

Did Google pull a fast one on Firefox and Safari users?

Summary

At the end of 2011, Chrome's protection rate steadily climbed to just over 50% before suddenly falling back to 20%. At the same time, Firefox and Safari's block rate moved in the opposite direction.

Chrome, Firefox and Safari all use Google's Safe Browsing API, and Google has publicly stated that it has not withheld data from their Safe Browsing feed. So what should end users make of the results? At the request of multiple enterprise clients, NSS Labs has performed an in-depth examination of Safe Browsing and how it is implemented in Chrome, Firefox, and Safari, and what this means to enterprise users.

Overview

NSS Labs has conducted significant research over time into the protection capabilities of Chrome, Firefox, Internet Explorer, and Safari. Throughout 2009 and 2010, protection provided by both Firefox and Safari exceeded that of Chrome¹. Since the adoption of Safe Browsing API v2 and the elimination of proprietary solutions, both have demonstrated a reduction in effectiveness at blocking traditional malware downloads.

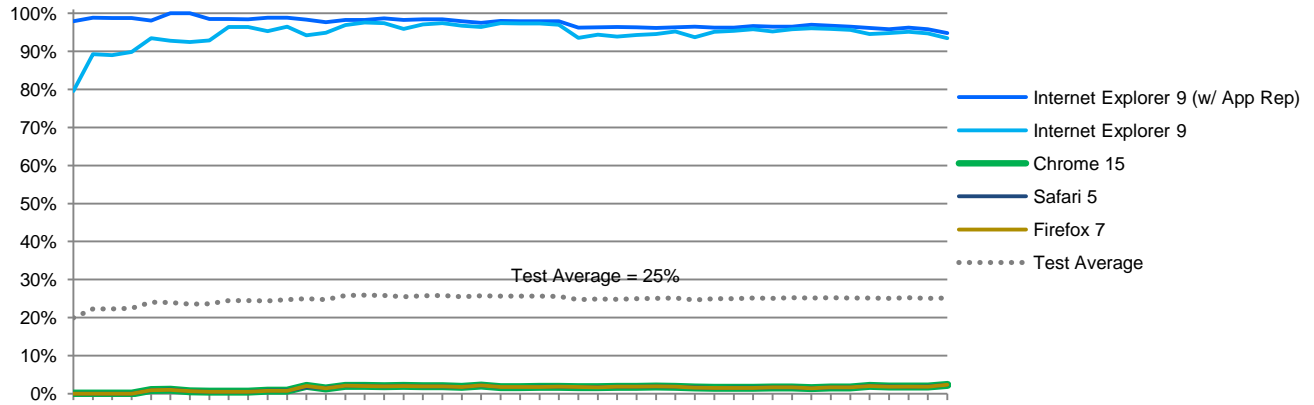
The latest round of testing occurred from November 21, 2011 to January 5, 2012, during which NSS researchers observed what appears to be a significant change when compared with historical results. Chrome's protection rate steadily climbed to just over 50% before suddenly falling back to 20%. Over the same time period (Nov 21, 2011 – December 21, 2011), Firefox and Safari's block rate remained at 2%, and then inexplicably jumped to 7% on the same day Chrome's protection fell precipitously (December 22nd).

Chrome, Safari and Firefox all use the Safe Browsing API to identify potential malicious content and warn users against accessing it unintentionally. The implementation of the Safe Browsing API is different for each browser, however. For example, Chrome uses an undocumented API call to block malware once download begins. This API is not utilized by Firefox or Safari, apparently due to lack of documentation and a proprietary format.

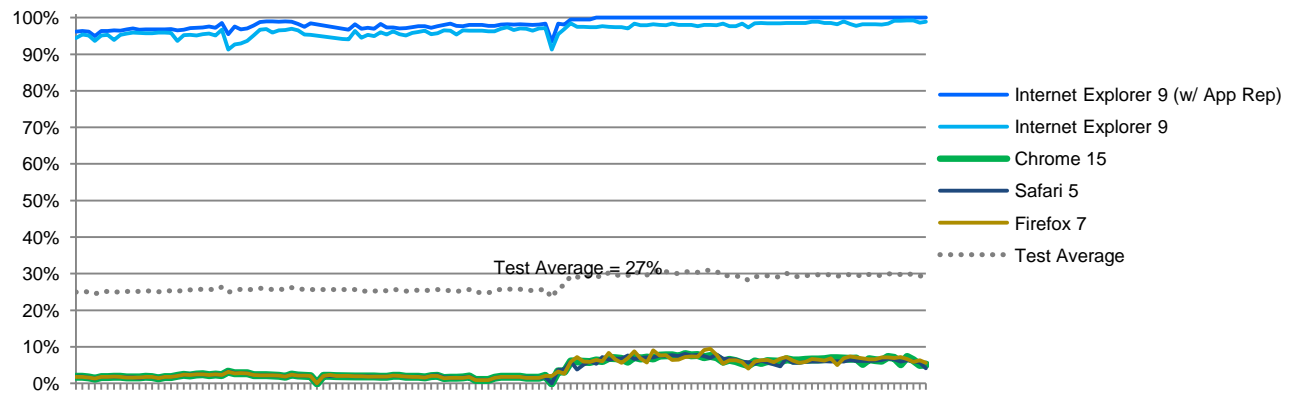
NSS research indicates that Chrome, Firefox and Safari offer nearly equal protection when Chrome's new "malicious download" protection is not factored into the results. This is what Google is calling "Safe Browsing API v2"².

¹ http://www.nsslabs.com/assets/summaries/NSSLabs_Q12010_BrowserSEM_Summ_FINAL.pdf

² <http://code.google.com/apis/safebrowsing/>



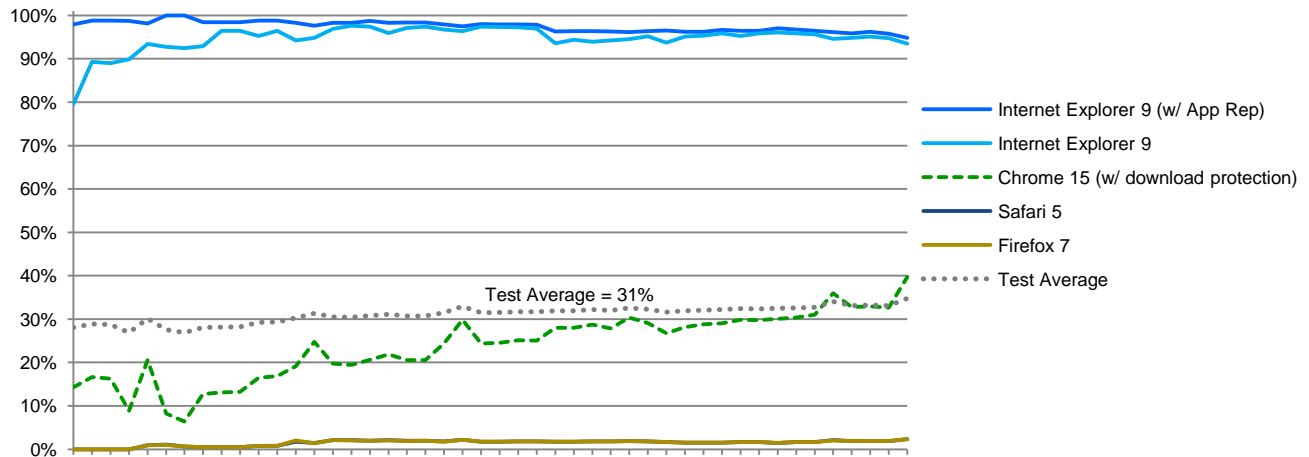
Socially Engineered Malware Protection over time - North America
(November 21, 2011 - December 1, 2011)



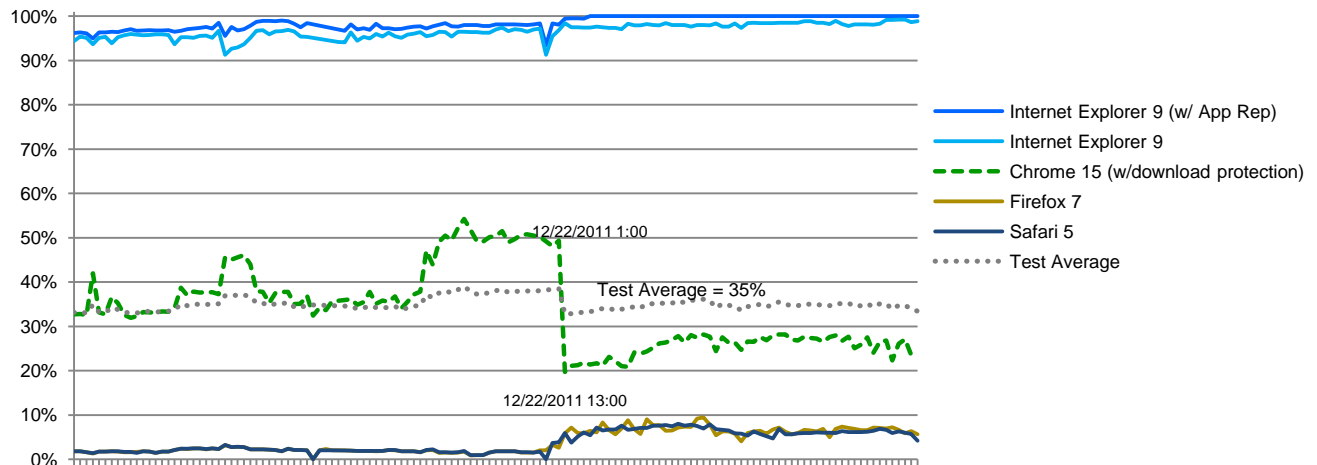
Socially Engineered Malware Protection over time - North America
(December 2, 2011 - January 5, 2012)

However, when Chrome's new malicious download³ protection is factored in, the results are markedly different. Note the significant reduction in Chrome's malicious download protection on December 22, 2011 that coincides with an uplift in Safe Browsing API v2 protection.

³ <http://blog.chromium.org/2011/04/protecting-users-from-malicious.html>



Socially Engineered Malware Protection over time - North America
November 21, 2011 - December 1, 2011

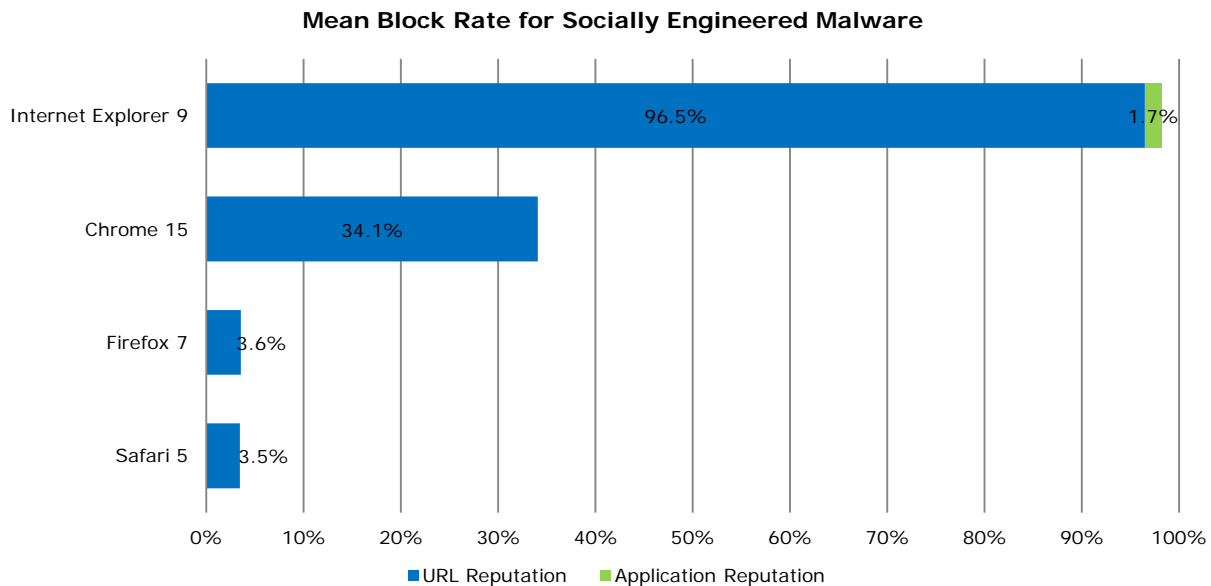


Socially Engineered Malware Protection over time - North America
(December 2, 2011 - January 5, 2012)

This analysis brief was produced as part of NSS Labs' independent testing information services. Leading vendors were invited to participate fully at no cost, and NSS Labs received no vendor funding to produce this analysis brief.

NSS Labs Findings:

- Internet Explorer 9 remains the most effective at blocking traditional malware downloads (a.k.a. social-engineered malware).



- Google and Mozilla agreed on terms of their search agreement December 20, 2011. On December 21-22, 2011 NSS Labs observed a reorientation of protection whereby proprietary protection offered by Chrome dropped dramatically while shared Safe Browsing protection within Chrome, Firefox and Safari increased. While these events may not be related, the timing raises questions.
- Despite claims to the contrary, Google has developed proprietary functionality via Safe Browsing to block malicious downloads. This functionality is not available to the other Safe Browsing API v2 browsers (Firefox and Safari).
- Google has made it extremely difficult for a third party to access the contents of the malicious download list. Using a one-way hash, Google has effectively prevented third parties from garnering useful information from the undocumented portions of Safe Browsing, while maintaining the façade of sharing with the community.
- While Google is technically sharing the Safe Browsing API v2 with Firefox and Safari, the spirit of that arrangement is in question given the poor performance of Safe Browsing *without* malicious download protection.
- While (drive-by) exploits are a growing threat, traditional malware still remains the most prevalent threat to users.

NSS Labs Recommends:

- Remain vigilant against all threats, not only those that are new and “news worthy”. Do not download anything from an untrusted source on the Internet.

- While NSS does not recommend switching browsers based on the results of these tests alone, if you currently have a free choice of browser then Internet Explorer 9 offers the most comprehensive protection from these particular threats.
- Exploits were not the subject of this test, and readers should not draw conclusions based upon this one analysis brief.
- Keep up to date with the current version of the browser of your choice as well as third party applications.

Analysis

Given that Chrome, Firefox and Safari all use Google's Safe Browsing API, and Google has publicly stated that it has not withheld data from their Safe Browsing feed, what should end users make of these results?

The following sections provide detailed analysis performed by NSS Labs in order to determine the cause of this discrepancy.

How does Safe Browsing API v2 work (before download)?

Chrome, Firefox and Safari all maintain lists of bad URLs locally. Firefox and Safari maintain the list in a SQLite Database; Chrome maintains the URLs in an undocumented proprietary format.

When a user tries to navigate to a potentially bad URL, the URL is matched against the local blacklist of bad URLs and local whitelist. The final lookup is performed online to get the full hash of the URL. This is achieved by issuing a POST request to the *google-malware-shavar* at *hash suffix server* to get the full hash. Access to the URL is blocked with a warning message if the full hash matches up. This process is fully documented within Safe Browsing API v2⁴ and Chrome, Firefox and Safari all exhibit the same behavior.

⁴ http://code.google.com/apis/safebrowsing/developers_guide_v2.html

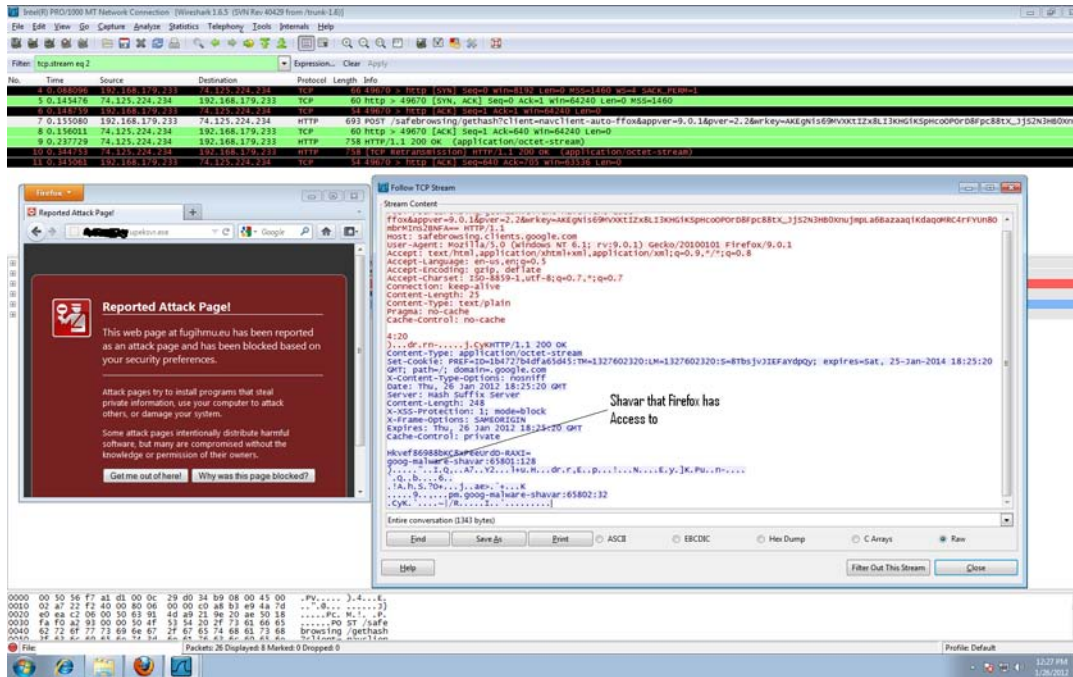
The screenshot displays a network analysis tool interface. At the top, a table lists network packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. Packet 16 is highlighted, showing an HTTP 200 OK response from 192.168.179.234 to 74.125.224.229.

Below the table, a browser window shows a "Warning: Something's Not Right Here!" message. The text indicates that the site "fughmu.eu" contains malware and that Google has found malicious software that may be installed onto the user's computer. A "Go back" button is visible.

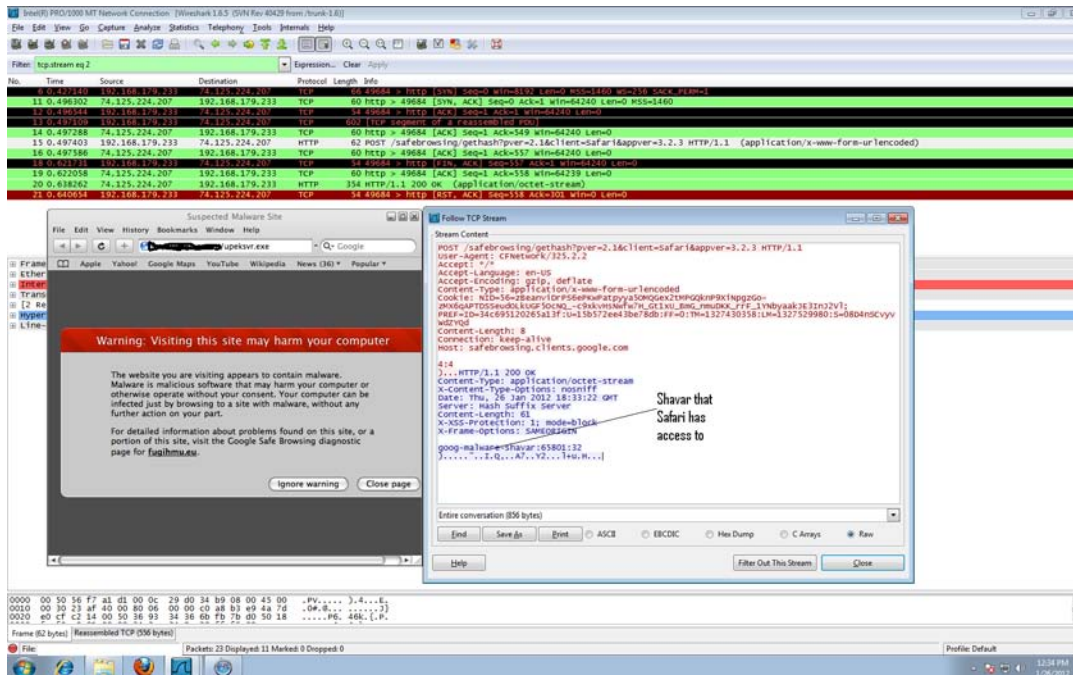
To the right, a "Follow TCP Stream" window shows the raw data for the selected packet. The stream content includes an HTTP 1.1 200 OK response with headers such as "Content-type: application/octet-stream", "Set-Cookie: PkLI-ID=a662435811c2b62;TM=1327603466;LM=1327603466;S=fkuclq15h2k896f; expires=Sat, 25-Jan-2014 18:44:26 GMT; path=/; domain=.google.com", and "X-Content-Type-Options: nosniff". The body of the response contains a warning message in HTML format.

At the bottom, a hex dump shows the raw bytes of the frame, with a note indicating it is "Rassembled TCP (458 bytes)".

Chrome: Malware block based on malware shavar list documented within Safe Browsing API v2



Firefox: Malware block based on malware shavar list documented within Safe Browsing API v2

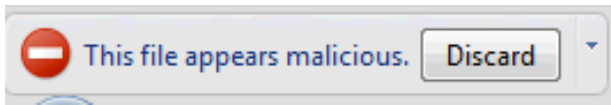


Safari: Malware block based on malware shavar list documented within Safe Browsing API v2

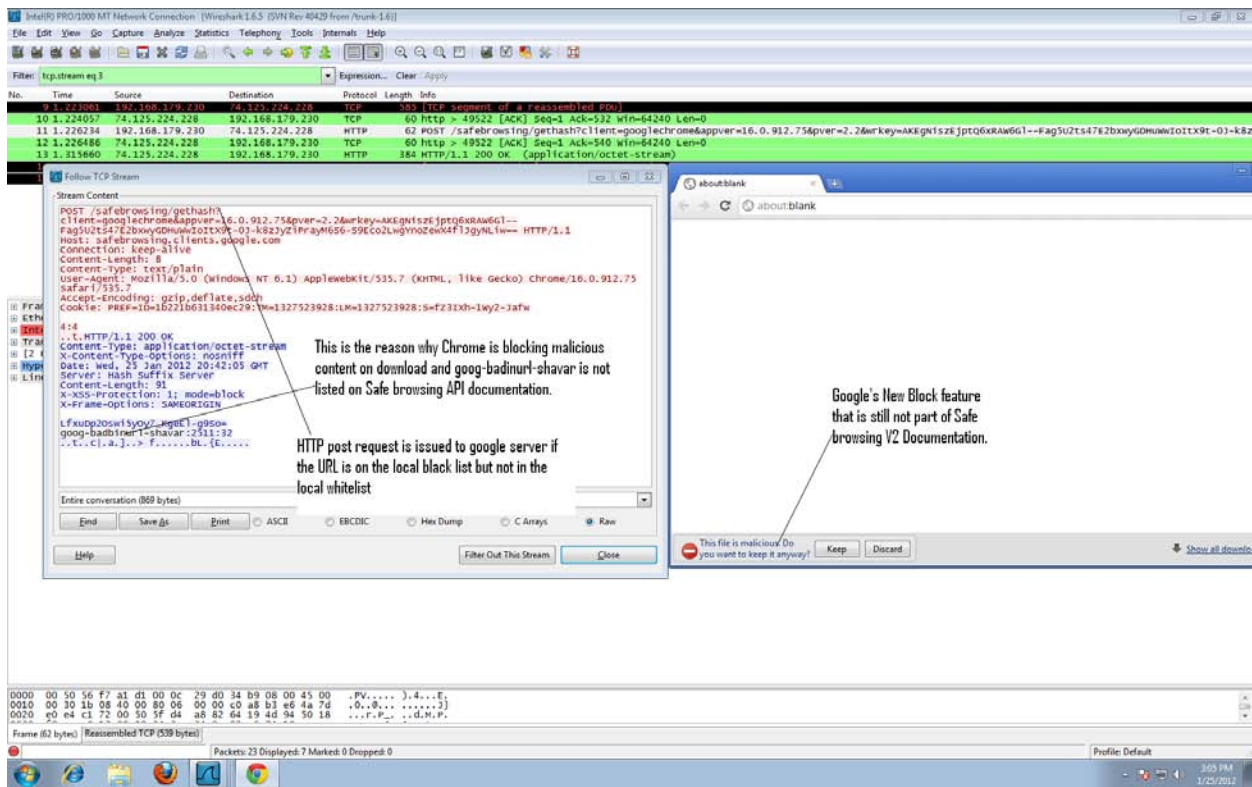
As the above examples demonstrate, Chrome, Firefox, and Safari all have access to the *google-malware-shavar* that is documented as part of Safe Browsing API V2. All three browsers block the same URL. This is well documented and should not come as a surprise to anyone.

Proprietary functionality unique to Chrome

NSS Labs determined that Safe Browsing API v2 includes undocumented functionality that has been integrated into Chrome, but not Firefox or Safari. This undocumented functionality provides reputation services for executable files, or as Google describes it “malicious downloads”.



The new service conducts a lookup *after* download begins; Chrome issues a POST request to the *goog-badinurl-shavar* at *hash suffix server* to get the full hash. If the download is determined to be malicious the user is given a warning saying that “*the file seems to be malicious.*”



The image shows a Wireshark network capture of a browser's communication. The main pane displays a list of packets, with a selected POST request to `/safebrowsing/gethash?client=googlechrome&appver=16.0.912.75&ver=2.2&urkey=AKEgntsz6JptQ6KRAWG1--Fag5U2ts47E2bxyGDUHwTolTX9L-0J-k8z`. The packet details pane shows the request body, including a `Content-Type: text/plain` and a `User-Agent` identifying the browser as Chrome. The packet bytes pane shows the raw data of the request. A secondary window shows the 'Follow TCP Stream' for the selected packet, displaying the raw HTTP request and response. A third window shows a browser warning: 'This file is malicious! Do you want to keep it anyway?' with 'Keep' and 'Discard' buttons. Annotations with arrows point to the POST request in the packet list and the warning window, explaining that the URL is on a local black list but not in the local whitelist, and that this is the reason why Chrome is blocking malicious content on download while the `goog-badinurl-shavar` is not listed in the Safe Browsing API documentation. A final annotation points to the warning window, stating that this is Google's new block feature not yet documented in the Safe Browsing V2 documentation.

Chrome: Malware block on download

Can the new features used by Chrome be accessed with a thin client?

Issuing an HTTP post request⁵ with correct parameters NSS researchers received the following list:

- *goog-badbin-digestvar*
- *goog-badbinurl-shavar*
- *goog-csdwhite-sha256*
- *goog-downloadwhite-digest256*
- *goog-malware-shavar*
- *goog-phish-shavar*
- *goog-regtest-shavar*
- *goog-test-shavar*
- *goog-whitedomain-shavar*
- *googpub-phish-shavar*

While numerous lists are visible to clients, there is no supporting documentation that would allow third parties like Firefox and Safari to implement the new solutions. Looking to further validate and confirm their findings, NSS researchers investigated how each of the browsers maintains its Safe Browsing data.

Chrome Safe Browsing data store

On a Windows 7 Machine, Chrome stores the Safe Browsing information in

“*C:\Users\<username>\AppData\Local\Google\Chrome\User Data*” utilizing 4 different files to interact with safe browsing API. These are:

- safe browsing bloom -> data structure pointing to key elements required by Safe Browsing
- safe browsing bloom filter 2 -> the bloom filter
- safe browsing download -> download store
- safe browsing csd whitelist -> client-side phishing detection whitelist store

Bloom filter file format is:

- 4 byte version number
- 4 byte number of hash keys (n)
- n * 8 bytes of hash keys
- Remaining bytes are the filter data

A simple bloom filter uses a large number (20) of hashes to reduce the possibility of false positives. The bloom filter's hashing uses random keys in order to minimize the chance that a false positive for one user is a false positive for all. Prefixes are sorted and stored as 16-bit deltas from the previous prefix. An index structure provides quick random access, and also handles cases where 16 bits cannot encode a delta. The on-disk format appears as follows:

- 4 byte magic number

⁵<http://safebrowsing.clients.google.com/safebrowsing/list?client=api&apikey=APIKEY&appver=CLIENTVER&pver=PVER&wrkey=MACKEY>

- 4 byte version number
- 4 byte |index_size|
- 4 byte |deltas_size|
- n * 8 byte |&index_[0]..&index_[n]|
- m * 2 byte |&deltas_[0]..&deltas_[m]|
- 16 byte digest

kMagic should be reasonably unique, and not match itself across endianness changes. The value was generated with:

```
static uint32 kMagic = 0x864088dd;
static uint32 kVersion = 0x1;
typedef struct {
    uint32 magic;
    uint32 version;
    uint32 index_size;
    uint32 deltas_size;
} FileHeader;
```

When a user first installs and enables the phishing and malware protection, Chrome contacts Google servers to download the list. It typically takes around 30 minutes, depending upon the available bandwidth, to obtain the full list. However in some cases NSS has observed delays of up to 24hrs for the list to be fully populated.

Due to the proprietary nature of Chrome's reputation store, NSS researchers were unable to determine which lists are maintained.

Firefox Safe Browsing DB

On a Windows 7 Machine, Firefox stores the Safe Browsing information in "C:\Users\<username>\AppData\Local\Mozilla\Firefox\Profiles\<randomname>.default\urlclassifier3" utilizing a SQLite database.

The screenshot shows a window titled "SQL Lite Database Browser" with the file "urlclassifier3.sqlite" open. The interface includes a menu bar (File, Edit, View, Help), a toolbar with various icons, and tabs for "Database Structure", "Browse Data", and "Execute SQL". The "Browse Data" tab is active, showing a table named "moz_tables". The table has four columns: "id", "name", "add chunks", and "sub chunks". There are four rows of data, with the last row selected.

id	name	add chunks	sub chunks
1	test-malware-simple		1
2	test-phish-simple		1
3	goog-malware-shavar	48070-66897	59340-74383
4	goog-phish-shavar	181230-197185	86438-91261

Firefox Safe Browsing DB

This confirms that Firefox maintains the two lists that are documented in Safe Browsing API v2.

Safari Safe Browsing DB

On a Windows 7 Machine, Safari stores the Safe Browsing information in "*C:\Users\<username>\AppData\Local\Apple Computer\Safari\Safe Browsing*" utilizing a SQLite database.

The screenshot shows a window titled "SQL Lite Database Browser" with the file "SafeBrowsing.db" open. The interface is similar to the previous screenshot, with the "Browse Data" tab active. The table shown is named "Lists" and has three columns: "ID", "Name", and "LastUpdateTime". There are two rows of data, with the first row selected.

ID	Name	LastUpdateTime
1	goog-phish-shavar	349124328.050117
2	goog-malware-shavar	349124328.050117

Safari Safe Browsing DB

This confirms that Safari maintains the two lists that are documented in Safe Browsing API v2.

The Tested Browsers

The web browsers were obtained independently by NSS Labs. Generally, available software releases were used in all cases. Each product was updated to the most current version available at the time testing began. The following is a current list of the web browsers that were tested:

- Google Chrome v15.0.874.106 m
- Windows Internet Explorer 9 (build 9.0.8112.16421)
- Mozilla Firefox v7.0.1
- Safari v5.1.1 (7534.51.22)

Once testing began, the product version was frozen in order to preserve the integrity of the test. This test relied upon Internet access for the reputation systems and access to live content. Generally, there is a configurable separation between software updates and database or signature updates, to draw analogies from anti-virus, intrusion prevention, and general software practices.

©2012 NSS Labs, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the authors.

Please note that access to or use of this report is conditioned on the following:

The information in this brief is subject to change by NSS Labs without notice.

The information in this brief is believed by NSS Labs to be accurate and reliable at the time of publication, but is not guaranteed.

All use of and reliance on this brief are at the reader's sole risk. NSS Labs is not liable or responsible for any damages, losses, or expenses arising from any error or omission in this report.

NO WARRANTIES, EXPRESS OR IMPLIED ARE GIVEN BY THE NSS LABS. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT ARE DISCLAIMED AND EXCLUDED BY NSS LABS. IN NO EVENT SHALL NSS LABS BE LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES, OR FOR ANY LOSS OF PROFIT, REVENUE, DATA, COMPUTER PROGRAMS, OR OTHER ASSETS, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

This brief does not imply any endorsement, sponsorship, affiliation, or verification by or with any organizations mentioned in this brief.

All trademarks, service marks, and trade names used in this brief are the trademarks, service marks, and trade names of their respective owners.