<https://blog.cobaltstrike.com/2016/09/28/cobalt-strike-rce-active-exploitation-reported/>

“**The Vulnerability**

The vulnerability is a [directory traversal attack](https://en.wikipedia.org/wiki/Directory_traversal_attack) allowed by improper sanitization of parameters in the file download feature of the Beacon and SSH session payloads.

One may connect to a Cobalt Strike listener, download the payload stage, use the information in the stage to fake a session, and craft a message to force Cobalt Strike to write a file to an arbitrary location.”

<https://blog.cobaltstrike.com/2016/10/03/cobalt-strike-3-5-1-important-security-update/>

### **The Vulnerability**

Cobalt Strike’s team server is a controller for the [Beacon post-exploitation payload](https://www.cobaltstrike.com/help-beacon). Cobalt Strike has options to serve and control the Beacon payload over HTTP, HTTPS, and DNS. Embedded within the Beacon payload are directives that tell the Beacon payload how to communicate with its team server.

By design, any party can download the Beacon payload and its embedded configuration. This allows a Beacon to [bootstrap on a newly compromised system](https://blog.cobaltstrike.com/2013/06/28/staged-payloads-what-pen-testers-should-know/) and take steps to [authenticate and communicate with its team server](https://www.youtube.com/watch?v=3gBJOJb8Oi0&feature=youtu.be&t=49m55s). Conversely, this means the information a malicious actor needs to establish communication with a team server is available to them.

Once Beacon runs, its first job is to securely send a randomly generated session key and other information about itself (username, IP address, process ID, etc.) to its team server. Cobalt Strike refers to this information as session metadata.

After this, the Beacon periodically connects to its team server, asks for tasks, sends response directives, and goes back to sleep. These response directives are a limited set of actions that a Beacon may ask its team server to execute. Most of the responses simply format and present output to the user (e.g., keystrokes, output from a command, etc.).

Some response directives work together to support more complicated tasks. For example, there are three response directives that support file downloads.

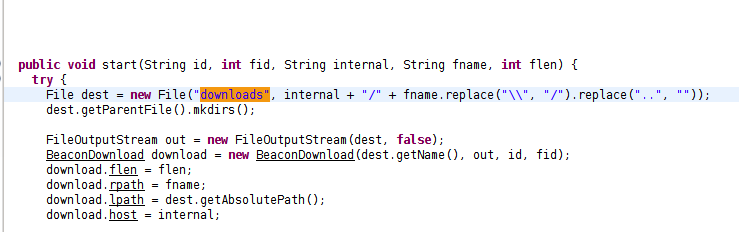
The first file response directive starts a file download. This directive accepts as input an integer file ID, an integer that is the file length, and a string with the full path to the file on the remote system. This directive notifies users that a download has begun and it opens a handle to write the downloaded file to disk. This directive then associates the file ID with this handle.

The second file response directive accepts an integer file ID and a binary blob. This directive writes the binary blob to the file handle that maps to the file ID for the current session. A Beacon session may make multiple requests with this directive to send a large file to a team server.

The third file response directive accepts an integer file ID. This directive formally informs the team server that the file download is complete.

The team server does not map response directives to previous tasks. Once a client establishes a session, it has the freedom to request execution of any response directives in any order or quantity.

The team server stores files it downloads into a fixed path. That path is **downloads/***[internal IP address of session]***/***[path/to/remote file]*.



***Screenshot: CS 3.1 vuln function***

The *[path/to/remote file]* input comes from the first file response directive. The team server took steps to sanitize this value in an attempt to prevent a directory traversal attack. These steps were not [best practice](https://www.securecoding.cert.org/confluence/display/java/FIO16-J.+Canonicalize+path+names+before+validating+them) for the Java platform, but some measure was in place. The information provided to Strategic Cyber LLC did not indicate that this value was the source of the directory traversal input.

What other input is there? There’s the *[internal IP address of the session]*. The team server uses this value to organize downloaded files and to organize its logs. Where does this value come from? It comes from the session metadata. Who controls the session metadata? The Beacon session controls this value.

This led to the root cause of the issue: The team server extracts information from the session metadata and makes that information available to other features as trusted information about that session. The team server did not validate these metadata parameters for expected form or sanitize these parameters for malicious inputs.

Hot Fix 1 took steps to mitigate the in-the-wild exploit and buy time for further investigation. Hot Fix 2 mitigated the identified root cause of this vulnerability and potential variants by adding strict checks and sanitization to the session metadata.

### **The Hardening Measures**

This release restores functionality degraded by last week’s Hot Fixes for this vulnerability, improves on Hot Fix 2’s measures, and hardens the Cobalt Strike team server against this vulnerability and potential variants.

Here are the changes:

1. This release reworks the download response directives to use randomly generated names for downloaded files stored on the team server. Information about the downloaded files (name, where they came from, etc.) is logged to **logs/***[date]***/downloads.log**. The **View** -> **Downloads** tab displays the real file name and original remote path. The **Sync Files** button works as it did prior to the Hot Fixes.

2. The team server now uses a safe path concatenation function throughout its codebase. This function compares the canonical paths of the parent and candidate result to make sure the result doesn’t break out of its parent.

3. This release adds a **host\_stage** option to [Malleable C2](https://www.cobaltstrike.com/help-malleable-c2). This option controls whether or not Cobalt Strike hosts Beacon stages for download over HTTP, HTTPS, and DNS. If set to false, staging functionality will be unavailable, but this is useful for teams with a [no-network staging policy](https://blog.cobaltstrike.com/2016/06/22/talk-to-your-children-about-payload-staging/).

4. The team server now refuses to process a session if any of its metadata fails validity checks. This is a minor improvement on the changes made in Hot Fix 2.

5. The team server now denies new sessions with no prior tasks access to most response directives.