The default: 63 6f 62 61 6c 74 strike

By Intel Operator

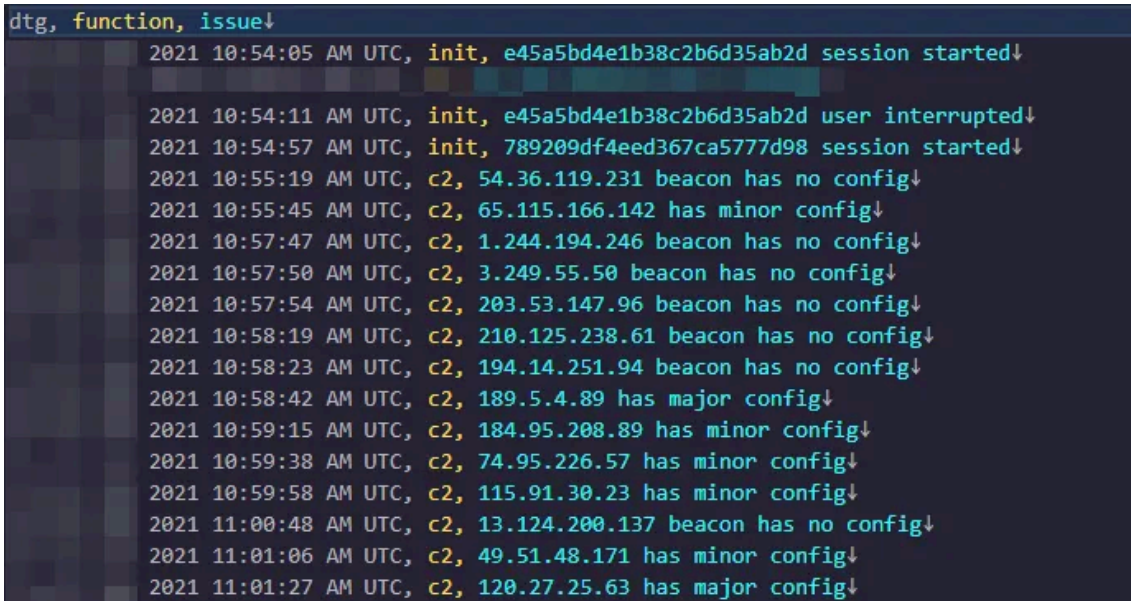
Published: 2021-09-17 · Archived: 2026-04-05 20:55:35 UTC

Press enter or click to view image in full size

Grabbing the configs

So the first thing we do is run cb_looper against the Shodan export to collect all the major and minor cobalt strike configs exported into a Splunk index for later parsing and analysis.

The below image is the cb_looper log file, and as we can see, we have started to collect major and minor configurations, plus some IP's do not have enumerable beacon configs.



Logging is working!

After looper has finished, we can check to ensure the configurations were extracted and have the correct information we need, as displayed below.


```
{
  "x64": {
    "sha256": "31c8d2a6dbbc334304c097555cd5b7f7313ea0fd03d554d4cae2882b5d069eaa",
    "uri_queried": "/7Kfu",
    "time": 1631801693496.5,
    "sha1": "be3f2b39db7b9dcd8ba50bd42a5b3617a45a4ed3",
    "config": {
      "Header 2": "",
      "Beacon Type": "8 (HTTPS)",
      "Watermark": 305419896,
      "Max DNS": 235,
      "Port": 443,
      "C2 Server": "43.128.31.241,/c/msdownload/update/others/2016/12/29136388_",
      "Spawn To x86": "%windir%\\syswow64\\rundll32.exe",
      "User Agent": "Windows-Update-Agent/10.0.10011.16384 Client-Protocol/1.40",
      "Header 1": "",
      "C2 Host Header": "",
      "Method 2": "GET",
      "Method 1": "GET",
      "DNS Sleep": 0,
      "Spawn To x64": "%windir%\\sysnative\\rundll32.exe",
      "Jitter": 20,
      "DNS Idle": "8.8.4.4",
      "Polling": 60000,
      "HTTP Method Path 2": "/c/msdownload/update/others/2016/12/3215234_",
      "Pipe Name": ""
    },
    "md5": "2b5ebe93cbf49755b5dcd69f85b94559"
```


Example Beacon config

Lets check the Splunk!

Get Intel Operator’s stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

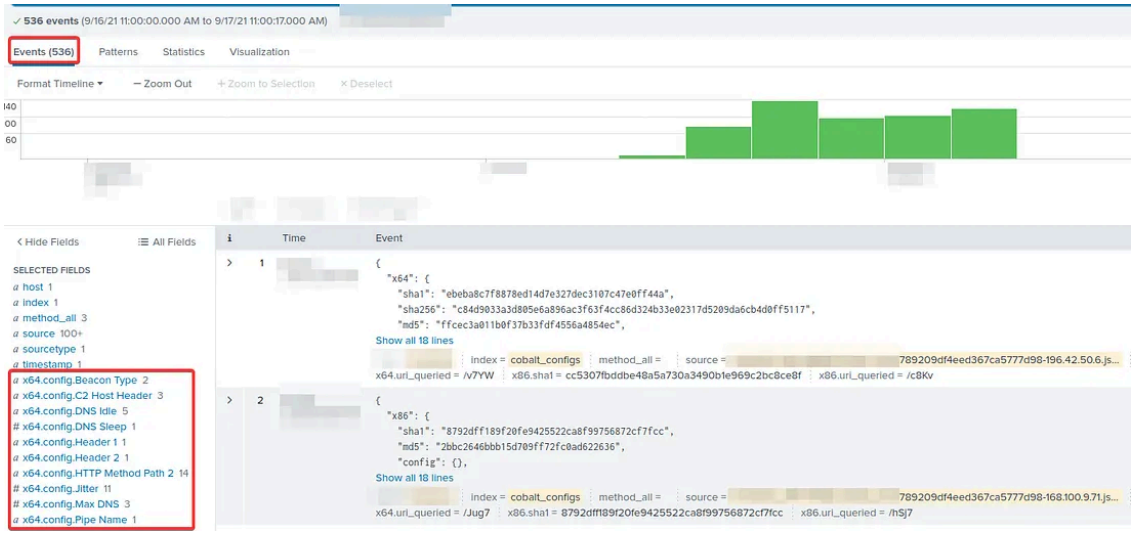
As we can see, all the configurations are indexed within Splunk correctly. There are approx. 40 or so fields that contain the data we need (Image was snipped for visibility)

> If you are doing this yourself and using Splunk's data models I suggest you rename the fields with spaces and join certain fields like C2, user agent etc.

```
|rename "x64.config.C2 Server" as x64c2
|rename "x86.config.C2 Server" as x86c2
|rename "x86.config.User Agent" as x86ua
|rename "x64.config.User Agent" as x64ua
```

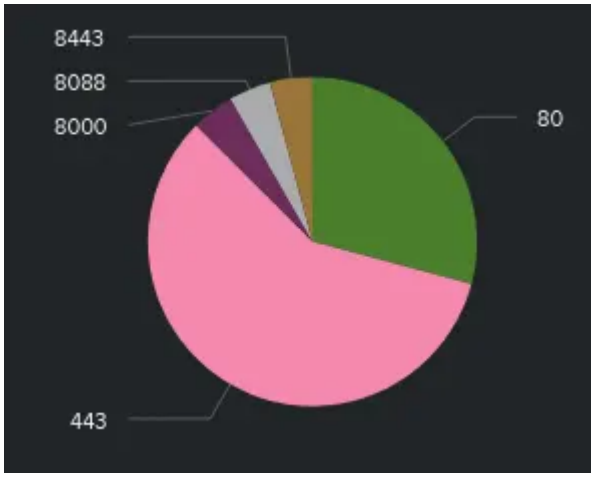
```
|rename "x64.config.Spawn To x86" as x64spwnx86
|rename "x64.config.Spawn To x64" as x64spwnx64
```

Press enter or click to view image in full size



Splunk data.

Press enter or click to view image in full size



C2 Port %

So after I configured all the correct data models, I started analysing the data for this subset of C2 beacons and identified that port:443 was the most popular and port:8443 was the least. It's also interesting to note that not one of these beacons uses any custom ports.

After that, I wanted to see the main injection processes, as seen below in the code snippet and image.

```
"%windir%\syswow64\WUAUCLT.exe"
"%windir%\syswow64\WerFault.exe"
"%windir%\syswow64\dllhost.exe"
"%windir%\syswow64\eventvwr.exe"
"%windir%\syswow64\msdt.exe"
"%windir%\syswow64\mstsc.exe"
"%windir%\syswow64\rundll32.exe"
"%windir%\syswow64\spoolsv.exe"
"%windir%\syswow64\svchost.exe -k netsvcs"
```

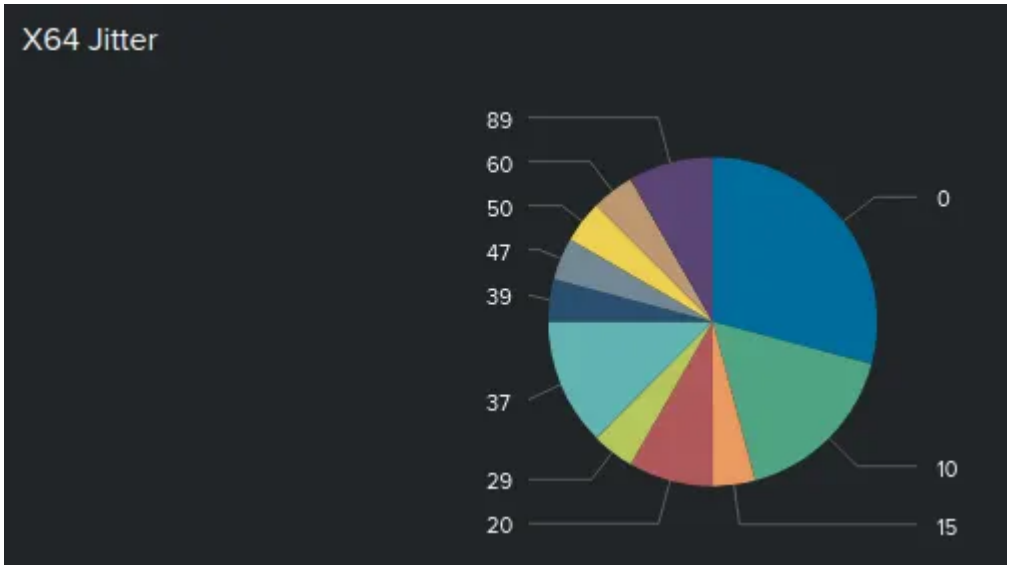
Then I checked the C2 URLs for both X86 and X64 beacons.

```
[SNIP]
"121[.]4[.]213[.]91,/push"
"129[.]28[.]201[.]96,/geo/collect/v1"
"164[.]138[.]25[.]191,/resolve/alter/"
"46[.]19[.]37[.]133,/resolve/alter/"
"clubuz[.]com,/us/ky/louisville/312-s-fourth-st[.]html"
[SNIP]
```

User agents used for X64 beacons.

```
[SNIP]
"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; InfoPath.2; InfoPath.3)"
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; BOIE8;ENUS)"
```

```
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; InfoPath.2)"
"Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)"
"Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0;) like Gecko"
"Windows-Update-Agent/10.0.10011.16384 Client-Protocol/1.40"
[SNIP]
```



X64 Beacon Jitters

Finally, I wanted to see if I could create detection methods based on time-based analysis, and I realised most of the beacons use zero jitters, and none of the beacons uses any custom jitter, which is a bit sad :(.

Thanks for reading, and if you have any questions or “positive feedback”, feel free to reach out, and as always…..

Source: https://inteloperator.medium.com/the-default-63-6f-62-61-6c-74-strike-8ac9ee0de1b7