Alleged source code of Cobalt Strike toolkit shared online

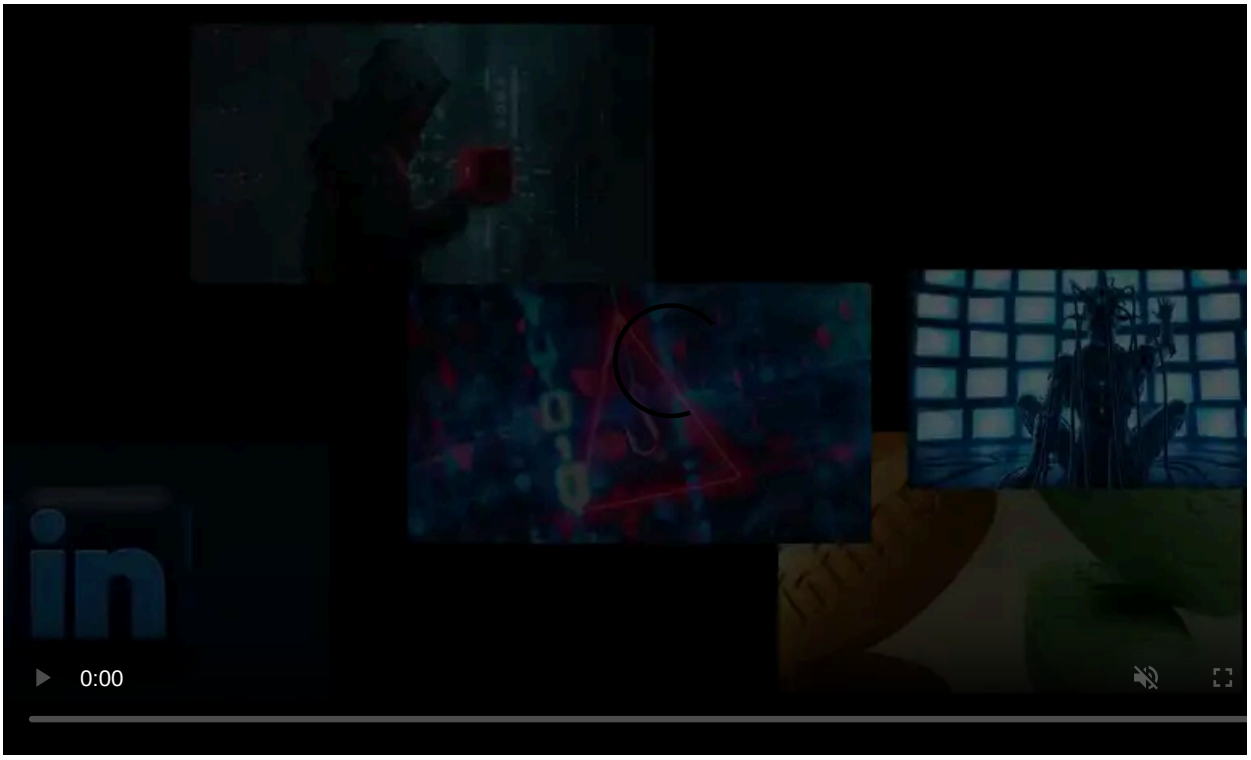
By Lawrence Abrams

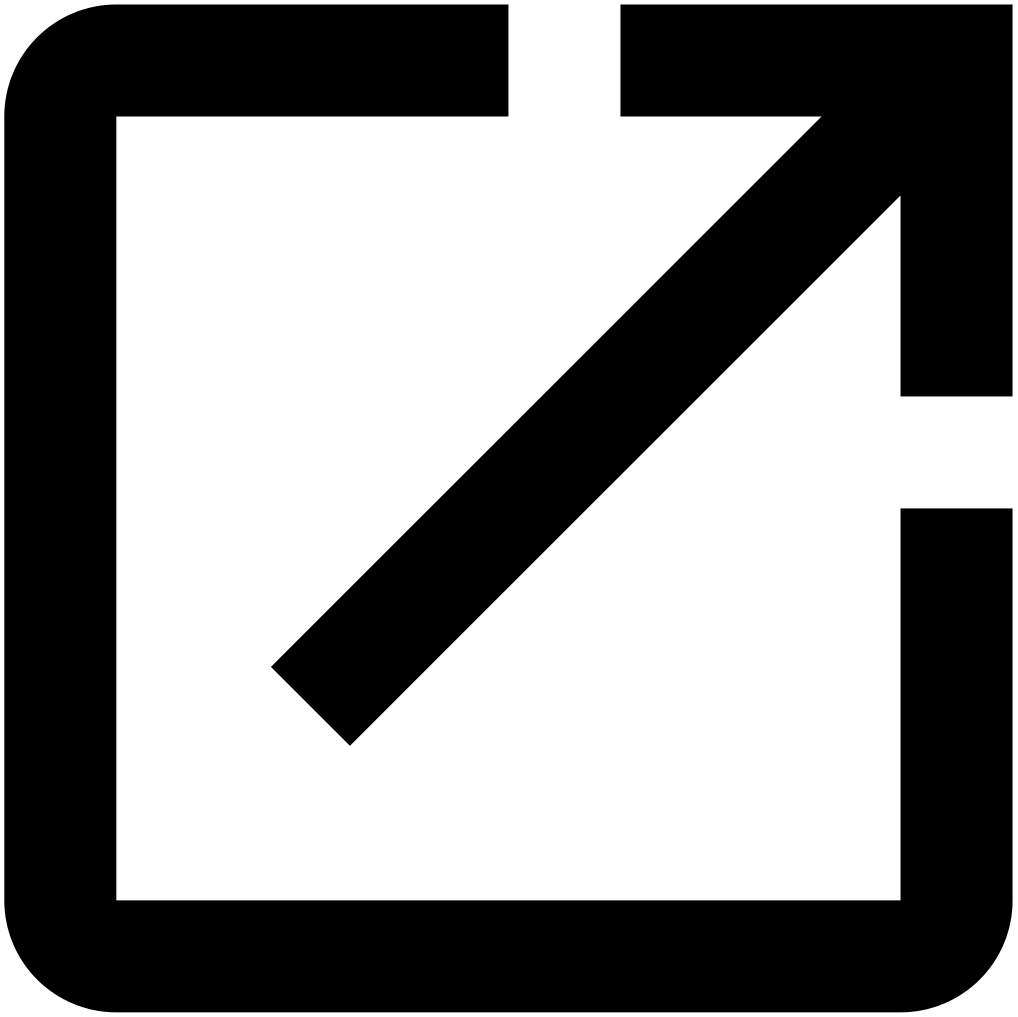
Published: 2020-11-11 · Archived: 2026-04-05 16:54:55 UTC

The source code for the widely-used Cobalt Strike post-exploitation toolkit has allegedly been leaked online in a GitHub repository.

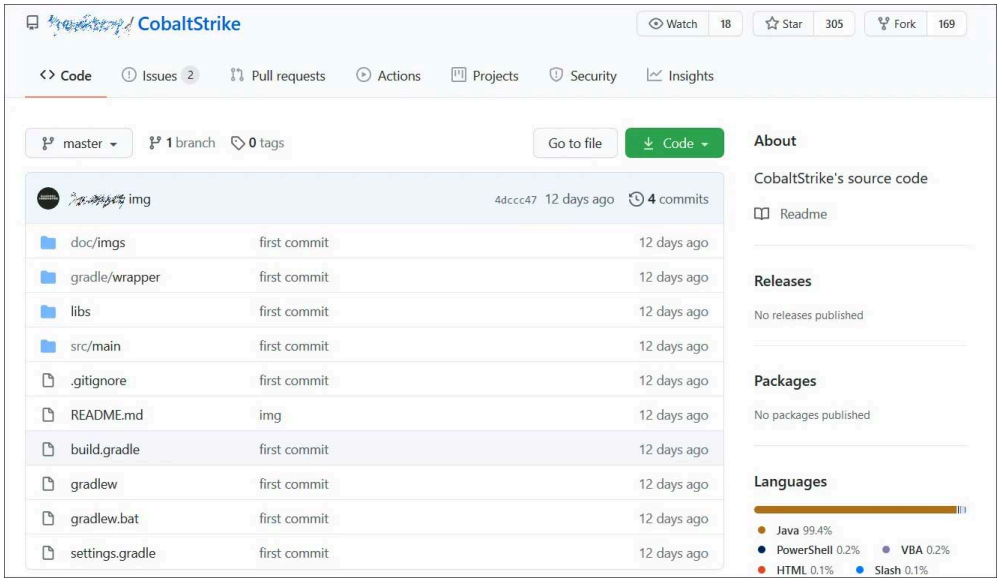
Cobalt Strike is a legitimate penetration testing toolkit that allows attackers to deploy "beacons" on compromised devices to remotely "create shells, execute PowerShell scripts, perform privilege escalation, or spawn a new session to create a listener on the victim system."

Cobalt Strike is an immensely popular tool among threat actors who use cracked versions to gain persistent remote access to a compromised network. This tool is commonly seen used during ransomware attacks.


0:00

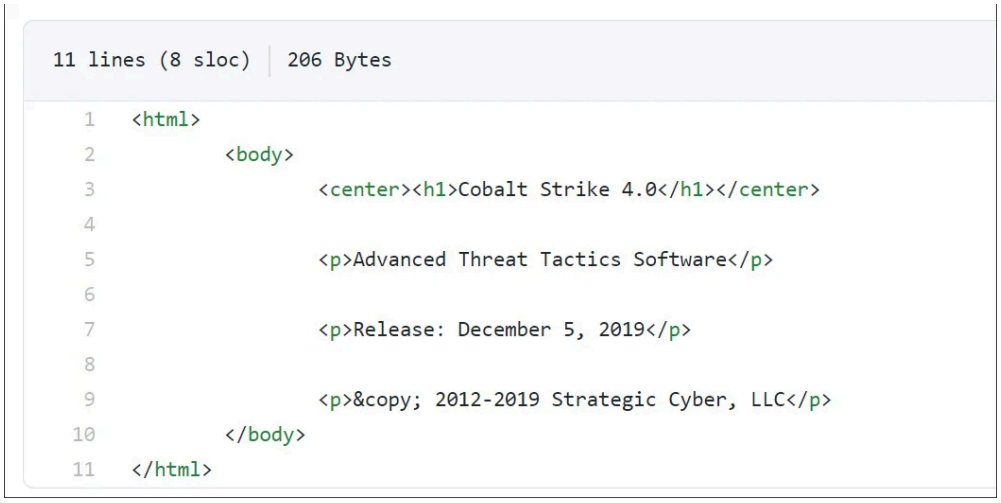


Twelve days ago, a repository was created on GitHub that contains what appears to be the source code for Cobalt Strike 4.0.



CobaltStrike GitHub repository

Based on the 'src/main/resources/about.html' file, this source code is for Cobalt Strike 4.0 released on December 5th, 2019.



11 lines (8 sloc) | 206 Bytes

```
1  <html>
2          <body>
3                  <center><h1>Cobalt Strike 4.0</h1></center>
4
5                  <p>Advanced Threat Tactics Software</p>
6
7                  <p>Release: December 5, 2019</p>
8
9                  <p>&copy; 2012-2019 Strategic Cyber, LLC</p>
10         </body>
11 </html>
```

Source code showing Cobalt Strike version

As can be seen from the source code below, the license check for Cobalt Strike has been commented out, which essentially cracks the program for anyone who decides to compile it.

Aggressor.java

```
1  package aggressor;
2
3  import aggressor.dialogs.ConnectDialog;
4  import aggressor.ui.UseSynthetica;
5  import common.Authorization;
6  import common.License;
7  import common.Requirements;
8  import sleep.parser.ParserConfig;
9
10 import java.io.IOException;
11
12 public class Aggressor {
13
14     public static final String VERSION = "4.0 (20191205) " + (License.isTrial() ? "Trial" : "Licensed");
15     public static MultiFrame frame = null;
16
17     public static MultiFrame getFrame() {
18         return frame;
19     }
20
21     public static void main(String[] args) throws IOException {
22         ParserConfig.installEscapeConstant('c', "\003");
23         ParserConfig.installEscapeConstant('U', "\037");
24         ParserConfig.installEscapeConstant('o', "\017");
25         new UseSynthetica().setup();
26         Requirements.checkGUI();
27         // License.checkLicenseGUI(new Authorization());
28         frame = new MultiFrame();
29         new ConnectDialog(frame).show();
30     }
31 }
32
```

Cobalt Strike license check commented out

Advanced Intel's Vitali Kremez, who examined the source code, told BleepingComputer that he believes the Java code was manually decompiled. The person then fixed any dependencies and removed the license check, so that it could be compiled.

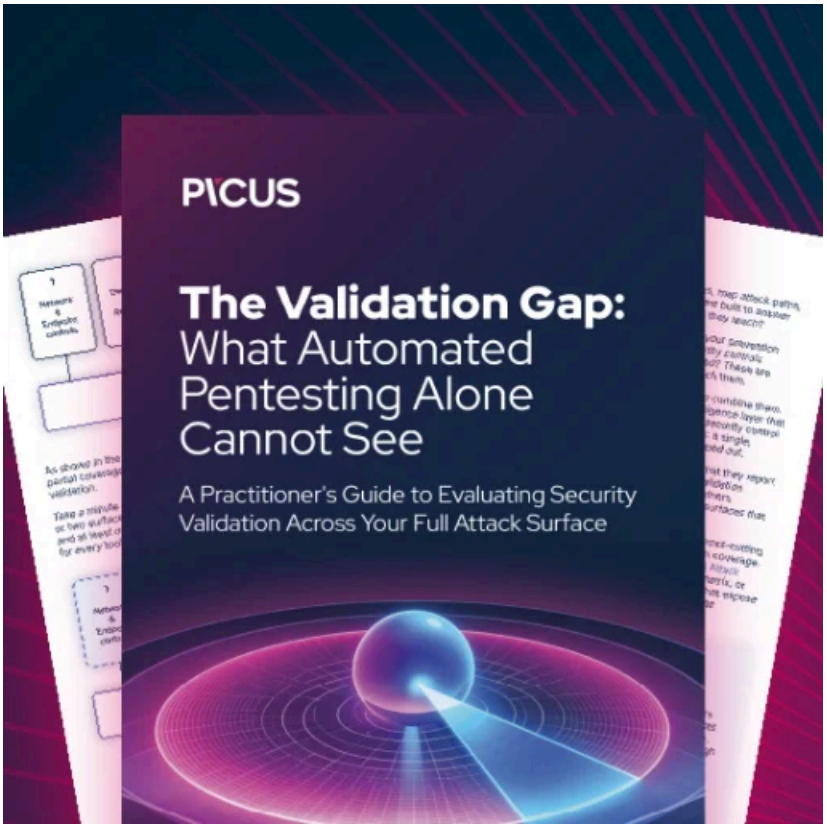
Since being posted, the repository has been forked 172 times, making it hard to contain the spread of the source code.

Even though it is not the original source code, it is enough to be of serious concern to security professionals.

"The possible re-compiled source code exposure of the “2019” Cobalt Strike 4.0 version has significant consequences for all defenders as it removes barriers of entry to obtaining the tool and essentially makes its easy for the crime groups to procure and modify code as needed on the fly."

"The leak of the offensive tool opens the door for the additional crime actor enhancement of the tooling as it happens with the many malware tool leaks such as for Zeus 2.0.8.9. leak and TinyNuke one as they continuously re-used and updated by the crimewave goops and live their own “life” after the leak," Kremez told BleepingComputer in a conversation.

BleepingComputer has contacted Cobalt Strike and their parent company Help Systems to confirm the source code's authenticity but has not heard back.



Automated Pentesting Covers Only 1 of 6 Surfaces.

Automated pentesting proves the path exists. BAS proves whether your controls stop it. Most teams run one without the other.

This whitepaper maps six validation surfaces, shows where coverage ends, and provides practitioners with three diagnostic questions for any tool evaluation.

Source: https://www.bleepingcomputer.com/news/security/alleged-source-code-of-cobalt-strike-toolkit-shared-online/