XCOLD Information Leak

New Functionality for Developers,
New Security Concerns

Based on my original article: http://zaproxy.blogspot.co.uk/2016/01/zap-newsletter-2016-january.html#Feature
Ottawa ended up with an uncharacteristically green Christmas this year, but finally got snow as I’m drafting this (Dec 29th), so I'm going to dub this the XCOLD Info Leak (X-ChrOmeLogger-Data).
Intro

Markup view shows indicators for pseudo-classes locked for elements

Bind F1 key to open the settings when the toolbox is focused

New 'Use in Console' context menu item in Inspector to store selected element in a temporary variable

Search button next to overridden CSS properties to find similar properties in the rules view

Ability to filter styles from their property names in the rules view

Stack traces are now shown for exceptions inside the console

Added ability to display server-side logs in the console
Functional Summary

Basically it boils down as follows:

- Decide you want server side messages available to clients.
- Stick some code into your web app.
  - Support seems to be quite extensive: Python, PHP, Ruby, Node.js, .NET, ColdFusion, Go, Java, and Perl. Though the majority of the existing install/usage seems to be PHP.
- Boom get messages in the browser (Chrome requires an extension, while Firefox now has native support).
  - Messages are returned as Base64 encoded JSON via the response header X-ChromeLogger-Data.
  - Browsing https://github.com/ccampbell/chromellogger/issues one can also learn that older or alternate implementations use(d) X-ChromePhp-Data.
Ok so X-ChromeLogger-Data is in use. Not extensively but the numbers are increasing. The image here is from 2015-12-29, I had previously poked around on this topic around the 18th. Though I don’t remember the specific number I’d previously queried there seems to have been an increase in a short period and over the holidays.
Use?

It also seems that X-ChromePhp-Data is in use to a much lesser extent. So off I go to see what kind of data people might be exposing.
Looking around...

Keep in mind the following findings are based on public data, I’m not revealing anything here that the site(s) haven’t already revealed to the world.

Below I’ve just copied the Base64 encoded header values from the Shodan results and run them through the handy online decoder at https://www.base64decode.org/.
Looking around...

**Alternative 1:**
Proxy your shodan result browsing through ZAP, select the base64 string and use the “Encode / Decode / Hash” context menu:
Looking around...

Alternative 2: You could also pop the dev tools console (F12) and do a `atob("<some base64 string>");` and hit enter. For example `atob("ZW5jb2RlIHRoaXM=");` returns `encode this`. I used the online service below since the dev tools console doesn’t line wrap :(
How Bad Is It? … Meh

Picking the first X-ChromePhp-Data shodan result I got the following:

Ok not terribly revealing, however it does give the entire disk location of AppController.php. Might be able to leverage that in other attacks or use the knowledge to social engineer something.
How Bad Is It?...Warming Up.

Here’s another example that’s really verbose, note Undefined index: admin references:
How Bad Is It?...Warmer Still.

Here’s another, note the failures in processing various tokens, PREBODY, and HTTP_USER_AGENT. Those details might lead a pentester to another useful finding such as UA specific response or UA injection of some sort.
How Bad Is It?...Hot in here?

Here’s one that appears to be a WordPress plugin. And, another that seems to be a Windows host running some WordPress gallery.
How Bad Is It?...Smoke...Fire?

To top it off, here’s an example leaking raw SQL details:

```
{"version":"4.1.0","columns":["log","backtrace","type"],"rows":[["bij de bakker: sb16siteVindex"],"Vhome\dev\www\admin\routedusoleil\_org\vprotected\vcomponents\CookieBaker.php : 11",""],["Nog geen Resident cookie!"],"Vhome\dev\www\admin\routedusoleil\_org\vprotected\vcomponents\CookieBaker.php : 16",""],["Cookie gebakken!"],"Vhome\dev\www\admin\routedusoleil\_org\vprotected\vcomponents\CookieBaker.php : 19",""],["SQL: INSERT INTO session (cookie, timestamp, uid, sid, referer, page) VALUES ("1451658941c6158", "1451658941c6158", "1451658941c6158", "0", "1", "0", "Vhome\dev\www\admin\routedusoleil\_org\vprotected\vcomponents\CookieBaker.php : 42" )"],"request_uri":"V\snowbreak"}
```
Now What?

While the benefits to a developer are obvious I would suggest that the following considerations or implementation choices be made if using the functionality:

1. Do you want this turned on for Production use?
2. If you do want it on for Production use can you ensure that you don’t leak information that might be leveraged by an attacker or malicious individual?
3. Can you tie it to your authorization framework so that the information (header + content) is returned only to admins and support personnel not “all” users?
Scanner Support...

Curious if your app or site is exposing anything similar to the examples above? Do you have the XCOLD Info Leak?

Checkout the new ZAP passive plugin that looks for and identifies information leaks via X-ChromeLogger-Data and X-ChromePhp-Data. Available via the ZAP Market Place as of 'Passive Scan Rules Alpha' version 8.
Updates

That’s +113 occurrences at the top (Sweden) in 3 wks…

That’s +134 on the HTTPS service counts in 3 wks.
Updates

That’s +6 occurrences at the top (US) in 3 wks…

That’s +7 on the HTTPS service counts in 3 wks.