh: BinaryIO, start_offset: int = 0, maxrange: int = 1024) → bytes | None`

Find and returns the MZ header bytes or `None` if cannot be found

Cobalt Strike allows changing the MZ magic header using `magic_mz_x86` or `magic_mz_x64` in the c2 profile. This function recovers these bytes.

Side effects: file handle position due to seeking

Parameters

- **fh** – file like object
- **start_offset** – offset to start searching from, `None` indicates from current file position
- **maxrange** – how far to search for into the file object

Returns

MZ header bytes or `None` if not found.

`dissect.cobaltstrike.pe.find_magic_pe(fh: BinaryIO, start_offset: int = 0, maxrange: int = 1024) → bytes | None`

Find and returns the PE header (`magic_pe`) bytes or `None` if cannot be found

Cobalt Strike allows changing the PE magic header using the `magic_pe` in the malleable c2 profile. This function tries to recovers these bytes.

Side effects: file handle position due to seeking

Parameters

- **fh** – file like object
- **start_offset** – offset to start searching from, None indicates from current file position
- **maxrange** – how far to search for into the file object

Returns

PE header bytes or None if not found.

`dissect.cobaltstrike.pe.find_stage_prepend_append(fh: BinaryIO, start_offset: int = 0, maxrange: int = 1024) → Tuple[bytes | None, bytes | None]`

Find and return the stage prepend and append bytes as a tuple.

Cobalt Strike allows prepending and appending extra bytes to the beacon using malleable c2 profile settings. This function tries to recover these bytes.

Side effects: file handle position due to seeking

Parameters

- **fh** – file like object
- **start_offset** – offset to start searching from, None indicates from current file position
- **maxrange** – how far to search for into the file object

Returns

Tuple containing (prepend_bytes, append_bytes). Either tuple values can be None if it's not found.

`dissect.cobaltstrike.pe.find_architecture(fh: BinaryIO, start_offset: int = 0, maxrange: int = 1024) → str | None`

Find and return the PE image architecture, either "x86" or "x64" or None if not found.

It uses `IMAGE_DOS_HEADER.e_lfanew` and `IMAGE_FILE_HEADER.Machine` as a constraint.

Only `x86` and `x64` are considered, other machine architectures are ignored.

Side effects: file handle position due to seeking

Parameters

- **fh** – file like object
- **start_offset** – offset to start searching from, None indicates from current file position
- **maxrange** – how far to search for into the file object

Returns

"x86" or "x64", None if not found.

11.1.9 dissect.cobaltstrike.utils

This module contains generic helper functions used by `dissect.cobaltstrike`.

Module Contents

Classes

<code>LRUDict</code>	Limit size, evicting the least recently looked-up key when full
----------------------	---

Functions

<code>xor(→ bytes)</code>	XOR data with key (simd version)
<code>netbios_encode(→ bytes)</code>	Encode <i>data</i> using NetBIOS encoding and return the encoded bytes.
<code>netbios_decode(→ bytes)</code>	Decode the netbios encoded <i>data</i> and return the decoded bytes.
<code>retain_file_offset(fobj[, offset, whence])</code>	Return a context manager that changes the position of the file-like object <i>fobj</i> to the given byte <i>offset</i> .
<code>catch_sigpipe(func)</code>	Decorator for catching KeyboardInterrupt and BrokenPipeError (OSError 22 on Windows).
<code>unpack(→ int)</code>	
<code>pack(→ bytes)</code>	
<code>iter_find_needle(→ Iterator[int])</code>	Return an iterator yielding <i>offset</i> for found <i>needle</i> bytes in file <i>fp</i> .
<code>checksum8(→ int)</code>	Compute the <i>checksum8</i> value of text
<code>is_stager_x86(→ bool)</code>	Return True if URI is a x86 stager URI, otherwise False
<code>is_stager_x64(→ bool)</code>	Return True if URI is a x64 stager URI, otherwise False
<code>random_stager_uri(→ str)</code>	Generate a random (valid <i>checksum8</i>) stager URI. Defaults to x86 URIs unless <i>x64</i> is True.
<code>namedtuple_reprlib_repr(→ str)</code>	Return a <i>reprlib</i> version of <code>__repr__</code> for namedtuple <i>nt</i>
<code>enable_reprlib_cstruct()</code>	Enable <i>reprlib</i> style <code>__repr__</code> for <i>dissect.cstruct</i> instances.
<code>enable_reprlib_flow_record()</code>	Enable <i>reprlib</i> style <code>__repr__</code> for <i>flow.record</i> instances.

Attributes

<code>unpack_be</code>
<code>pack_be</code>
<code>u8</code>
<code>p8</code>
<code>u16</code>
<code>p16</code>
<code>u16be</code>
<code>p16be</code>
<code>u32</code>
<code>p32</code>
<code>u32be</code>
<code>p32be</code>
<code>u64</code>
<code>p64</code>
<code>u64be</code>
<code>p64be</code>

`dissect.cobaltstrike.utils.xor(data: bytes, key: bytes) → bytes`

XOR data with key (simd version)

`dissect.cobaltstrike.utils.netbios_encode(data: bytes, offset: int = 65) → bytes`

Encode *data* using NetBIOS encoding and return the encoded bytes.

Parameters

- **data** – bytes to be NetBIOS encoded
- **offset** – offset used for encoding, defaults to char A (0x41)

Returns

NetBIOS encoded bytes

`dissect.cobaltstrike.utils.netbios_decode(data: bytes, offset: int = 65) → bytes`

Decode the netbios encoded *data* and return the decoded bytes.

Parameters

- **data** – bytes to be NetBIOS decoded

- **offset** – offset used for decoding, defaults to char A (0x41)

Returns

NetBIOS decoded bytes

`dissect.cobaltstrike.utils.retain_file_offset(fobj, offset=None, whence=io.SEEK_SET)`

Return a context manager that changes the position of the file-like object *fobj* to the given byte *offset*. After completion of the block it restores the original position of the file.

Parameters

- **fobj** – file-like object
- **offset** – offset to seek to relative to position indicated by *whence*. If *None* no seek will be done.
- **whence** – default is `SEEK_SET`, values for *whence* are:
 - `SEEK_SET` or `0` – start of the stream (the default); offset should be zero or positive
 - `SEEK_CUR` or `1` – current stream position; offset may be negative
 - `SEEK_END` or `2` – end of the stream; offset is usually negative

Returns

context manager

`dissect.cobaltstrike.utils.catch_sigpipe(func)`

Decorator for catching `KeyboardInterrupt` and `BrokenPipeError` (`OSError 22` on Windows).

`dissect.cobaltstrike.utils.unpack(data: bytes, size: int = None, byteorder='little', signed=False) → int`

`dissect.cobaltstrike.utils.pack(n: int, size: int = None, byteorder='little', signed=False) → bytes`

`dissect.cobaltstrike.utils.unpack_be`

`dissect.cobaltstrike.utils.pack_be`

`dissect.cobaltstrike.utils.u8`

`dissect.cobaltstrike.utils.p8`

`dissect.cobaltstrike.utils.u16`

`dissect.cobaltstrike.utils.p16`

`dissect.cobaltstrike.utils.u16be`

`dissect.cobaltstrike.utils.p16be`

`dissect.cobaltstrike.utils.u32`

`dissect.cobaltstrike.utils.p32`

`dissect.cobaltstrike.utils.u32be`

`dissect.cobaltstrike.utils.p32be`

`dissect.cobaltstrike.utils.u64`

`dissect.cobaltstrike.utils.p64`

`dissect.cobaltstrike.utils.u64be`

dissect.cobaltstrike.utils.p64be

dissect.cobaltstrike.utils.iter_find_needle(fp: BinaryIO, needle: bytes, start_offset: int = None, max_offset: int = 0) → Iterator[int]

Return an iterator yielding *offset* for found *needle* bytes in file *fp*.

Side effects: file handle position due to seeking.

Parameters

- **fp** – file like object
- **needle** – needle to search for
- **start_offset** – offset in file object to start searching from, if None it will search from current position
- **max_offset** – how far we search for into the file, 0 for no limit

Yields

offset where *needle* was found in file *fp*

dissect.cobaltstrike.utils.checksum8(text: str) → int

Compute the *checksum8* value of text

dissect.cobaltstrike.utils.is_stager_x86(uri: str) → bool

Return True if URI is a x86 stager URI, otherwise False

dissect.cobaltstrike.utils.is_stager_x64(uri: str) → bool

Return True if URI is a x64 stager URI, otherwise False

dissect.cobaltstrike.utils.random_stager_uri(*, x64: bool = False, length: int = 4) → str

Generate a random (valid *checksum8*) stager URI. Defaults to x86 URIs unless *x64* is True.

Parameters

- **x64** – generate a x64 stager URI if True, False for a x86 stager URI. (default: False)
- **length** – length of URI to generate, excluding the “/” prefix. (default: 4)

Returns

random stager URI

dissect.cobaltstrike.utils.namedtuple_reprlib_repr(nt: NamedTuple) → str

Return a *reprlib* version of `__repr__` for namedtuple *nt*

dissect.cobaltstrike.utils.enable_reprlib_cstruct()

Enable *reprlib* style `__repr__` for *dissect.cstruct* instances.

dissect.cobaltstrike.utils.enable_reprlib_flow_record()

Enable *reprlib* style `__repr__` for *flow.record* instances.

class dissect.cobaltstrike.utils.LRUDict(maxsize=128, *args, **kwargs)

Bases: collections.OrderedDict

Limit size, evicting the least recently looked-up key when full

`__getitem__`(key)

x.__getitem__(y) <==> x[y]

`__setitem__`(key, value)

Set self[key] to value.

11.1.10 dissect.cobaltstrike.version

This module contains the *BeaconVersion* class and mappings for determining the Cobalt Strike version of beacon payloads.

Note: Deducing the Cobalt Strike version using *BeaconVersion.from_pe_export_stamp()* is more accurate than *BeaconVersion.from_max_setting_enum()*. However, if the *pe_export_stamp* is not known, deducing from *max_setting_enum* is still a good version estimate.

Module Contents

Classes

<i>BeaconVersion</i>	Helper class for dealing with Cobalt Strike version strings
----------------------	---

Attributes

<i>MAX_ENUM_TO_VERSION</i>	Max setting enum to Cobalt Strike version mapping
<i>PE_EXPORT_STAMP_TO_VERSION</i>	PE export timestamp to Cobalt Strike version mapping

`dissect.cobaltstrike.version.MAX_ENUM_TO_VERSION: Dict[int, str]`

Max setting enum to Cobalt Strike version mapping

`dissect.cobaltstrike.version.PE_EXPORT_STAMP_TO_VERSION: Dict[int, str]`

PE export timestamp to Cobalt Strike version mapping

`class dissect.cobaltstrike.version.BeaconVersion(version: str)`

Bases: `str`

Helper class for dealing with Cobalt Strike version strings

property version_string: str

The version string without the date. e.g. "Cobalt Strike 4.5"

property version_only: str

The version number only string. e.g. "4.5", or "Unknown" if version is unknown.

REGEX_VERSION = 'Cobalt Strike

(?P<major>\\d+)\\. (?P<minor>\\d+) (\\. (?P<patch>\\d+))? \\((?P<date>.*))\\)

version: str

full version string including date, e.g. "Cobalt Strike 4.5 (Dec 14, 2021)"

tuple: Tuple[int, int] | Tuple[int, int, int] | None

the version as tuple of (major, minor) or (major, minor, patch), e.g. (4, 5) or (4, 7, 1). Otherwise, None.

date: `datetime.date` | `None`

date of version as `datetime.date` object, e.g. `datetime.date(2021, 12, 14)`. Otherwise, `None`.

classmethod `from_pe_export_stamp(pe_export_stamp: int) → BeaconVersion`

Construct `BeaconVersion` by looking up `pe_export_stamp` in the `PE_EXPORT_STAMP_TO_VERSION` map.

classmethod `from_max_setting_enum(enum: int) → BeaconVersion`

Construct `BeaconVersion` by looking up `enum` in the `MAX_ENUM_TO_VERSION` map.

__str__() → `str`

Return `str(self)`.

__repr__() → `str`

Return `repr(self)`.

11.1.11 dissect.cobaltstrike.xordecode

This module is responsible for decoding XorEncoded Cobalt Strike payloads. Not to be confused with the single byte XOR key that is used to obfuscate the beacon configuration block.

Module Contents

Classes

<code>XorEncodedFile</code>	A file object providing transparent decoding of XorEncoded files.
-----------------------------	---

Functions

<code>iter_nonce_offsets(→ Iterator[int])</code>	Returns a generator that yields nonce offset candidates based on encoded <code>real_size</code> .
<code>main()</code>	Entrypoint for <code>beacon-xordecode</code>

Attributes

<code>logger</code>

`dissect.cobaltstrike.xordecode.logger`

`dissect.cobaltstrike.xordecode.iter_nonce_offsets(fh: BinaryIO, real_size: int = None, maxrange: int = 1024) → Iterator[int]`

Returns a generator that yields nonce offset candidates based on encoded `real_size`.

If `real_size` is `None` it will automatically determine the size from `fh`. It tries to find the *nonce offset* using the following structure.

| nonce (dword) | encoded_size (dword) | encoded MZ + payload |

Side effects: file handle position due to seeking

Parameters

- **fh** – file like object
- **real_size** – encoded_size to search for, or automatically determined from fh if None.
- **maxrange** – maximum range to search for

Yields

nonce_offset candidates

class dissect.cobaltstrike.xordecode.XorEncodedFile(fh: BinaryIO, nonce_offset: int = 0)

Bases: io.RawIOBase

A file object providing transparent decoding of XorEncoded files.

To verify if a file is a XorEncoded Beacon, use the [XorEncodedFile.from_file\(\)](#) constructor which raises ValueError if it cannot find a nonce candidate or valid MZ header.

To skip any validation checks, construct via [XorEncodedFile\(\)](#) using *nonce_offset*.

EOF_SHELLCODE_MARKER = b'\xff\xff\xff'

__repr__() → str

Return repr(self).

classmethod from_file(fh: BinaryIO, maxrange: int = 1024) → [XorEncodedFile](#)

Constructs a XorEncodedFile from file *fh*, raises ValueError if file not determined as a XorEncoded Beacon.

This constructor will try to find the correct *nonce_offset* by using the following methods:

- **end of shellcode offset**: will try to find the end of the shellcode stub.
- **real_size**: using [iter_nonce_offsets\(\)](#) to find candidate offsets based on size.

The *nonce_offset* candidates are then checked to see if there is a valid MZ header.

Parameters

- **fh** – file-like object
- **maxrange** – how far into the file should be try to find the *nonce_offset* candidates (default 1024)

Returns

XorEncodedFile instance

Raises

ValueError – If it cannot find a *nonce_offset* or valid MZ header

classmethod from_path(path: str | os.PathLike, maxrange: int = 1024) → [XorEncodedFile](#)

Constructs a [XorEncodedFile](#) from path *path*.

This is more of a convenience method as it calls [XorEncodedFile.from_file\(\)](#) under the hood.

Parameters

- **path** – path or path-like to xorencoded file
- **maxrange** – how far into the file should be try to find the *nonce_offset* candidates (default 1024)

Returns

XorEncodedFile instance

Raises

ValueError – If it cannot find a *nonce_offset* or valid *MZ header*

read_nonce()

Return nonce for current file position or 0 if it cannot be read

tell()

Return current stream position.

seek(*offset*, *whence*=*io.SEEK_SET*)

Change stream position.

Change the stream position to the given byte offset. The offset is interpreted relative to the position indicated by whence. Values for whence are:

- 0 – start of stream (the default); offset should be zero or positive
- 1 – current stream position; offset may be negative
- 2 – end of stream; offset is usually negative

Return the new absolute position.

read(*n=-1*)**dissect.cobaltstrike.xordecode.main()**

Entrypoint for *beacon-xordecode*

STRUCTURE DEFINITIONS

`dissect.cobaltstrike` uses `dissect.cstruct` for parsing data using C structures.

12.1 `dissect.cobaltstrike.beacon.CS_DEF`

Structures for parsing Cobalt Strike Beacon configuration and settings.

```
enum BeaconSetting: uint16 {
    SETTING_PROTOCOL = 1,
    SETTING_PORT = 2,
    SETTING_SLEEPTIME = 3,
    SETTING_MAXGET = 4,
    SETTING_JITTER = 5,
    SETTING_MAXDNS = 6,
    SETTING_PUBKEY = 7,
    SETTING_DOMAINS = 8,
    SETTING_USERAGENT = 9,
    SETTING_SUBMITURI = 10,
    SETTING_C2_RECOVER = 11,
    SETTING_C2_REQUEST = 12,
    SETTING_C2_POSTREQ = 13,
    SETTING_SPAWNTO = 14,           // releasenotes.txt

    // CobaltStrike version >= 3.4 (27 Jul, 2016)
    SETTING_PIPENAME = 15,

    SETTING_KILLDATE_YEAR = 16,     // Deprecated since Cobalt Strike 4.7
    SETTING_BOF_ALLOCATOR = 16,    // Introduced in Cobalt Strike 4.7

    SETTING_KILLDATE_MONTH = 17,    // Deprecated since Cobalt Strike 4.8
    SETTING_SYSCALL_METHOD = 17,    // Introduced in Cobalt Strike 4.8

    SETTING_KILLDATE_DAY = 18,
    SETTING_DNS_IDLE = 19,
    SETTING_DNS_SLEEP = 20,

    // CobaltStrike version >= 3.5 (22 Sept, 2016)
    SETTING_SSH_HOST = 21,
    SETTING_SSH_PORT = 22,
    SETTING_SSH_USERNAME = 23,
```

(continues on next page)

(continued from previous page)

```
SETTING_SSH_PASSWORD = 24,  
SETTING_SSH_KEY = 25,  
SETTING_C2_VERB_GET = 26,  
SETTING_C2_VERB_POST = 27,  
SETTING_C2_CHUNK_POST = 28,  
SETTING_SPAWNT0_X86 = 29,  
SETTING_SPAWNT0_X64 = 30,  
  
// CobaltStrike version >= 3.6 (8 Dec, 2016)  
SETTING_CRYPT0_SCHEME = 31,  
  
// CobaltStrike version >= 3.7 (15 Mar, 2016)  
SETTING_PROXY_CONFIG = 32,  
SETTING_PROXY_USER = 33,  
SETTING_PROXY_PASSWORD = 34,  
SETTING_PROXY_BEHAVIOR = 35,  
  
// CobaltStrike version >= 3.8 (23 May 2017)  
// DEPRECATED_SETTING_INJECT_OPTIONS = 36,  
  
// Renamed from DEPRECATED_SETTING_INJECT_OPTIONS in CobaltStrike 4.5  
SETTING_WATERMARKHASH = 36,  
  
// CobaltStrike version >= 3.9 (Sept 26, 2017)  
SETTING_WATERMARK = 37,  
  
// CobaltStrike version >= 3.11 (April 9, 2018)  
SETTING_CLEANUP = 38,  
  
// CobaltStrike version >= 3.11 (May 24, 2018)  
SETTING_CFG_CAUTION = 39,  
  
// CobaltStrike version >= 3.12 (Sept 6, 2018)  
SETTING_KILLDATE = 40,  
SETTING_GARGLE_NOOK = 41, // https://www.youtube.com/watch?v=nLTgWdXrx3U  
SETTING_GARGLE_SECTIONS = 42,  
SETTING_PROCINJ_PERMS_I = 43,  
SETTING_PROCINJ_PERMS = 44,  
SETTING_PROCINJ_MINALLOC = 45,  
SETTING_PROCINJ_TRANSFORM_X86 = 46,  
SETTING_PROCINJ_TRANSFORM_X64 = 47,  
  
SETTING_PROCINJ_ALLOWED = 48, // Deprecated since Cobalt Strike 4.7  
SETTING_PROCINJ_BOF_REUSE_MEM = 48, // Introduced in Cobalt Strike 4.7  
  
// CobaltStrike version >= 3.13 (Jan 2, 2019)  
SETTING_BINDHOST = 49,  
  
// CobaltStrike version >= 3.14 (May 4, 2019)  
SETTING_HTTP_NO_COOKIES = 50,  
SETTING_PROCINJ_EXECUTE = 51,  
SETTING_PROCINJ_ALLOCATOR = 52,
```

(continues on next page)

(continued from previous page)

```

SETTING_PROCIJN_STUB = 53,      // .self = MD5(cobaltstrike.jar)

// CobaltStrike version >= 4.0 (Dec 5, 2019)
SETTING_HOST_HEADER = 54,
SETTING_EXIT_FUNK = 55,

// CobaltStrike version >= 4.1 (June 25, 2020)
SETTING_SSH_BANNER = 56,
SETTING_SMB_FRAME_HEADER = 57,
SETTING_TCP_FRAME_HEADER = 58,

// CobaltStrike version >= 4.2 (Nov 6, 2020)
SETTING_HEADERS_REMOVE = 59,

// CobaltStrike version >= 4.3 (Mar 3, 2021)
SETTING_DNS_BEACON_BEACON = 60,
SETTING_DNS_BEACON_GET_A = 61,
SETTING_DNS_BEACON_GET_AAAA = 62,
SETTING_DNS_BEACON_GET_TXT = 63,
SETTING_DNS_BEACON_PUT_METADATA = 64,
SETTING_DNS_BEACON_PUT_OUTPUT = 65,
SETTING_DNSRESOLVER = 66,
SETTING_DOMAIN_STRATEGY = 67,
SETTING_DOMAIN_STRATEGY_SECONDS = 68,
SETTING_DOMAIN_STRATEGY_FAIL_X = 69,
SETTING_DOMAIN_STRATEGY_FAIL_SECONDS = 70,

// CobaltStrike version >= 4.5 (Dec 14, 2021)
SETTING_MAX_RETRY_STRATEGY_ATTEMPTS = 71,
SETTING_MAX_RETRY_STRATEGY_INCREASE = 72,
SETTING_MAX_RETRY_STRATEGY_DURATION = 73,

// CobaltStrike version >= 4.7 (Aug 17, 2022)
SETTING_MASKED_WATERMARK = 74,
};

enum DeprecatedBeaconSetting: uint16 {
    SETTING_KILLDATE_YEAR = 16,
    SETTING_INJECT_OPTIONS = 36,
};

enum TransformStep: uint32 {
    APPEND = 1,
    PREPEND = 2,
    BASE64 = 3,
    PRINT = 4,
    PARAMETER = 5,
    HEADER = 6,
    BUILD = 7,
    NETBIOS = 8,
    _PARAMETER = 9,
    _HEADER = 10,

```

(continues on next page)

(continued from previous page)

```

    NETBIOSU = 11,
    URI_APPEND = 12,
    BASE64URL = 13,
    STRREP = 14,
    MASK = 15,
    // CobaltStrike version >= 4.0 (Dec 5, 2019)
    _HOSTHEADER = 16,
};

enum SettingsType: uint16 {
    TYPE_NONE = 0,
    TYPE_SHORT = 1,
    TYPE_INT = 2,
    TYPE_PTR = 3,
};

struct Setting {
    BeaconSetting index;    // uint16
    SettingsType type;      // uint16
    uint16 length;          // uint16
    char value[length];
};

flag BeaconProtocol {
    http = 0,
    dns = 1,
    smb = 2,
    tcp = 4,
    https = 8,
    bind = 16
};

flag ProxyServer {
    MANUAL = 0,
    DIRECT = 1,
    PRECONFIG = 2,
    MANUAL_CREDS = 4
};

enum CryptoScheme: uint16 {
    CRYPTO_LICENSED_PRODUCT = 0,
    CRYPTO_TRIAL_PRODUCT = 1
};

enum InjectAllocator: uint8 {
    VirtualAllocEx = 0,
    NtMapViewOfSection = 1,
};

enum InjectExecutor: uint8 {
    CreateThread = 1,
    SetThreadContext = 2,
};

```

(continues on next page)

(continued from previous page)

```

    CreateRemoteThread = 3,
    RtlCreateUserThread = 4,
    NtQueueApcThread = 5,
    CreateThread_ = 6,
    CreateRemoteThread_ = 7,
    NtQueueApcThread_s = 8
};

```

12.2 dissect.cobaltstrike.pe.PE_DEF

Structures for parsing PE headers.

```

#define IMAGE_FILE_MACHINE_AMD64    0x8664
#define IMAGE_FILE_MACHINE_I386     0x014c
#define IMAGE_FILE_MACHINE_IA64     0x0200

#define IMAGE_NUMBEROF_DIRECTORY_ENTRIES 16
#define IMAGE_SIZEOF_SHORT_NAME      8

#define IMAGE_DIRECTORY_ENTRY_EXPORT 0
#define IMAGE_DIRECTORY_ENTRY_IMPORT 1
#define IMAGE_DIRECTORY_ENTRY_RESOURCE 2

typedef struct _IMAGE_DOS_HEADER
{
    WORD e_magic;
    WORD e_cblp;
    WORD e_cp;
    WORD e_crlc;
    WORD e_cparhdr;
    WORD e_minalloc;
    WORD e_maxalloc;
    WORD e_ss;
    WORD e_sp;
    WORD e_csum;
    WORD e_ip;
    WORD e_cs;
    WORD e_lfarlc;
    WORD e_ovno;
    WORD e_res[4];
    WORD e_oemid;
    WORD e_oeminfo;
    WORD e_res2[10];
    LONG e_lfanew;
} IMAGE_DOS_HEADER;

typedef struct _IMAGE_FILE_HEADER {
    WORD Machine;
    WORD NumberOfSections;
    DWORD TimeDateStamp;

```

(continues on next page)

(continued from previous page)

```

    DWORD PointerToSymbolTable;
    DWORD NumberOfSymbols;
    WORD   SizeOfOptionalHeader;
    WORD   Characteristics;
} IMAGE_FILE_HEADER;

typedef struct _IMAGE_DATA_DIRECTORY {
    ULONG   VirtualAddress;
    ULONG   Size;
} IMAGE_DATA_DIRECTORY;

typedef struct _IMAGE_OPTIONAL_HEADER {
    WORD           Magic;
    BYTE           MajorLinkerVersion;
    BYTE           MinorLinkerVersion;
    DWORD          SizeOfCode;
    DWORD          SizeOfInitializedData;
    DWORD          SizeOfUninitializedData;
    DWORD          AddressOfEntryPoint;
    DWORD          BaseOfCode;
    DWORD          BaseOfData;
    DWORD          ImageBase;
    DWORD          SectionAlignment;
    DWORD          FileAlignment;
    WORD           MajorOperatingSystemVersion;
    WORD           MinorOperatingSystemVersion;
    WORD           MajorImageVersion;
    WORD           MinorImageVersion;
    WORD           MajorSubsystemVersion;
    WORD           MinorSubsystemVersion;
    DWORD          Win32VersionValue;
    DWORD          SizeOfImage;
    DWORD          SizeOfHeaders;
    DWORD          CheckSum;
    WORD           Subsystem;
    WORD           DllCharacteristics;
    DWORD          SizeOfStackReserve;
    DWORD          SizeOfStackCommit;
    DWORD          SizeOfHeapReserve;
    DWORD          SizeOfHeapCommit;
    DWORD          LoaderFlags;
    DWORD          NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER;

typedef struct _IMAGE_OPTIONAL_HEADER64 {
    WORD           Magic;
    BYTE           MajorLinkerVersion;
    BYTE           MinorLinkerVersion;
    DWORD          SizeOfCode;
    DWORD          SizeOfInitializedData;
    DWORD          SizeOfUninitializedData;

```

(continues on next page)

(continued from previous page)

```

    DWORD    AddressOfEntryPoint;
    DWORD    BaseOfCode;
    ULONGLONG ImageBase;
    DWORD    SectionAlignment;
    DWORD    FileAlignment;
    WORD     MajorOperatingSystemVersion;
    WORD     MinorOperatingSystemVersion;
    WORD     MajorImageVersion;
    WORD     MinorImageVersion;
    WORD     MajorSubsystemVersion;
    WORD     MinorSubsystemVersion;
    DWORD    Win32VersionValue;
    DWORD    SizeOfImage;
    DWORD    SizeOfHeaders;
    DWORD    CheckSum;
    WORD     Subsystem;
    WORD     DllCharacteristics;
    ULONGLONG SizeOfStackReserve;
    ULONGLONG SizeOfStackCommit;
    ULONGLONG SizeOfHeapReserve;
    ULONGLONG SizeOfHeapCommit;
    DWORD    LoaderFlags;
    DWORD    NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER64;

typedef struct _IMAGE_SECTION_HEADER {
    char    Name[IMAGE_SIZEOF_SHORT_NAME];
    ULONG   VirtualSize;
    ULONG   VirtualAddress;
    ULONG   SizeOfRawData;
    ULONG   PointerToRawData;
    ULONG   PointerToRelocations;
    ULONG   PointerToLinenumbers;
    USHORT  NumberOfRelocations;
    USHORT  NumberOfLinenumbers;
    ULONG   Characteristics;
} IMAGE_SECTION_HEADER;

typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        ULONG   Characteristics;
        ULONG   OriginalFirstThunk;
    } u;
    ULONG   TimeDateStamp;
    ULONG   ForwarderChain;
    ULONG   Name;
    ULONG   FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;

typedef struct _IMAGE_EXPORT_DIRECTORY {
    ULONG   Characteristics;

```

(continues on next page)

(continued from previous page)

```

ULONG    TimeDateStamp;
USHORT   MajorVersion;
USHORT   MinorVersion;
ULONG    Name;
ULONG    Base;
ULONG    NumberOfFunctions;
ULONG    NumberOfNames;
ULONG    AddressOfFunctions;    // RVA from base of image
ULONG    AddressOfNames;       // RVA from base of image
ULONG    AddressOfNameOrdinals; // RVA from base of image
} IMAGE_EXPORT_DIRECTORY;

```

12.3 dissect.cobaltstrike.c_c2.C2_DEF

Structures for parsing C2 headers.

```

// Callback data from: Beacon -> Team Server
typedef struct CallbackPacket {
    uint32 counter;
    uint32 size;
    BeaconCallback callback;
    char data[size];
};

// Task from: Team Server -> Beacon
typedef struct TaskPacket {
    uint32 epoch;
    uint32 total_size;
    BeaconCommand command;
    uint32 size;
    char data[size];
};

struct BeaconMetadata {
    uint32 magic;
    uint32 size;
    char aes_rand[16];
    uint16 ansi_cp;    // GetACP
    uint16 oem_cp;    // GetOEMCP
    uint32 bid;
    uint32 pid;
    uint16 port;
    uint8 flag;
    uint8 ver_major;
    uint8 ver_minor;
    uint16 ver_build;
    uint32 ptr_x64;    // for x64 addressing
    uint32 ptr_gmh;    // GetModuleHandle
    uint32 ptr_gpa;    // GetProcAddress
    uint32 ip;

```

(continues on next page)

(continued from previous page)

```
char info[size - 51];
};
```

Enums:

```
class BeaconCommand(IntEnum):
    COMMAND_SPAWN = 1
    COMMAND_SHELL = 2
    COMMAND_DIE = 3
    COMMAND_SLEEP = 4
    COMMAND_CD = 5
    COMMAND_KEYLOG_START = 6
    COMMAND_NOOP = 6
    COMMAND_KEYLOG_STOP = 7
    COMMAND_CHECKIN = 8
    COMMAND_INJECT_PID = 9
    COMMAND_UPLOAD = 10
    COMMAND_DOWNLOAD = 11
    COMMAND_EXECUTE = 12
    COMMAND_SPAWN_PROC_X86 = 13
    COMMAND_CONNECT = 14
    COMMAND_SEND = 15
    COMMAND_CLOSE = 16
    COMMAND_LISTEN = 17
    COMMAND_INJECT_PING = 18
    COMMAND_CANCEL_DOWNLOAD = 19
    COMMAND_PIPE_ROUTE = 22
    COMMAND_PIPE_CLOSE = 23
    COMMAND_PIPE_REOPEN = 24
    COMMAND_TOKEN_GETUID = 27
    COMMAND_TOKEN_REV2SELF = 28
    COMMAND_TIMESTOMP = 29
    COMMAND_STEAL_TOKEN = 31
    COMMAND_PS_LIST = 32
    COMMAND_PS_KILL = 33
    COMMAND_PSH_IMPORT = 37
    COMMAND_RUNAS = 38
    COMMAND_PWD = 39
    COMMAND_JOB_REGISTER = 40
    COMMAND_JOBS = 41
    COMMAND_JOB_KILL = 42
    COMMAND_INJECTX64_PID = 43
    COMMAND_SPAWNX64 = 44
    COMMAND_INJECT_PID_PING = 45
    COMMAND_INJECTX64_PID_PING = 46
    COMMAND_PAUSE = 47
    COMMAND_LOGINUSER = 49
    COMMAND_LSOCKET_BIND = 50
    COMMAND_LSOCKET_CLOSE = 51
    COMMAND_STAGE_PAYLOAD = 52
    COMMAND_FILE_LIST = 53
    COMMAND_FILE_MKDIR = 54
```

(continues on next page)

(continued from previous page)

```

COMMAND_FILE_DRIVES = 55
COMMAND_FILE_RM = 56
COMMAND_STAGE_PAYLOAD_SMB = 57
COMMAND_WEBSERVER_LOCAL = 59
COMMAND_ELEVATE_PRE = 60
COMMAND_ELEVATE_POST = 61
COMMAND_JOB_REGISTER_IMPERSONATE = 62
COMMAND_SPAWN_POWERSHELLX86 = 63
COMMAND_SPAWN_POWERSHELLX64 = 64
COMMAND_INJECT_POWERSHELLX86_PID = 65
COMMAND_INJECT_POWERSHELLX64_PID = 66
COMMAND_UPLOAD_CONTINUE = 67
COMMAND_PIPE_OPEN_EXPLICIT = 68
COMMAND_SPAWN_PROC_X64 = 69
COMMAND_JOB_SPAWN_X86 = 70
COMMAND_JOB_SPAWN_X64 = 71
COMMAND_SETENV = 72
COMMAND_FILE_COPY = 73
COMMAND_FILE_MOVE = 74
COMMAND_PPID = 75
COMMAND_RUN_UNDER_PID = 76
COMMAND_GETPRIVS = 77
COMMAND_EXECUTE_JOB = 78
COMMAND_PSH_HOST_TCP = 79
COMMAND_DLL_LOAD = 80
COMMAND_REG_QUERY = 81
COMMAND_LSOCKET_TCPPIVOT = 82
COMMAND_ARGUE_ADD = 83
COMMAND_ARGUE_REMOVE = 84
COMMAND_ARGUE_LIST = 85
COMMAND_TCP_CONNECT = 86
COMMAND_JOB_SPAWN_TOKEN_X86 = 87
COMMAND_JOB_SPAWN_TOKEN_X64 = 88
COMMAND_SPAWN_TOKEN_X86 = 89
COMMAND_SPAWN_TOKEN_X64 = 90
COMMAND_INJECTX64_PING = 91
COMMAND_BLOCKDLLS = 92
COMMAND_SPAWNAS_X86 = 93
COMMAND_SPAWNAS_X64 = 94
COMMAND_INLINE_EXECUTE = 95
COMMAND_RUN_INJECT_X86 = 96
COMMAND_RUN_INJECT_X64 = 97
COMMAND_SPAWNU_X86 = 98
COMMAND_SPAWNU_X64 = 99
COMMAND_INLINE_EXECUTE_OBJECT = 100
COMMAND_JOB_REGISTER_MSGMODE = 101
COMMAND_LSOCKET_BIND_LOCALHOST = 102

```

```
class BeaconCallback(IntEnum):
```

```

    CALLBACK_OUTPUT = 0
    CALLBACK_KEYSTROKES = 1
    CALLBACK_FILE = 2

```

(continues on next page)

(continued from previous page)

```
CALLBACK_SCREENSHOT = 3
CALLBACK_CLOSE = 4
CALLBACK_READ = 5
CALLBACK_CONNECT = 6
CALLBACK_PING = 7
CALLBACK_FILE_WRITE = 8
CALLBACK_FILE_CLOSE = 9
CALLBACK_PIPE_OPEN = 10
CALLBACK_PIPE_CLOSE = 11
CALLBACK_PIPE_READ = 12
CALLBACK_POST_ERROR = 13
CALLBACK_PIPE_PING = 14
CALLBACK_TOKEN_STOLEN = 15
CALLBACK_TOKEN_GETUID = 16
CALLBACK_PROCESS_LIST = 17
CALLBACK_POST_REPLAY_ERROR = 18
CALLBACK_PWD = 19
CALLBACK_JOBS = 20
CALLBACK_HASHDUMP = 21
CALLBACK_PENDING = 22
CALLBACK_ACCEPT = 23
CALLBACK_NETVIEW = 24
CALLBACK_PORTSCAN = 25
CALLBACK_DEAD = 26
CALLBACK_SSH_STATUS = 27
CALLBACK_CHUNK_ALLOCATE = 28
CALLBACK_CHUNK_SEND = 29
CALLBACK_OUTPUT_OEM = 30
CALLBACK_ERROR = 31
CALLBACK_OUTPUT_UTF8 = 32
```


C2PROFILE GRAMMAR

`dissect.cobaltstrike` utilizes the `Lark` parser for parsing and generating Cobalt Strike Malleable C2 Profiles.

The `Lark` grammar file to parse the *Profile Language* is defined in `c2profile.lark` and listed below for reference.

Note: Currently, the grammar implementation is pretty naive and could be improved upon. For example, the values are all *STRING* but could benefit from other types as well.

```
start: value*

?value: "set" OPTION string ";"                                -> option
      | "http-config" "{" http_config_options* "}"           -> http_config
      | "https-certificate" variant? "{" https_certificate_options* "}" -> https_
      ↪ certificate
      | "code-signer" "{" code_signer_options* "}"           -> code_signer
      | "http-stager" variant? "{" http_stager_options* "}"   -> http_stager
      | "http-get" variant? "{" http_get_options* "}"         -> http_get
      | "http-post" variant? "{" http_post_options* "}"       -> http_post
      | "stage" "{" stage_options* "}"                       -> stage
      | "process-inject" "{" process_inject_options* "}"     -> process_inject
      | "post-ex" "{" postex_options* "}"                   -> post_ex
      | "dns-beacon" "{" dns_beacon_options* "}"             -> dns_beacon

OPTION: "sample_name"
      | "data_jitter"
      | "dns_idle"
      | "dns_max_txt"
      | "dns_sleep"
      | "dns_stager_prepend"
      | "dns_stager_subhost"
      | "dns_ttl"
      | "host_stage"
      | "jitter"
      | "maxdns"
      | "pipename"
      | "pipename_stager"
      | "sleeptime"
      | "smb_frame_header"
      | "ssh_banner"
      | "ssh_pipename"
```

(continues on next page)

(continued from previous page)

```

| "tcp_frame_header"
| "tcp_port"
| "useragent"
| "spawnnto"           // deprecated since Cobalt Strike 3.6
| "spawnnto_x86"       // moved to post-ex since Cobalt Strike 3.14
| "spawnnto_x64"       // moved to post-ex since Cobalt Strike 3.14
| "amsi_disable"       // moved to post-ex since Cobalt Strike 3.14
| "create_remote_thread" // deprecated since Cobalt Strike 3.12
| "hijack_remote_thread" // deprecated since Cobalt Strike 3.12
| "tasks_max_size"     // introduced in Cobalt Strike 4.6
| "tasks_proxy_max_size" // introduced in Cobalt Strike 4.6
| "tasks_dns_proxy_max_size" // introduced in Cobalt Strike 4.6

http_config_options: "set" "headers" string ";"      -> headers
| "header" string string ";"                        -> header
| "set" "trust_x_forwarded_for" string ";"          -> trust_x_forwarded_for
| "set" "block_useragents" string ";"              -> block_useragents
| "set" "allow_useragents" string ";"              -> allow_useragents

http_stager_options: "set" "uri_x86" string ";"      -> uri_x86
| "set" "uri_x64" string ";"                        -> uri_x64
| "client" "{" http_options* "}"                  -> client
| "server" "{" http_options* "}"                  -> server

http_options: "header" string string ";"           -> header
| "parameter" string string ";"                   -> parameter
| "output" "{" data_transform* "}"                 -> output

data_transform: steps termination

steps: transform_statement*
termination: termination_statement ~ 1

transform_statement: "append" string ";"           -> append
| "base64" ";"                                     -> base64
| "base64url" ";"                                 -> base64url
| "mask" ";"                                       -> mask
| "netbios" ";"                                   -> netbios
| "netbiosu" ";"                                  -> netbiosu
| "prepend" string ";"                           -> prepend

termination_statement: "header" string ";"         -> header
| "parameter" string ";"                         -> parameter
| "print" ";"                                    -> print
| "uri-append" ";"                              -> uri_append

stage_transform: "prepend" string ";"              -> prepend
| "append" string ";"                            -> append
| "strrep" string string ";"                      -> strrep

http_get_options: "set" "uri" string ";"           -> uri
| "set" "verb" string ";"                        -> verb

```

(continues on next page)

(continued from previous page)

```

| "client" "{" http_get_client_options* "}" -> client
| "server" "{" http_options* "}" -> server

http_get_client_options: "header" string string ";" -> header
| "set" "verb" string ";" -> verb
| "metadata" "{" data_transform* "}" -> metadata
| "id" "{" data_transform* "}" -> id
| "parameter" string string ";" -> parameter
| "output" "{" data_transform* "}" -> output

http_post_options: "set" "uri" string ";" -> uri
| "set" "verb" string ";" -> verb
| "client" "{" http_get_client_options* "}" -> client
| "server" "{" http_options* "}" -> server

https_certificate_options: "set" "C" string ";" -> country
| "set" "CN" string ";" -> common_name
| "set" "L" string ";" -> locality
| "set" "OU" string ";" -> org_unit
| "set" "O" string ";" -> org
| "set" "ST" string ";" -> state
| "set" "validity" string ";" -> validity
| "set" "keystore" string ";" -> keystore
| "set" "password" string ";" -> password

code_signer_options: "set" "keystore" string ";" -> keystore
| "set" "password" string ";" -> password
| "set" "alias" string ";" -> alias
| "set" "digest_algorithm" string ";" -> digest_algorithm
| "set" "timestamp" string ";" -> timestamp
| "set" "timestamp_url" string ";" -> timestamp_url

stage_options: "string" string ";" -> string
| "stringw" string ";" -> stringw
| "transform-x86" "{" stage_transform* "}" -> transform_x86
| "transform-x64" "{" stage_transform* "}" -> transform_x64
| "set" "allocator" string ";" -> allocator
| "set" "cleanup" string ";" -> cleanup
| "set" "magic_pe" string ";" -> magic_pe
| "set" "magic_mz_x86" string ";" -> magic_mz_x86
| "set" "magic_mz_x64" string ";" -> magic_mz_x64
| "set" "obfuscate" string ";" -> obfuscate
| "set" "sleep_mask" string ";" -> sleep_mask
| "set" "smartinject" string ";" -> smartinject
| "set" "stomppe" string ";" -> stomppe
| "set" "userwx" string ";" -> userwx
| "set" "compile_time" string ";" -> compile_time
| "set" "entry_point" string ";" -> entry_point
| "set" "module_x86" string ";" -> module_x86
| "set" "module_x64" string ";" -> module_x86
| "set" "image_size_x86" string ";" -> image_size_x86
| "set" "image_size_x64" string ";" -> image_size_x64

```

(continues on next page)

(continued from previous page)

```

| "set" "name" string ";" -> name
| "set" "rich_header" string ";" -> rich_header
| "set" "checksum" string ";" -> checksum
| "set" "syscall_method" string ";" -> syscall_method //
↳ introduced in Cobalt Strike 4.8

process_inject_options: "set" "allocator" string ";" -> allocator
| "set" "min_alloc" string ";" -> min_alloc
| "set" "starttrwx" string ";" -> starttrwx
| "set" "userwx" string ";" -> userwx
| "transform-x86" "{" stage_transform* "}" -> transform_x86
| "transform-x64" "{" stage_transform* "}" -> transform_x64
| "execute" "{" execute_options* "}" -> execute
| "disable" string ";" -> disable
| "set" "bof_allocator" string ";" -> bof_allocator //
↳ introduced in Cobalt Strike 4.7
| "set" "bof_reuse_memory" string ";" -> bof_reuse_memory //
↳ introduced in Cobalt Strike 4.7

execute_options: "CreateThread" string ";" -> createthread_special
| "CreateRemoteThread" string ";" -> createremotethread_special
| "CreateThread" ";" -> createthread
| "CreateRemoteThread" ";" -> createremotethread
| "NtQueueApcThread" ";" -> ntqueueapcthread
| "NtQueueApcThread-s" ";" -> ntqueueapcthread_s
| "RtlCreateUserThread" ";" -> rtlcreateuserthread
| "SetThreadContext" ";" -> setthreadcontext

postex_options: "set" "spawnto_x86" string ";" -> spawnto_x86
| "set" "spawnto_x64" string ";" -> spawnto_x64
| "set" "obfuscate" string ";" -> obfuscate
| "set" "pipename" string ";" -> pipename
| "set" "smartinject" string ";" -> smartinject
| "set" "amsi_disable" string ";" -> amsi_disable
| "set" "keylogger" string ";" -> keylogger
| "set" "thread_hint" string ";" -> thread_hint

dns_beacon_options: "set" "dns_idle" string ";" -> dns_idle
| "set" "dns_max_txt" string ";" -> dns_max_txt
| "set" "dns_sleep" string ";" -> dns_sleep
| "set" "dns_ttl" string ";" -> dns_ttl
| "set" "maxdns" string ";" -> maxdns
| "set" "dns_stager_prepend" string ";" -> dns_stager_prepend
| "set" "dns_stager_subhost" string ";" -> dns_stager_subhost
| "set" "beacon" string ";" -> beacon
| "set" "get_A" string ";" -> get_a
| "set" "get_AAAA" string ";" -> get_aaaa
| "set" "get_TXT" string ";" -> get_txt
| "set" "put_metadata" string ";" -> put_metadata
| "set" "put_output" string ";" -> put_output
| "set" "ns_response" string ";" -> ns_response
| "#" "dns_resolver" string ";" -> comment_dns_resolver

```

(continues on next page)

(continued from previous page)

```
header: string
string: STRING
variant: string

STRING: "\" /(.|\n)*? /(?<!\)(\\)*? / "\"

%import common.WS
%import common.SH_COMMENT
%import common.NEWLINE

%ignore WS
%ignore SH_COMMENT
%ignore NEWLINE
```


INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

d

- `dissect.cobaltstrike`, 49
- `dissect.cobaltstrike.artifact`, 49
- `dissect.cobaltstrike.beacon`, 50
- `dissect.cobaltstrike.c2`, 63
- `dissect.cobaltstrike.c2profile`, 69
- `dissect.cobaltstrike.c_c2`, 73
- `dissect.cobaltstrike.client`, 80
- `dissect.cobaltstrike.pcap`, 84
- `dissect.cobaltstrike.pe`, 86
- `dissect.cobaltstrike.utils`, 92
- `dissect.cobaltstrike.version`, 96
- `dissect.cobaltstrike.xordecode`, 97

Symbols

- `__eq__()` (*dissect.cobaltstrike.c_c2.BeaconMetadata method*), 80
- `__eq__()` (*dissect.cobaltstrike.c_c2.CallbackPacket method*), 80
- `__eq__()` (*dissect.cobaltstrike.c_c2.TaskPacket method*), 80
- `__getitem__()` (*dissect.cobaltstrike.utils.LRUDict method*), 95
- `__hash__()` (*dissect.cobaltstrike.c_c2.BeaconMetadata method*), 80
- `__hash__()` (*dissect.cobaltstrike.c_c2.CallbackPacket method*), 80
- `__hash__()` (*dissect.cobaltstrike.c_c2.TaskPacket method*), 80
- `__iter__()` (*dissect.cobaltstrike.c2profile.StringIterator method*), 70
- `__iter__()` (*dissect.cobaltstrike.pcap.BeaconCapture method*), 85
- `__name__` (*dissect.cobaltstrike.c2profile.C2Profile attribute*), 73
- `__name__` (*dissect.cobaltstrike.c2profile.ConfigBlock attribute*), 70
- `__name__` (*dissect.cobaltstrike.c2profile.DataTransformBlock attribute*), 71
- `__name__` (*dissect.cobaltstrike.c2profile.DnsBeaconBlock attribute*), 72
- `__name__` (*dissect.cobaltstrike.c2profile.ExecuteOptionsBlock attribute*), 72
- `__name__` (*dissect.cobaltstrike.c2profile.HttpConfigBlock attribute*), 71
- `__name__` (*dissect.cobaltstrike.c2profile.HttpGetBlock attribute*), 72
- `__name__` (*dissect.cobaltstrike.c2profile.HttpOptionsBlock attribute*), 71
- `__name__` (*dissect.cobaltstrike.c2profile.HttpPostBlock attribute*), 72
- `__name__` (*dissect.cobaltstrike.c2profile.HttpStagerBlock attribute*), 71
- `__name__` (*dissect.cobaltstrike.c2profile.PostExBlock attribute*), 72
- `__name__` (*dissect.cobaltstrike.c2profile.ProcessInjectBlock attribute*), 72
- `__name__` (*dissect.cobaltstrike.c2profile.StageBlock attribute*), 71
- `__name__` (*dissect.cobaltstrike.c2profile.StageTransformBlock attribute*), 71
- `__next__()` (*dissect.cobaltstrike.c2profile.StringIterator method*), 70
- `__repr__()` (*dissect.cobaltstrike.beacon.BeaconConfig method*), 62
- `__repr__()` (*dissect.cobaltstrike.version.BeaconVersion method*), 97
- `__repr__()` (*dissect.cobaltstrike.xordecode.XorEncodedFile method*), 98
- `__setitem__()` (*dissect.cobaltstrike.utils.LRUDict method*), 95
- `__str__()` (*dissect.cobaltstrike.c2profile.C2Profile method*), 73
- `__str__()` (*dissect.cobaltstrike.version.BeaconVersion method*), 97
- `_beacon_loop()` (*dissect.cobaltstrike.client.HttpBeaconClient method*), 83
- `_enable()` (*dissect.cobaltstrike.c2profile.ConfigBlock method*), 70
- `_header()` (*dissect.cobaltstrike.c2profile.ConfigBlock method*), 71
- `_initial_get_request()` (*dissect.cobaltstrike.client.HttpBeaconClient method*), 82
- `_initial_post_request()` (*dissect.cobaltstrike.client.HttpBeaconClient method*), 82
- `_pair()` (*dissect.cobaltstrike.c2profile.ConfigBlock method*), 70
- `_parameter()` (*dissect.cobaltstrike.c2profile.ConfigBlock method*), 71

A

- `add_step()` (*dissect.cobaltstrike.c2profile.DataTransformBlock method*), 71
- `add_termination()` (*dissect.cobaltstrike.c2profile.DataTransformBlock method*), 71

method), 71
 aes_key (dissect.cobaltstrike.c2.BeaconKeys attribute), 65
 aes_rand (dissect.cobaltstrike.c_c2.BeaconMetadata attribute), 79
 ansi_cp (dissect.cobaltstrike.c_c2.BeaconMetadata attribute), 79
 architecture (dissect.cobaltstrike.beacon.BeaconConfig attribute), 61
 ArtifactKitPayload (class in dissect.cobaltstrike.artifact), 49
 as_dict() (dissect.cobaltstrike.c2profile.C2Profile method), 73
 as_text() (dissect.cobaltstrike.c2profile.C2Profile method), 73

B

BeaconCallback (class in dissect.cobaltstrike.c_c2), 77
 BeaconCapture (class in dissect.cobaltstrike.pcap), 84
 BeaconCommand (class in dissect.cobaltstrike.c_c2), 74
 BeaconConfig (class in dissect.cobaltstrike.beacon), 58
 BeaconKeys (class in dissect.cobaltstrike.c2), 65
 BeaconMetadata (class in dissect.cobaltstrike.c_c2), 79
 BeaconProtocol (in module dissect.cobaltstrike.beacon), 56
 BeaconSetting (in module dissect.cobaltstrike.beacon), 56
 BeaconVersion (class in dissect.cobaltstrike.version), 96
 bid (dissect.cobaltstrike.c_c2.BeaconMetadata attribute), 79
 body (dissect.cobaltstrike.c2.HttpRequest attribute), 65
 body (dissect.cobaltstrike.c2.HttpResponse attribute), 65
 build_parser() (in module dissect.cobaltstrike.beacon), 62
 build_parser() (in module dissect.cobaltstrike.c2profile), 73
 build_parser() (in module dissect.cobaltstrike.client), 83

C

C2_DEF (in module dissect.cobaltstrike.c_c2), 78
 C2Data (class in dissect.cobaltstrike.c2), 64
 C2Http (class in dissect.cobaltstrike.c2), 66
 C2Packet (in module dissect.cobaltstrike.c2), 64
 c2packet_to_record() (in module dissect.cobaltstrike.c2), 66
 c2packet_to_record() (in module dissect.cobaltstrike.pcap), 84
 C2Profile (class in dissect.cobaltstrike.c2profile), 72
 c2profile_parser (in module dissect.cobaltstrike.c2profile), 70
 c2struct (in module dissect.cobaltstrike.c_c2), 79

callback (dissect.cobaltstrike.c_c2.CallbackPacket attribute), 80
 CALLBACK_ACCEPT (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_CHUNK_ALLOCATE (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_CHUNK_SEND (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_CLOSE (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 77
 CALLBACK_CONNECT (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 77
 CALLBACK_DEAD (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_ERROR (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_FILE (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 77
 CALLBACK_FILE_CLOSE (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 77
 CALLBACK_FILE_WRITE (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 77
 CALLBACK_HASHDUMP (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_JOBS (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_KEYSTROKES (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 77
 CALLBACK_NETVIEW (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_OUTPUT (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 77
 CALLBACK_OUTPUT_OEM (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_OUTPUT_UTF8 (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_PENDING (dissect.cobaltstrike.c_c2.BeaconCallback attribute), 78
 CALLBACK_PING (dissect.cobaltstrike.c_c2.BeaconCallback

<i>attribute</i>), 77			<i>tribute</i>), 64
CALLBACK_PIPE_CLOSE	(dis-	ClientC2Data (class in dissect.cobaltstrike.c2), 64	
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	command (dissect.cobaltstrike.c_c2.TaskPacket attribute),	
<i>tribute</i>), 77		80	
CALLBACK_PIPE_OPEN	(dis-	COMMAND_ARGUE_ADD	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>tribute</i>), 77		attribute), 76	
CALLBACK_PIPE_PING	(dis-	COMMAND_ARGUE_LIST	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>tribute</i>), 77		attribute), 76	
CALLBACK_PIPE_READ	(dis-	COMMAND_ARGUE_REMOVE	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>tribute</i>), 77		attribute), 76	
CALLBACK_PORTSCAN	(dis-	COMMAND_BLOCKDLLS	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>tribute</i>), 78		attribute), 77	
CALLBACK_POST_ERROR	(dis-	COMMAND_CANCEL_DOWNLOAD	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>tribute</i>), 77		attribute), 74	
CALLBACK_POST_REPLAY_ERROR	(dis-	COMMAND_CD (dissect.cobaltstrike.c_c2.BeaconCommand	
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	attribute), 74	
<i>tribute</i>), 78		COMMAND_CHECKIN	(dis-
CALLBACK_PROCESS_LIST	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	attribute), 74	
<i>tribute</i>), 78		COMMAND_CLOSE (dissect.cobaltstrike.c_c2.BeaconCommand	
CALLBACK_PWD (dissect.cobaltstrike.c_c2.BeaconCallback		attribute), 74	
<i>tribute</i>), 78		COMMAND_CONNECT	(dis-
CALLBACK_READ (dissect.cobaltstrike.c_c2.BeaconCallback		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>tribute</i>), 77		attribute), 74	
CALLBACK_SCREENSHOT	(dis-	COMMAND_DIE (dissect.cobaltstrike.c_c2.BeaconCommand	
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	attribute), 74	
<i>tribute</i>), 77		COMMAND_DLL_LOAD	(dis-
CALLBACK_SSH_STATUS	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	attribute), 76	
<i>tribute</i>), 78		COMMAND_DOWNLOAD	(dis-
CALLBACK_TOKEN_GETUID	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	attribute), 74	
<i>tribute</i>), 77		COMMAND_ELEVATE_POST	(dis-
CALLBACK_TOKEN_STOLEN	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCallback</i>	at-	attribute), 76	
<i>tribute</i>), 77		COMMAND_ELEVATE_PRE	(dis-
CallbackDebugMessage() (in module dis-		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.client</i>), 82		attribute), 75	
CallbackError() (in module dis-		COMMAND_EXECUTE	(dis-
<i>sect.cobaltstrike.client</i>), 82		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
CallbackOutputMessage() (in module dis-		attribute), 74	
<i>sect.cobaltstrike.client</i>), 82		COMMAND_EXECUTE_JOB	(dis-
CallbackPacket (class in dissect.cobaltstrike.c_c2), 80		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
catch_all() (dissect.cobaltstrike.client.HttpBeaconClient		attribute), 76	
method), 83		COMMAND_FILE_COPY	(dis-
catch_sigpipe() (in module dissect.cobaltstrike.utils),		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
94		attribute), 76	
checksum8() (in module dissect.cobaltstrike.utils), 95		COMMAND_FILE_DRIVES	(dis-
ciphertext (dissect.cobaltstrike.c2.EncryptedPacket at-		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	

<i>attribute</i>), 75		<i>attribute</i>), 76	
COMMAND_FILE_LIST	(dis-	COMMAND_JOB_REGISTER_MSGMODE	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 75		<i>attribute</i>), 77	
COMMAND_FILE_MKDIR	(dis-	COMMAND_JOB_SPAWN_TOKEN_X64	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 75		<i>attribute</i>), 76	
COMMAND_FILE_MOVE	(dis-	COMMAND_JOB_SPAWN_TOKEN_X86	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 76		<i>attribute</i>), 76	
COMMAND_FILE_RM	(dis-	COMMAND_JOB_SPAWN_X64	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 75		<i>attribute</i>), 76	
COMMAND_GETPRIVS	(dis-	COMMAND_JOB_SPAWN_X86	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 76		<i>attribute</i>), 76	
COMMAND_INJECT_PID	(dis-	COMMAND_JOBS (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 75	
<i>attribute</i>), 74		COMMAND_KEYLOG_START	(dis-
COMMAND_INJECT_PID_PING	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 74	
<i>attribute</i>), 75		COMMAND_KEYLOG_STOP	(dis-
COMMAND_INJECT_PING	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 74	
<i>attribute</i>), 74		COMMAND_LISTEN	(dis-
COMMAND_INJECT_POWERSHELLX64_PID	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 74	
<i>attribute</i>), 76		COMMAND_LOGINUSER	(dis-
COMMAND_INJECT_POWERSHELLX86_PID	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 75	
<i>attribute</i>), 76		COMMAND_LSOCKET_BIND	(dis-
COMMAND_INJECTX64_PID	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 75	
<i>attribute</i>), 75		COMMAND_LSOCKET_BIND_LOCALHOST	(dis-
COMMAND_INJECTX64_PID_PING	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 77	
<i>attribute</i>), 75		COMMAND_LSOCKET_CLOSE	(dis-
COMMAND_INJECTX64_PING	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 75	
<i>attribute</i>), 76		COMMAND_LSOCKET_TCPPIVOT	(dis-
COMMAND_INLINE_EXECUTE	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 76	
<i>attribute</i>), 77		COMMAND_NOOP (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_INLINE_EXECUTE_OBJECT	(dis-	<i>attribute</i>), 74	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_PAUSE (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 77		<i>attribute</i>), 75	
COMMAND_JOB_KILL	(dis-	COMMAND_PIPE_CLOSE	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 75		<i>attribute</i>), 75	
COMMAND_JOB_REGISTER	(dis-	COMMAND_PIPE_OPEN_EXPLICIT	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 75		<i>attribute</i>), 76	
COMMAND_JOB_REGISTER_IMPERSONATE	(dis-	COMMAND_PIPE_REOPEN	(dis-
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	

<i>attribute</i>), 75		COMMAND_SPAWN_PROC_X86	(dis-
COMMAND_PIPE_ROUTE	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 75		<i>attribute</i>), 74	
COMMAND_PPID (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_SPAWN_TOKEN_X64	(dis-
<i>attribute</i>), 76		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_PS_KILL	(dis-	<i>attribute</i>), 76	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_SPAWN_TOKEN_X86	(dis-
<i>attribute</i>), 75		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_PS_LIST	(dis-	<i>attribute</i>), 76	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_SPAWNAS_X64	(dis-
<i>attribute</i>), 75		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_PSH_HOST_TCP	(dis-	<i>attribute</i>), 77	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_SPAWNAS_X86	(dis-
<i>attribute</i>), 76		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_PSH_IMPORT	(dis-	<i>attribute</i>), 77	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_SPAWNU_X64	(dis-
<i>attribute</i>), 75		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_PWD (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 77	
<i>attribute</i>), 75		COMMAND_SPAWNU_X86	(dis-
COMMAND_REG_QUERY	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 77	
<i>attribute</i>), 76		COMMAND_SPAWNX64	(dis-
COMMAND_RUN_INJECT_X64	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 75	
<i>attribute</i>), 77		COMMAND_STAGE_PAYLOAD	(dis-
COMMAND_RUN_INJECT_X86	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 75	
<i>attribute</i>), 77		COMMAND_STAGE_PAYLOAD_SMB	(dis-
COMMAND_RUN_UNDER_PID	(dis-	<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 75	
<i>attribute</i>), 76		COMMAND_STEAL_TOKEN	(dis-
COMMAND_RUNAS (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 75		<i>attribute</i>), 75	
COMMAND_SEND (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_TCP_CONNECT	(dis-
<i>attribute</i>), 74		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_SETENV	(dis-	<i>attribute</i>), 76	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_TIMESTOMP	(dis-
<i>attribute</i>), 76		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_SHELL (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>		<i>attribute</i>), 75	
<i>attribute</i>), 74		COMMAND_TOKEN_GETUID	(dis-
COMMAND_SLEEP (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
<i>attribute</i>), 74		<i>attribute</i>), 75	
COMMAND_SPAWN (<i>dissect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_TOKEN_REV2SELF	(dis-
<i>attribute</i>), 74		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_SPAWN_POWERSHELLX64	(dis-	<i>attribute</i>), 75	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_UPLOAD	(dis-
<i>attribute</i>), 76		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_SPAWN_POWERSHELLX86	(dis-	<i>attribute</i>), 74	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_UPLOAD_CONTINUE	(dis-
<i>attribute</i>), 76		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
COMMAND_SPAWN_PROC_X64	(dis-	<i>attribute</i>), 76	
<i>sect.cobaltstrike.c_c2.BeaconCommand</i>		COMMAND_WEBSERVER_LOCAL	(dis-
<i>attribute</i>), 76		<i>sect.cobaltstrike.c_c2.BeaconCommand</i>	
		<i>attribute</i>), 75	

COMPUTERNAME_TEMPLATES (in module *dissect.cobaltstrike.client*), 82
 config_block (*dissect.cobaltstrike.beacon.BeaconConfig* attribute), 61
 ConfigBlock (class in *dissect.cobaltstrike.c2profile*), 70
 counter (*dissect.cobaltstrike.c_c2.CallbackPacket* attribute), 80
 createremotethread (in *dissect.cobaltstrike.c2profile.ExecuteOptionsBlock* attribute), 72
 createremotethread_special (in *dissect.cobaltstrike.c2profile.ExecuteOptionsBlock* attribute), 72
 createthread (*dissect.cobaltstrike.c2profile.ExecuteOptionsBlock* attribute), 72
 createthread_special (in *dissect.cobaltstrike.c2profile.ExecuteOptionsBlock* attribute), 72
 CryptoScheme (in module *dissect.cobaltstrike.beacon*), 56
 CS_DEF (in module *dissect.cobaltstrike.beacon*), 52
 cs_struct (in module *dissect.cobaltstrike.beacon*), 56
D
 data (*dissect.cobaltstrike.c_c2.CallbackPacket* attribute), 80
 data (*dissect.cobaltstrike.c_c2.TaskPacket* attribute), 80
 DataTransformBlock (class in *dissect.cobaltstrike.c2profile*), 71
 date (*dissect.cobaltstrike.version.BeaconVersion* attribute), 96
 decrypt_data() (in module *dissect.cobaltstrike.c2*), 68
 decrypt_metadata() (in module *dissect.cobaltstrike.c2*), 67
 decrypt_packet() (in module *dissect.cobaltstrike.c2*), 68
 DEFAULT_AES_IV (*dissect.cobaltstrike.c2.BeaconKeys* attribute), 65
 DEFAULT_XOR_KEYS (in module *dissect.cobaltstrike.beacon*), 56
 DeprecatedBeaconSetting (in module *dissect.cobaltstrike.beacon*), 56
 derive_aes_hmac_keys() (in module *dissect.cobaltstrike.c2*), 68
 dissect.cobaltstrike module, 49
 dissect.cobaltstrike.artifact module, 49
 dissect.cobaltstrike.beacon module, 50
 dissect.cobaltstrike.c2 module, 63
 dissect.cobaltstrike.c2profile module, 69
 dissect.cobaltstrike.c_c2 module, 73
 dissect.cobaltstrike.client module, 80
 dissect.cobaltstrike.pcap module, 84
 dissect.cobaltstrike.pe module, 86
 dissect.cobaltstrike.utils module, 92
 dissect.cobaltstrike.version module, 96
 dissect.cobaltstrike.xordecode module, 97
 DnsBeaconBlock (class in *dissect.cobaltstrike.c2profile*), 72
 domain_uri_pairs (*dissect.cobaltstrike.beacon.BeaconConfig* property), 60
 domains (*dissect.cobaltstrike.beacon.BeaconConfig* property), 60
 DOSHEADER_X64 (in module *dissect.cobaltstrike.pe*), 89
 DOSHEADER_X86 (in module *dissect.cobaltstrike.pe*), 89
 dumps() (*dissect.cobaltstrike.c2.EncryptedPacket* method), 64
E
 enable_replib_c2() (in module *dissect.cobaltstrike.c2*), 65
 enable_replib_cstruct() (in module *dissect.cobaltstrike.utils*), 95
 enable_replib_flow_record() (in module *dissect.cobaltstrike.utils*), 95
 encrypt_data() (in module *dissect.cobaltstrike.c2*), 68
 encrypt_metadata() (in module *dissect.cobaltstrike.c2*), 67
 encrypt_packet() (in module *dissect.cobaltstrike.c2*), 69
 EncryptedPacket (class in *dissect.cobaltstrike.c2*), 64
 EOF_SHELLCODE_MARKER (*dissect.cobaltstrike.xordecode.XorEncodedFile* attribute), 98
 epoch (*dissect.cobaltstrike.c_c2.TaskPacket* attribute), 80
 ExecuteOptionsBlock (class in *dissect.cobaltstrike.c2profile*), 72
F
 find_architecture() (in module *dissect.cobaltstrike.pe*), 91
 find_beacon_config_bytes() (in module *dissect.cobaltstrike.beacon*), 57
 find_compile_stamps() (in module *dissect.cobaltstrike.pe*), 90

find_magic_mz() (in module dissect.cobaltstrike.pe), 90
 find_magic_pe() (in module dissect.cobaltstrike.pe), 90
 find_mz_offset() (in module dissect.cobaltstrike.pe), 89
 find_stage_prepend_append() (in module dissect.cobaltstrike.pe), 91
 find_staged_beacon() (dissect.cobaltstrike.pcap.BeaonCapture method), 85
 FIRST_NAMES (in module dissect.cobaltstrike.client), 81
 flag (dissect.cobaltstrike.c_c2.BeaonMetadata attribute), 79
 from_aes_rand() (dissect.cobaltstrike.c2.BeaonKeys class method), 65
 from_beacon_config() (dissect.cobaltstrike.c2profile.C2Profile class method), 73
 from_beacon_metadata() (dissect.cobaltstrike.c2.BeaonKeys class method), 65
 from_bytes() (dissect.cobaltstrike.beacon.BeaonConfig class method), 62
 from_execute_list() (dissect.cobaltstrike.c2profile.ExecuteOptionsBlock class method), 72
 from_file() (dissect.cobaltstrike.beacon.BeaonConfig class method), 61
 from_file() (dissect.cobaltstrike.xordecode.XorEncodedFile class method), 98
 from_max_setting_enum() (dissect.cobaltstrike.version.BeaonVersion class method), 97
 from_path() (dissect.cobaltstrike.beacon.BeaonConfig class method), 62
 from_path() (dissect.cobaltstrike.c2profile.C2Profile class method), 73
 from_path() (dissect.cobaltstrike.xordecode.XorEncodedFile class method), 98
 from_pe_export_stamp() (dissect.cobaltstrike.version.BeaonVersion class method), 97
 from_text() (dissect.cobaltstrike.c2profile.C2Profile class method), 73
G
 get_handlers() (dissect.cobaltstrike.client.HttpBeaconClient method), 83
 get_sleep_time() (dissect.cobaltstrike.client.HttpBeaconClient method), 82
 get_task() (dissect.cobaltstrike.client.HttpBeaconClient method), 83
 get_transform_for_http() (dissect.cobaltstrike.c2.C2Http method), 66
 grouper() (in module dissect.cobaltstrike.beacon), 58
H
 handle() (dissect.cobaltstrike.client.HttpBeaconClient method), 83
 has_next() (dissect.cobaltstrike.c2profile.StringIterator method), 70
 header (dissect.cobaltstrike.c2profile.HttpConfigBlock attribute), 71
 header (dissect.cobaltstrike.c2profile.HttpOptionsBlock attribute), 71
 headers (dissect.cobaltstrike.c2.HttpRequest attribute), 65
 headers (dissect.cobaltstrike.c2.HttpResponse attribute), 65
 hints (dissect.cobaltstrike.artifact.ArtifactKitPayload attribute), 50
 hmac_key (dissect.cobaltstrike.c2.BeaonKeys attribute), 65
 HttpBeaconClient (class in dissect.cobaltstrike.client), 82
 HttpConfigBlock (class in dissect.cobaltstrike.c2profile), 71
 HttpDataTransform (class in dissect.cobaltstrike.c2), 66
 HttpGetBlock (class in dissect.cobaltstrike.c2profile), 72
 HttpOptionsBlock (class in dissect.cobaltstrike.c2profile), 71
 HttpPostBlock (class in dissect.cobaltstrike.c2profile), 72
 HttpRequest (class in dissect.cobaltstrike.c2), 65
 HttpResponse (class in dissect.cobaltstrike.c2), 65
 HttpStagerBlock (class in dissect.cobaltstrike.c2profile), 71
I
 id (dissect.cobaltstrike.c2.C2Data attribute), 64
 info (dissect.cobaltstrike.c_c2.BeaonMetadata attribute), 80
 init_kwargs() (dissect.cobaltstrike.c2profile.ConfigBlock method), 70
 InjectAllocator (in module dissect.cobaltstrike.beacon), 56
 InjectExecutor (in module dissect.cobaltstrike.beacon), 56
 ip (dissect.cobaltstrike.c_c2.BeaonMetadata attribute), 79
 is_stager_x64() (in module dissect.cobaltstrike.utils), 95

[is_stager_x86\(\)](#) (in module *dissect.cobaltstrike.utils*), 95
[is_trial](#) (*dissect.cobaltstrike.beacon.BeaconConfig* property), 61
[iter_artifactkit_payloads\(\)](#) (in module *dissect.cobaltstrike.artifact*), 50
[iter_beacon_config_blocks\(\)](#) (in module *dissect.cobaltstrike.beacon*), 57
[iter_encrypted_packets\(\)](#) (*dissect.cobaltstrike.c2.ClientC2Data* method), 65
[iter_encrypted_packets\(\)](#) (*dissect.cobaltstrike.c2.ServerC2Data* method), 64
[iter_find_needle\(\)](#) (in module *dissect.cobaltstrike.utils*), 95
[iter_nonce_offsets\(\)](#) (in module *dissect.cobaltstrike.xordecode*), 97
[iter_parse_pcap\(\)](#) (*dissect.cobaltstrike.pcap.BeaconCapture* method), 85
[iter_recover_http\(\)](#) (*dissect.cobaltstrike.c2.C2HttpRequest* method), 67
[iter_settings\(\)](#) (in module *dissect.cobaltstrike.beacon*), 57
[iv](#) (*dissect.cobaltstrike.c2.BeaconKeys* attribute), 65

J

[jitter](#) (*dissect.cobaltstrike.beacon.BeaconConfig* property), 61

K

[killdate](#) (*dissect.cobaltstrike.beacon.BeaconConfig* property), 60

L

[LAST_NAMES](#) (in module *dissect.cobaltstrike.client*), 82
[log_task\(\)](#) (in module *dissect.cobaltstrike.client*), 82
[logger](#) (in module *dissect.cobaltstrike.artifact*), 49
[logger](#) (in module *dissect.cobaltstrike.beacon*), 52
[logger](#) (in module *dissect.cobaltstrike.c2*), 64
[logger](#) (in module *dissect.cobaltstrike.c2profile*), 70
[logger](#) (in module *dissect.cobaltstrike.client*), 81
[logger](#) (in module *dissect.cobaltstrike.pcap*), 84
[logger](#) (in module *dissect.cobaltstrike.pe*), 86
[logger](#) (in module *dissect.cobaltstrike.xordecode*), 97
[LRUDict](#) (class in *dissect.cobaltstrike.utils*), 95

M

[magic](#) (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79
[main\(\)](#) (in module *dissect.cobaltstrike.artifact*), 50
[main\(\)](#) (in module *dissect.cobaltstrike.beacon*), 62

[main\(\)](#) (in module *dissect.cobaltstrike.c2profile*), 73
[main\(\)](#) (in module *dissect.cobaltstrike.client*), 84
[main\(\)](#) (in module *dissect.cobaltstrike.pcap*), 86
[main\(\)](#) (in module *dissect.cobaltstrike.xordecode*), 99
[make_byte_list\(\)](#) (in module *dissect.cobaltstrike.beacon*), 57
[MAX_ENUM_TO_VERSION](#) (in module *dissect.cobaltstrike.version*), 96
[max_setting_enum](#) (*dissect.cobaltstrike.beacon.BeaconConfig* property), 59
[metadata](#) (*dissect.cobaltstrike.c2.C2Data* attribute), 64
[method](#) (*dissect.cobaltstrike.c2.HttpRequest* attribute), 65
[module](#)
 dissect.cobaltstrike, 49
 dissect.cobaltstrike.artifact, 49
 dissect.cobaltstrike.beacon, 50
 dissect.cobaltstrike.c2, 63
 dissect.cobaltstrike.c2profile, 69
 dissect.cobaltstrike.c_c2, 73
 dissect.cobaltstrike.client, 80
 dissect.cobaltstrike.pcap, 84
 dissect.cobaltstrike.pe, 86
 dissect.cobaltstrike.utils, 92
 dissect.cobaltstrike.version, 96
 dissect.cobaltstrike.xordecode, 97

N

[namedtuple_reprlib_repr\(\)](#) (in module *dissect.cobaltstrike.utils*), 95
[netbios_decode\(\)](#) (in module *dissect.cobaltstrike.utils*), 93
[netbios_encode\(\)](#) (in module *dissect.cobaltstrike.utils*), 93
[next\(\)](#) (*dissect.cobaltstrike.c2profile.StringIterator* method), 70
[ntqueueapcthread](#) (*dissect.cobaltstrike.c2profile.ExecuteOptionsBlock* attribute), 72
[ntqueueapcthread_s](#) (*dissect.cobaltstrike.c2profile.ExecuteOptionsBlock* attribute), 72
[null_terminated_bytes\(\)](#) (in module *dissect.cobaltstrike.beacon*), 58
[null_terminated_str\(\)](#) (in module *dissect.cobaltstrike.beacon*), 58

O

[oem_cp](#) (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79
[offset](#) (*dissect.cobaltstrike.artifact.ArtifactKitPayload* attribute), 49
[output](#) (*dissect.cobaltstrike.c2.C2Data* attribute), 64

P

- p16 (in module *dissect.cobaltstrike.utils*), 94
- p16be (in module *dissect.cobaltstrike.utils*), 94
- p32 (in module *dissect.cobaltstrike.utils*), 94
- p32be (in module *dissect.cobaltstrike.utils*), 94
- p64 (in module *dissect.cobaltstrike.utils*), 94
- p64be (in module *dissect.cobaltstrike.utils*), 94
- p8 (in module *dissect.cobaltstrike.utils*), 94
- pack() (in module *dissect.cobaltstrike.utils*), 94
- pack_be (in module *dissect.cobaltstrike.utils*), 94
- packet_to_record() (in module *dissect.cobaltstrike.pcap*), 84
- PacketRecord (in module *dissect.cobaltstrike.pcap*), 84
- pad() (in module *dissect.cobaltstrike.c2*), 68
- parameter (*dissect.cobaltstrike.c2profile.HttpOptionsBlock* attribute), 71
- params (*dissect.cobaltstrike.c2.HttpRequest* attribute), 65
- parse_commandline_options() (in module *dissect.cobaltstrike.client*), 83
- parse_execute_list() (in module *dissect.cobaltstrike.beacon*), 58
- parse_gargle() (in module *dissect.cobaltstrike.beacon*), 58
- parse_pivot_frame() (in module *dissect.cobaltstrike.beacon*), 58
- parse_process_injection_transform_steps() (in module *dissect.cobaltstrike.beacon*), 58
- parse_raw_http() (in module *dissect.cobaltstrike.c2*), 66
- parse_recover_binary() (in module *dissect.cobaltstrike.beacon*), 58
- parse_transform_binary() (in module *dissect.cobaltstrike.beacon*), 58
- payload (*dissect.cobaltstrike.artifact.ArtifactKitPayload* attribute), 50
- pe_compile_stamp (*dissect.cobaltstrike.beacon.BeaconConfig* attribute), 61
- PE_DEF (in module *dissect.cobaltstrike.pe*), 86
- pe_export_stamp (*dissect.cobaltstrike.beacon.BeaconConfig* attribute), 61
- PE_EXPORT_STAMP_TO_VERSION (in module *dissect.cobaltstrike.version*), 96
- pestruct (in module *dissect.cobaltstrike.pe*), 89
- pid (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79
- port (*dissect.cobaltstrike.beacon.BeaconConfig* property), 60
- port (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79
- PostExBlock (class in *dissect.cobaltstrike.c2profile*), 72
- print_settings() (*dissect.cobaltstrike.client.HttpBeaconClient* method), 83
- PROCESS_NAMES (in module *dissect.cobaltstrike.client*), 82
- ProcessInjectBlock (class in *dissect.cobaltstrike.c2profile*), 71
- properties (*dissect.cobaltstrike.c2profile.C2Profile* property), 73
- protocol (*dissect.cobaltstrike.beacon.BeaconConfig* property), 60
- ProxyServer (in module *dissect.cobaltstrike.beacon*), 56
- ptr_gmh (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79
- ptr_gpa (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79
- ptr_x64 (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79
- public_key (*dissect.cobaltstrike.beacon.BeaconConfig* property), 61

R

- raise_for_signature() (*dissect.cobaltstrike.c2.EncryptedPacket* method), 64
- random_computer_name() (in module *dissect.cobaltstrike.client*), 82
- random_internal_ip() (in module *dissect.cobaltstrike.client*), 82
- random_process_name() (in module *dissect.cobaltstrike.client*), 82
- random_stager_uri() (in module *dissect.cobaltstrike.utils*), 95
- random_username_name() (in module *dissect.cobaltstrike.client*), 82
- random_windows_ver() (in module *dissect.cobaltstrike.client*), 82
- raw_http_from_packet() (in module *dissect.cobaltstrike.pcap*), 84
- raw_settings (*dissect.cobaltstrike.beacon.BeaconConfig* property), 59
- raw_settings_by_index (*dissect.cobaltstrike.beacon.BeaconConfig* property), 59
- read() (*dissect.cobaltstrike.xordecode.XorEncodedFile* method), 99
- read_nonce() (*dissect.cobaltstrike.xordecode.XorEncodedFile* method), 99
- reason (*dissect.cobaltstrike.c2.HttpResponse* attribute), 65
- recover() (*dissect.cobaltstrike.c2.HttpDataTransform* method), 66
- REGEX_VERSION (*dissect.cobaltstrike.version.BeaconVersion* attribute), 96

- register_task() (dissect.cobaltstrike.client.HttpBeaconClient method), 82
- request (dissect.cobaltstrike.c2.HttpResponse attribute), 65
- retain_file_offset() (in module dissect.cobaltstrike.utils), 94
- rtlcreateuserthread (dissect.cobaltstrike.c2profile.ExecuteOptionsBlock attribute), 72
- run() (dissect.cobaltstrike.client.HttpBeaconClient method), 82
- ## S
- seek() (dissect.cobaltstrike.xordecode.XorEncodedFile method), 99
- send_callback() (dissect.cobaltstrike.client.HttpBeaconClient method), 83
- ServerC2Data (class in dissect.cobaltstrike.c2), 64
- set_config_block() (dissect.cobaltstrike.c2profile.ConfigBlock method), 70
- set_non_empty_config_block() (dissect.cobaltstrike.c2profile.ConfigBlock method), 70
- set_option() (dissect.cobaltstrike.c2profile.C2Profile method), 73
- set_option() (dissect.cobaltstrike.c2profile.ConfigBlock method), 70
- setthreadcontext (dissect.cobaltstrike.c2profile.ExecuteOptionsBlock attribute), 72
- Setting (in module dissect.cobaltstrike.beacon), 56
- setting_enums (dissect.cobaltstrike.beacon.BeaconConfig property), 59
- SETTING_TO_PRETTYFUNC (in module dissect.cobaltstrike.beacon), 58
- settings (dissect.cobaltstrike.beacon.BeaconConfig property), 59
- settings_by_index (dissect.cobaltstrike.beacon.BeaconConfig property), 60
- settings_map() (dissect.cobaltstrike.beacon.BeaconConfig method), 62
- settings_tuple (dissect.cobaltstrike.beacon.BeaconConfig attribute), 61
- SettingsType (in module dissect.cobaltstrike.beacon), 56
- sha256sum_pubkey() (in module dissect.cobaltstrike.beacon), 58
- signature (dissect.cobaltstrike.c2.EncryptedPacket attribute), 64
- size (dissect.cobaltstrike.artifact.ArtifactKitPayload attribute), 49
- size (dissect.cobaltstrike.c_c2.BeaconMetadata attribute), 79
- size (dissect.cobaltstrike.c_c2.CallbackPacket attribute), 80
- size (dissect.cobaltstrike.c_c2.TaskPacket attribute), 80
- sleeptime (dissect.cobaltstrike.beacon.BeaconConfig property), 61
- StageBlock (class in dissect.cobaltstrike.c2profile), 71
- StageTransformBlock (class in dissect.cobaltstrike.c2profile), 71
- status (dissect.cobaltstrike.c2.HttpResponse attribute), 65
- string_token_to_bytes() (in module dissect.cobaltstrike.c2profile), 70
- StringIterator (class in dissect.cobaltstrike.c2profile), 70
- strrep (dissect.cobaltstrike.c2profile.StageTransformBlock attribute), 71
- submit_uri (dissect.cobaltstrike.beacon.BeaconConfig property), 60
- ## T
- TaskPacket (class in dissect.cobaltstrike.c_c2), 80
- tell() (dissect.cobaltstrike.xordecode.XorEncodedFile method), 99
- total_size (dissect.cobaltstrike.c_c2.TaskPacket attribute), 80
- transform() (dissect.cobaltstrike.c2.HttpDataTransform method), 66
- TransformStep (in module dissect.cobaltstrike.beacon), 56
- TransformStep (in module dissect.cobaltstrike.c2), 64
- tree (dissect.cobaltstrike.c2profile.DataTransformBlock property), 71
- tuple (dissect.cobaltstrike.version.BeaconVersion attribute), 96
- typedef_for_enum() (in module dissect.cobaltstrike.c_c2), 79
- ## U
- u16 (in module dissect.cobaltstrike.utils), 94
- u16be (in module dissect.cobaltstrike.utils), 94
- u32 (in module dissect.cobaltstrike.utils), 94
- u32be (in module dissect.cobaltstrike.utils), 94
- u64 (in module dissect.cobaltstrike.utils), 94
- u64be (in module dissect.cobaltstrike.utils), 94
- u8 (in module dissect.cobaltstrike.utils), 94
- unpack() (in module dissect.cobaltstrike.utils), 94
- unpack_be (in module dissect.cobaltstrike.utils), 94
- uri (dissect.cobaltstrike.c2.HttpRequest attribute), 65

`uris` (*dissect.cobaltstrike.beacon.BeaconConfig* property), 60

V

`value_to_string()` (in module *dissect.cobaltstrike.c2profile*), 70

`ver_build` (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79

`ver_major` (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79

`ver_minor` (*dissect.cobaltstrike.c_c2.BeaconMetadata* attribute), 79

`version` (*dissect.cobaltstrike.beacon.BeaconConfig* property), 61

`version` (*dissect.cobaltstrike.version.BeaconVersion* attribute), 96

`version_only` (*dissect.cobaltstrike.version.BeaconVersion* property), 96

`version_string` (*dissect.cobaltstrike.version.BeaconVersion* property), 96

W

`watermark` (*dissect.cobaltstrike.beacon.BeaconConfig* property), 61

X

`xor()` (in module *dissect.cobaltstrike.utils*), 93

`xorencoded` (*dissect.cobaltstrike.beacon.BeaconConfig* attribute), 61

`XorEncodedFile` (class in *dissect.cobaltstrike.xordecode*), 98

`xorkey` (*dissect.cobaltstrike.artifact.ArtifactKitPayload* attribute), 50

`xorkey` (*dissect.cobaltstrike.beacon.BeaconConfig* attribute), 61