Cobalt Strike Configuration Extractor and Parser

With each passing month and each new ransomware case, the consultants at Stroz Friedberg Incident Response are finding Cobalt Strike Beacons to be the norm for persistence, lateral movement, and exfiltration rather than the exception. Finding command and control ("C2") servers and shutting off those connections is imperative before starting the rebuilding phase to help ensure threat actors can't regain entry into the network. Due to the large volume of Beacons in our casework, and given the opportunity to dive deep into this advanced post-exploitation framework, we set out to write a high-quality library and associated set of command line tools for extracting and parsing configuration data from Cobalt Strike Beacons to aid in accelerating response times. Today we are open sourcing the library and scripts, as well as extensive Beacon and library documentation, for the broader DFIR community to use and collaborate on with us!

Getting Started

The README contains detailed instructions for using pip to install the package from PyPI. There are two command line scripts that come bundled with the package:

1. csce: Parse Beacon configuration data to JSON
2. list-cs-settings: Attempt to discover, extract, and parse all possible Beacon configuration data by brute force

The first script, csce (Cobalt Strike Configuration Extractor), is intended for daily use to extract and parse Beacon configuration data and is the one most will likely be interested in. list-cs-settings is designed for those who want to conduct research on Beacon configurations by attempting to detect setting types by brute force. Both the library and (thus) the scripts currently support parsing Beacon PE files (EXE/DLL) and memory dumps from systems running a Beacon. As an example, you can parse a Beacon DLL sample using csce like this:

```
> csce --pretty path/to/beacon.{exe,dll,bin,dmp}
```

This will pretty-print Beacon configuration data as JSON (assuming the input file is a Beacon) in a structure that closely mimics the Malleable C2 Profile of the Team Server the Beacon was generated from. The output shown below is from an HTTP Beacon we recovered during one of our cases (with all sensitive IOCs redacted/replaced):

```
> csce --pretty path/to/beacon.dll
{
  "beacontype": [
    "HTTPS"
  ],
  "sleeptime": 60000,
  "jitter": 0,
  "maxgetsize": 1048576,
  "spawnto": "AAAAAAAAAAAAAAAAAAAAAA==",
  "license_id": 1111111111,
  "cfg_caution": false,
  "kill_date": null,
  "server": {
    "hostname": "55.55.55.55",
    "port": 443,
    "publickey": "REDACTED"
  },
  "host_header": null,
  "useragent_header": "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64;
Trident/6.0; Touch)",
  "http-get": {
    "uri": "/aa",
    "verb": "GET",
    "client": {
      "headers": [],
      "metadata": [
        "base64",
        "header 'Cookie'"
      ]
    },
    "server": {
      "output": [
        "print"
      ]
    }
  },
  "http-post": {
    "uri": "/endpoint.php",
    "verb": "POST",
    "client": {
      "headers": [
        "Content-Type: application/octet-stream"
      ],
      "id": [
        "parameter 'id'"
      ],
      "output": [
        "print"
      ]
    }
  },
  "tcp_frame_header": null,
  "crypto_scheme": 0,
  "proxy": {
    "type": null,
    "username": null,
    "password": null,
    "behavior": "Use IE settings"
  },
  "http_post_chunk": 0,
  "uses_cookies": true,
  "post-ex": {
    "spawnto_x86": "%windir%\\syswow64\\rundll32.exe",
    "spawnto_x64": "%windir%\\sysnative\\rundll32.exe"
  },
  "process-inject": {
    "allocator": "VirtualAllocEx",
    "execute": [
      "CreateThread",
      "SetThreadContext",
      "CreateRemoteThread",
      "RtlCreateUserThread"
    ],
    "min_alloc": 0,
    "startrwx": true,
    "stub": "REDACTED",
    "transform-x86": null,
    "transform-x64": null,
    "userwx": true
  },
  "dns-beacon": {
    "dns_idle": "0.0.0.0",
    "dns_sleep": 0,
    "maxdns": 255,
    "beacon": null,
    "get_A": null,
    "get_AAAA": null,
    "get_TXT": null,
    "put_metadata": null,
    "put_output": null
  },
  "pipename": "",
  "smb_frame_header": null,
  "stage": {
    "cleanup": false
  },
  "ssh": {
    "hostname": null,
    "port": null,
    "username": null,
    "password": null,
    "privatekey": null
  }
}
```

Beacons do not contain full Malleable C2 Profiles for space and OPSEC reasons, but as is shown above critical sections can be reconstructed from Beacon configuration data including C2 IP addresses and domains, beaconing behavior and payload settings, and more. These data points can be very important during the containment phase of an investigation. For Beacon-specific settings that aren't specified in the Malleable C2 Profile, we did our best to group items together where it makes sense (i.e., the ssh section).

Why Use It?

There are few primary reasons why we recommend the use of this library and these scripts over other open source options:

1. Fairly well tested against generated samples and samples discovered in the wild.
2. Relatively fast and will likely get faster over time.
3. To our knowledge, the only library/scripts that attempt to reconstruct the Malleable C2 Profile from Beacon configuration data.

Where are the Tests?

You might be thinking "they said it's fairly well tested, but I don't see any test samples in the GitHub repository. What Gives?" Many of the samples we use for testing were gathered from client environments, and thus we cannot publish them to GitHub. Moreover, we think it's generally a bad idea to publish malware to a repository others may download to computers not configured for malware sandboxing. However, we will likely add unit tests for the core parser logic in the future, and we currently test all the examples present in the documentation for correctness (shoutout to doctest).

Our Hope and Vision

These tools have helped speed up our own investigations, and our hope is that by sharing them with the community they will do the same for others. Our long-term vision is to work collectively with DFIR practitioners and researchers to help make these tools the best available for parsing Cobalt Strike Beacons, and raise the bar for threat actors going forward.

Honorable Mentions

Many others in the DFIR community have performed in-depth research on Cobalt Strike Beacons. We want to recognize those who published findings that helped us, directly or indirectly, along the way:

1. SentinelOne: Cobalt Strike Beacon parser
2. RomanEmelyanov: Cobalt Strike forensics scripts
3. Apr4h: Scan Windows processes for Cobalt Strike Beacons
4. JPCert: Cobalt Strike Volatility plugin
5. Didier Stevens: Cobalt Strike Beacon parser

Author: Noah Rubin
August 27, 2021

Cyber security services offered by Stroz Friedberg Inc. and its affiliates. Insurance products and services offered by Aon Risk Insurance Services West, Inc., Aon Risk Services Central, Inc., Aon Risk Services Northeast, Inc., Aon Risk Services Southwest, Inc., and Aon Risk Services, Inc. of Florida and their licensed affiliates. Aon UK Limited is authorised and regulated by the Financial Conduct Authority in respect of insurance distribution services. FP.AGRC.238.JJ The following products or services are not regulated by the Financial Conduct Authority:
• Cyber risk services provided by Aon UK Limited and its affiliates
• Cyber security services provided by Stroz Friedberg Limited and its affiliates.