Hunchly Evidence Guide

Revision: 1.0
Date: October 17, 2018
Author: Justin Seitz (justin@hunch.ly)
Contact: support@hunch.ly

Table of Contents

1. Introduction

This section of the Hunchly knowledgebase is designed to help educate Hunchly users and consumers of Hunchly evidence. The major areas of how Hunchly collects evidence are all covered including potential evidence challenges and attacks against evidence collected by Hunchly.

It is important to note that Hunchly does everything within reason to make sure that evidence is hashed, GPG signed and preserved accurately in an attempt to do our best to ensure no tampering has occurred. However, there are ALWAYS attacks that can be carried out against ANY software that collects and distributes forensic evidence. This is no different than submitting PDF evidence manually, screenshots, or anything else. Because YOU the investigator are in charge of your own data, you will always have the ability to mess with that data.

We believe in security and transparency at Hunchly, which means we are not afraid to disclose what we feel are weaknesses in our own system or ways that the evidence can be challenged. We would much prefer this than you hearing about it while in a witness box in court.

We ask that if you see additional attacks or gaps in how we are handling evidence in Hunchly that you reach out to us so that we can, test, document and disclose those weaknesses.

2. The MHTML File Format

Hunchly captures all web pages in the MHTML format. This format is very similar to how emails are structured, they contain headers with information describing the page itself, the timestamp of when Chrome itself captured the page and it also includes all of the text, CSS styles and images that are contained on the page. All in a single file. This is superior to PDF or screenshots as all links are maintained, the layout is generally more accurate and all metadata is preserved including the metadata in the captured images.

It is worth noting that the MHTML capture is 100% handled by Google Chrome, Hunchly simply instructs Chrome to do the capture and then pulls out the result for storage and analysis.

The beginning of the MHTML file has a header that has some high-level metadata that may be useful when doing disclosure or validation:

```
From: <Saved by Blink>

Snapshot-Content-Location: https://www.facebook.com

Subject: Facebook - Log In or Sign Up

Date: Tue, 28 Aug 2018 15:47:07 -0000

MIME-Version: 1.0

Content-Type: multipart/related;
```

From: this is just mentions that it was the Chrome Blink engine that captured the page.
Snapshot-Content-Location: the URL of the page that was captured.
Subject: the HTML title of the page.
Date: the timestamp in UTC of when Chrome saved the page and sent it to Hunchly for storage.

Hunchly's Storage and Identification of Pages

Hunchly stores the pages from your case by utilizing a PAGE ID that is unique to each page. This ID is global, which means that it is across all of your cases, so you could have CASE 1 that has Page ID 1 and CASE 2 that has Page ID 2. When Hunchly saves MHTML content the file is named: PAGEID.mhtml.

Potential Evidence Challenges

Timestamp Mismatches

When submitting MHTML pages for disclosure or court purposes you may have the issue where the timestamp in the MHTML file does not match the timestamp that Hunchly has listed for the capture. There are two explanations for this that you can communicate:

1. Hunchly timestamps the data in your local time zone. To get an accurate match between the MHTML file and the timestamp produced in a Hunchly export you need to convert the UTC time to your local timezone. You can alternatively change your local clock to UTC/GMT and then the timestamps should be the same.

2. There is a slight delay between when Chrome captures the page and when it forwards it to Hunchly for processing. This delay is dependent on how large the page is, how many pages Hunchly has queued up for storage or the overall performance of your computer. This may result in timestamps that are not the same between the MHTML file and what Hunchly shows, but is easily explained as other parts of the evidence such as the SHA-256 hash and GPG signature will still match the content to ensure that the evidence stands up to scrutiny.

Non-Continuous Page IDs

When submitting evidence to the court or a third party you may be questioned why there are non-continuous Page IDs in the submitted evidence, some explanations you can provide:

1. You work multiple cases that are separate but the Page ID is universal across the entire system. The number increments regardless of what case you are working on but the cases themselves are logically separated on the investigators hard drive and in the Hunchly database.

2. You have deleted a page which creates a "gap" in the Page IDs submitted. You can explain the deletion because Hunchly has a deletion log that you can view to explain any deletions for a case should you be required to do so.

3. Content, Photo and Attachment Hashing

Hunchly automatically hashes all MHTML files, photos and attachments before storing. The hashing algorithm is **SHA-256** and can easily be reproduced using common open source or command line tools. Extracting a page from a Hunchly export or by exporting a single page and then using SHA-256 from an independent tool should ALWAYS match the hash that Hunchly displays in the Dashboard for that capture.

If the hashes do not match then you have either modified the content or there has been an error when saving the files, and you should alert us immediately at support@hunch.ly.

Potential Evidence Challenges

Hashes are not a bulletproof means to validate that the content was not changed BEFORE the MHTML capture was sent for review or disclosure. For example, an evil investigator looking to modify evidence could do the following:

1. Capture a page that has a hash of ABCDEFG.
2. Export the page to disk.
3. Open the MHTML file and modify the content to say something other than what was captured.
4. Manually generate a NEW hash which would be HIJKLMNOP for example.
5. Submit the modified page and the new hash as evidence. When a third-party verifies this evidence it will mistakenly appear as though everything checks out.

This can be tricky to combat but there are a few strategies:

1. Demand that the page be submitted along with the GPG signature that Hunchly produces and the public key. While not perfect, it does place an additional roadblock on the evil investigator.

2. Validate the evidence through the source. For example, validate that the page content that is still active (if available) reflects what the investigator submitted for evidence. There are also sources of archived webpages such as the Wayback Machine that could be consulted.

3. Through a court order or other legal means, obtain the original material from the website where the evidence was collected, and compare to the submitted evidence to ensure that they match.

4. GPG Signing and Validation

When you install Hunchly it generates a GPG key pair. Your private key stays with your Hunchly installation, and the public key is included when you do a full case export to zip file. Each MHTML capture is signed with your private key and produces a .sig file.

For example after you capture a page, its filename will be in the format: PAGEID.mhtml and the GPG signature file will be PAGEID.mhtml.sig

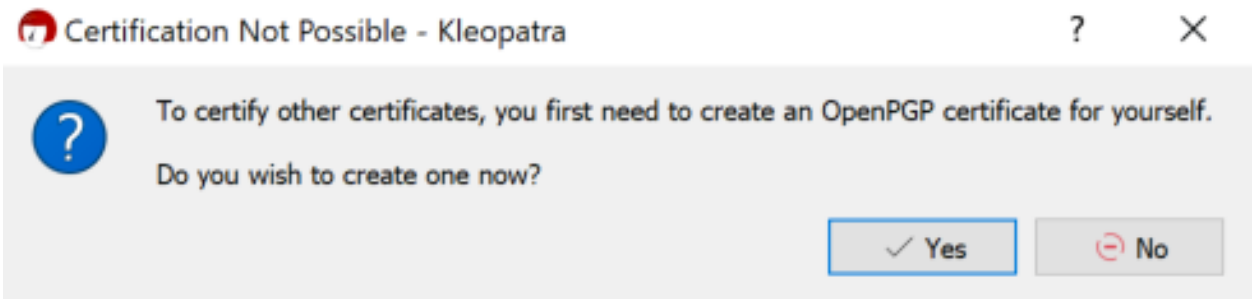
This signature can then be used to validate whether a produced MHTML file from Hunchly was tampered with or altered in any way. When the signature verification is performed it will fail and you will know that the MHTML content has changed *after* it was captured and signed by Hunchly.

Validating GPG Signatures

If you have submitted a full Hunchly case zip file there will be a public key named: **public.key** in the export. You will use this key to validate captured web pages. The steps for each operating system are below, although there are many GPG compliant tools available so use the tool you feel most comfortable with.

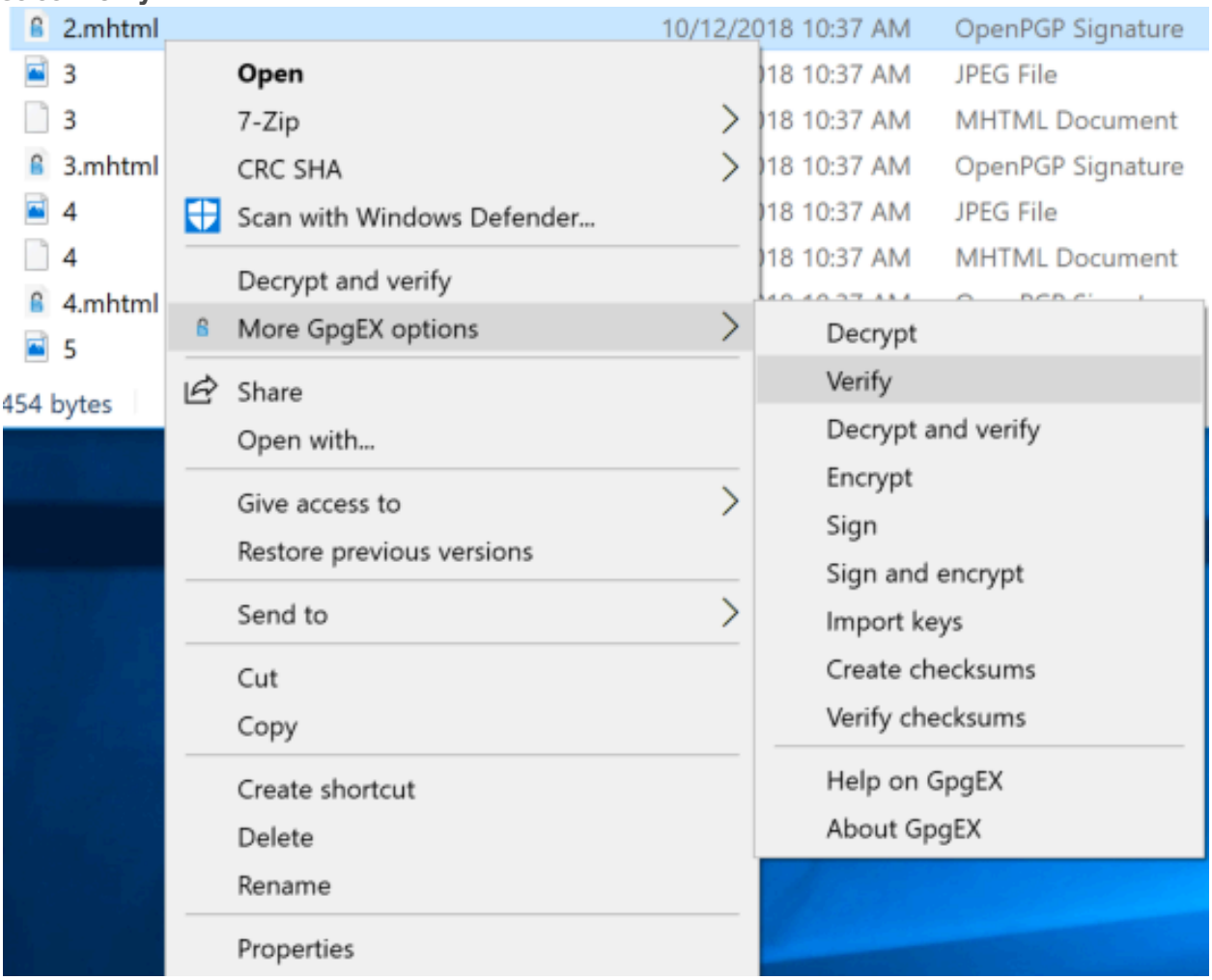
1. Windows

1. Download and install GPG4Win for Windows. We recommend the installer you can grab from here: https://gpg4win.org/thanks-for-download.html
2. GPG4Win has a nice GUI that you can use called Kleopatra. First right-click on the **public.key** file in your Hunchly export and from the **More GPGEx** menu select **Import Keys.** If you do not already have a private/public key pair on the system it will ask you to create one:

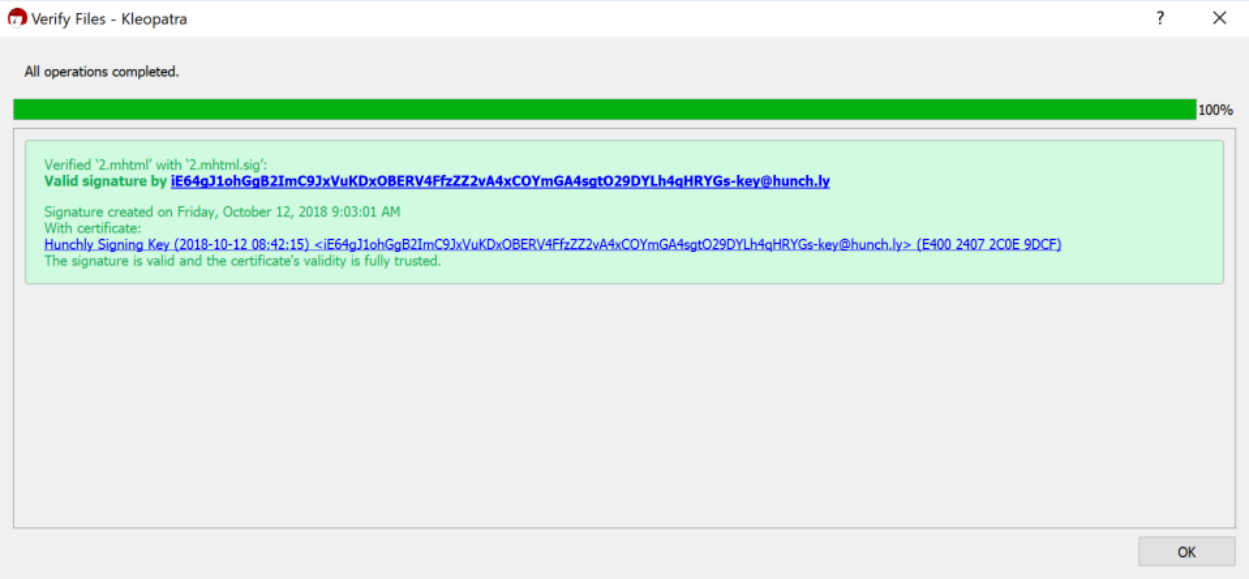


Simply follow the onscreen prompts, fill out a name and email address, and enter a passphrase to secure the keypair. Note that this keypair is not for use with secure emailing, we are just using it to import our Hunchly public key.

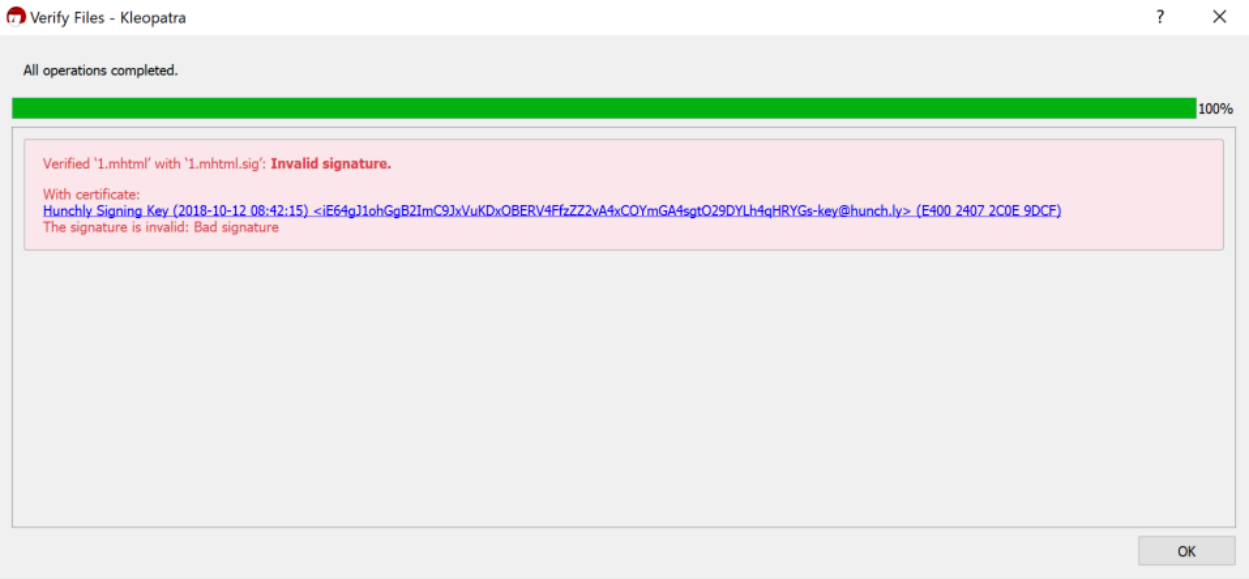
3. Once you have setup your keypair and imported the Hunchly public.key you can now verify captured pages by right-clicking on the PageID.mhtml.sig file and from the **More GPGEx** menu select **Verify.**



4. If the verification was successful you should see a message as shown below:



5. If the verification has failed then you will be shown a dialog as shown below, and this indicates you should not trust that the page has not been tampered with:



2. Mac OSX

1. Download and install GPG. We recommend the installer you can grab from here: https://sourceforge.net/p/gpgosx/docu/Download/

2. Using the Terminal app, import the public key from the Hunchly export.

   gpg2 --import public.key

3. Now you can verify any page by doing the following:

   gpg2 --verify PageID.mhtml.sig

   The resulting output should look like so when validating 1.mhtml as an example:

   $ gpg2 --verify pages/1.mhtml.sig

   gpg: assuming signed data in 'pages/1.mhtml'
   gpg: Signature made Wed 10 Oct 14:04:13 2018 PDT
   gpg: using RSA key C21DE145181D0125
   gpg: **Good signature** from "Hunchly Signing Key (2018-10-10 13:42:58) <Fj1CvL2tJWLTlSq1aNiS9YSND2bOCkHhw6AOos5p7VQT8PTJHh9yuFv8bES4WJGn-key@hunch.ly>" [unknown]
   gpg: WARNING: This key is not certified with a trusted signature!
   gpg: There is no indication that the signature belongs to the owner.
   Primary key fingerprint: 2E37 C58A A8F5 B65D 0710 4CA2 C21D E145 181D 0125

4. If you see "Good Signature" that is indicating that the GPG signature was validated. If we modify even a single character in the MHTML file and try to validate the signature you will see a message like so:

   $ gpg2 --verify pages/1.mhtml.sig

   gpg: assuming signed data in 'pages/1.mhtml'
   gpg: Signature made Wed 10 Oct 14:04:13 2018 PDT
   gpg: using RSA key C21DE145181D0125
   gpg: **BAD signature from** "Hunchly Signing Key (2018-10-10 13:42:58) <Fj1CvL2tJWLTlSq1aNiS9YSND2bOCkHhw6AOos5p7VQT8PTJHh9yuFv8bES4WJGn-key@hunch.ly>" [unknown]

5. The message "BAD signature" indicates that the data has been modified and you should no longer trust that the page has not been tampered with.

3. Linux

1. Install GPG 2.0 using the following command:

   sudo apt install gnupg2

2. From there the steps are identical to the steps in the Mac OSX section above.

Potential Evidence Challenges

There are a few ways that the GPG signing method we use can be attacked. One of the biggest challenges is that the GPG key generation is local to your machine, as the investigator. Ideally the key generation would happen with a trusted third-party and controlled centrally, but due to the fact that Hunchly is deployed in sensitive locations it does not make it feasible for this to occur.

This means that an evil investigator, with minor technical skills, could in fact attack the GPG signing in Hunchly by doing the following:

1. Extract the private / public key pair from the Hunchly database.
2. Modify the content of the MHTML page they wish to alter.
3. Re-sign the page manually using the private / public key pair.
4. Submit the page as evidence, and the GPG signing will still validate as if the content was accurate.

As covered in the Hashing section of this guide, there are still ways for you to verify that the content was tampered with if you see abnormalities in the content or question whether the GPG signature is valid.

5. Deletion Logging

Hunchly monitors any deletions that occur within your case and produces a deletion log that you can review or use to explain gaps in evidence (as discussed in the MHTML section of this guide).

This log file is located in your HunchlyData/cases/CASENAME folder and is named: deletion.log

An example of what is present in this log file:

time="2018-10-10T14:04:55-07:00" level=info msg="Delete tag from page: id - 2, tag id - 2, name - 'Blogs'"
time="2018-10-12T09:30:30-07:00" level=info msg="Delete note: id - 2, Page URL - 'https://exposingtheinvisible.org/resources/image-digging', Page ID - '6'"
time="2018-10-12T09:31:00-07:00" level=info msg="Delete page: id - 14, url - 'https://sourceforge.net/projects/gpgosx/', capture date - '2018-10-11 22:43:24 +0000 UTC'"

It is a very simple way to see whenever you have removed pieces of information from Hunchly that you may later have to explain in court or in your report writing.