Why Silent Updates Boost Security

Thomas Duebendorfer
Google Switzerland GmbH

Stefan Frei
Swiss Federal Institute of Technology
(ETH Zurich)

ETH Tech Report TIK 302
http://www.techzoom.net/silent-updates
insecurity-iceberg@tik.ee.ethz.ch

ABSTRACT

Security fixes and feature improvements don't benefit the end user of software if the update mechanism and strategy is not effective. In this paper we analyze the effectiveness of different Web browsers update mechanisms; from Google Chrome's silent update mechanism to Opera's update requiring a full re-installation. We use anonymized logs from Google's world wide distributed Web servers. An analysis of the logged HTTP user-agent strings that Web browsers report when requesting any Web page is used to measure the daily browser version shares in active use. To the best of our knowledge, this is the first global scale measurement of Web browser update effectiveness comparing four different Web browser update strategies including Google Chrome. Our measurements prove that silent updates and little dependency on the underlying operating system are most effective to get users of Web browsers to surf the Web with the latest browser version. However, there is still room for improvement as we found. Google Chrome's advantageous silent update mechanism has been open sourced in April 2009. We recommend any software vendor to seriously consider deploying silent updates as this benefits both the vendor and the user, especially for widely used attack-exposed applications like Web browsers and browser plug-ins.

Categories and Subject Descriptors

C.4 [**Measurement techniques**]: Miscellaneous
; C.2.3 [**Network monitoring**]: Security

General Terms

Updates, Security, Measurement

Keywords

Web browser, Update

1. INTRODUCTION

Our global scale measurements of Web browsers in use [1] from mid 2008 found that 45.2% of Internet users were not using the latest Web browser version when visiting Google Web servers. If people keep using an outdated Web browser version with known vulnerabilities, they can easily fall victim to any of the millions of malicious Websites that execute drive-by downloads to infect the visitor's computer with malware. In April 2009, Finjan discovered a bot network of more than 1.9 million [2] infected computers, which was built up since February 2009 by using drive-by downloads as primarily infection channel.

In our study [3], we found that in June 2008, the Mozilla Firefox Web browser was having the most effective update mechanism of any popular browser. However, throughout June 2008, at most 83% [1] of all active Mozilla Firefox users were using the latest Mozilla Firefox version. We were wondering if one cannot do even better than Mozilla Firefox by deploying a different update mechanism in a Web browser.

The Google Chrome Web browser [4], which was released to the public as beta in September 2008 and came out of beta in December 2008, is using a so-called *silent update mechanism.* The user currently cannot disable auto-updates in Google Chrome, which is different from any other browser update mechanism in use today. This gave us a great opportunity to evaluate the effectiveness of Google Chrome's update mechanism by comparing it to other Web browsers using the same data source and a similar measurement methodology as used in our two previous Web browser studies.

2. WHY UPDATE EFFECTIVENESS MATTERS

Keeping software up-to-date brings several benefits to the user:

- increased security thanks to timely deployment of security vulnerability fixes
- better software stability thanks to timely bug fixes
- new features that make software more powerful

At the same time, getting all users to work with the latest software release is also advantageous for the vendor:

- most likely happier users due to more stable, more secure applications with additional features
- less support required: only unfixed bugs in the latest version get reported by users
- less testing: engineers don't have to keep testing older versions on newer platforms and with new third party software or drivers

Now, imagine user experience of your software, if a third or more of the user base is not using the latest version as this was the case for Apple Safari, Opera, and Microsoft Internet Explorer in mid 2008. This large user base will never see the

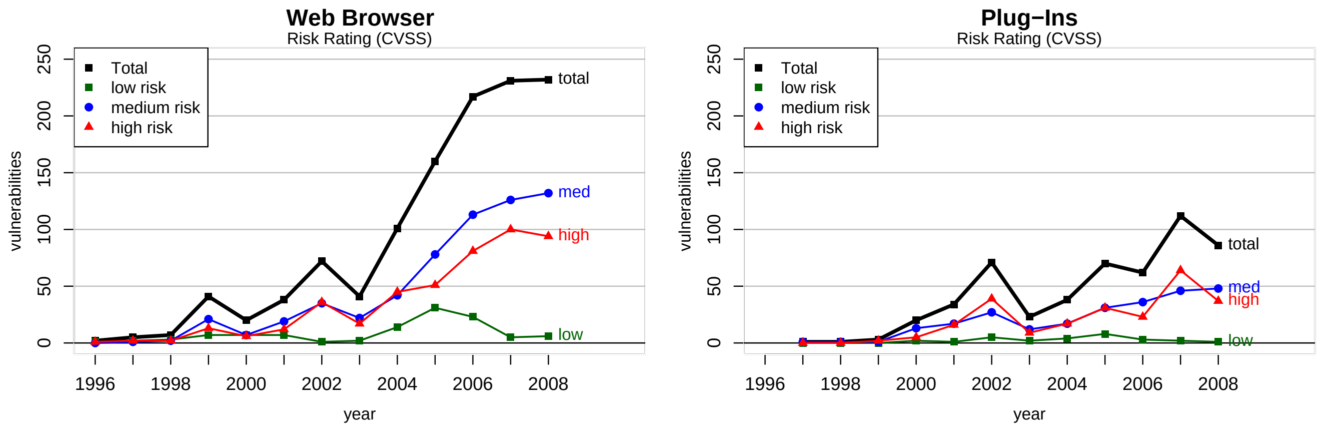

Figure 1: Number of vulnerabilities reported per year from 2000 to 2008 for Web browsers and popular plug-ins split by CVSS risk rating. Source: National Vulnerability Database (NVD)

improvements and new features of the latest version - and will be unnecessarily exposed to old threats.

In the optimal case, all users would always be using only the latest release of a software. However, this also introduces a big drawback: The software gets even more dynamic as each time it is used, it could be a different version with some unexpected features or new behavior. This takes control away from the user and gives it to the software vendor, which some power-users don't like. Furthermore, some developers need to control which version of a software they run, mostly for testing purposes. However, most of this testing would no longer be needed if all users were always using the latest version.

Interestingly, this optimal case is actually widely in use nowadays with software as a service in the form of Web based (e.g. AJAX) applications. Code in many Web application implementations changes frequently without the user even noticing it as most is done "under the hood" and not visible in the user interface. The large majority of Web applications does not even expose a version number to the end user because there's simply no need to care about updates of Web applications for the user. The underlying Web servers and frameworks often expose a version number in the HTTP headers. However, we're speaking here of the actual business logic code built on top of it. Despite the "loss of control" argument, most users have accepted this fact. We think that using silent updates makes sense also for many programs installed on end user systems. In the end, it should not matter for the user where and how the application is installed as long as a high service quality, compatibility and appropriate user data privacy is respected.

3. TRENDS IN WEB BROWSER AND WEB SERVER INSECURITY

Tracking the number of vulnerabilities discovered each year in Web browsers from year 1996 to 2008 as shown in Figure 1 reveals a clear trend: The total number of Web browser vulnerabilities reported in the National Vulnerability Database (NVD) [5], which were rated low, medium or high risk in the Common Vulnerability Scoring System (CVSS) [6], was increasing rapidly year over year since 2003. From 2000 to 2007, the number of new vulnerabilities discovered per year in popular Web browser plug-ins has more than quadrupled. From 1996 to 2008, the most vulnerable plug-ins were Apple Quicktime with a staggering number of 125 vulnerabilities found and Adobe Acrobat with 59 vulnerabilities reported.

On the other side, classical Web servers (i.e. Apache HTTP Server, Microsoft Internet Information Server, Netscape Enterprise Server, IBM HTTP Server and Sun One Web Server) had their last peak of reported new vulnerabilities in 2002 as shown in Figure 2. While classical Web servers became less vulnerable over time, new Web application servers and scripting engines became popular and introduced new vulnerabilities in Web applications. For the PHP scripting engine alone, 116 vulnerabilities were reported in year 2007 alone, without counting vulnerabilities found in PHP based Web applications (like php-nuke, phpbb, Typo3 and many more). Attackers exploit such vulnerabilities in Web applications and misuse them to initiate drive-by downloads of malware to infect visiting vulnerable Web browsers.

Combined with the fact that most vulnerabilities were exploitable from the network and almost none needed local access, the Web browser has clearly become a weak link in the chain of systems that make the Internet work. Extrapolating from the past, it's very likely that Web browsers will have vulnerabilities rated medium and high risk and which are exploitable from remote in the near future as well. A good practice to keep the network and end-users secure is to patch known vulnerabilities as fast as possible. However, to do so, Web browsers are in dire need of a very effective update mechanism or they will lose the battle for securing vulnerable Web browsers before their users fall victim to attackers.

4. UPDATE MECHANISM VARIETY

We consider the different types of update mechanisms in use by software today. The core steps on the client side in an update process are:

- discover the update: learn about availability of the update
- download the update: copy the update files to the local system (after checking authenticity of the update source)

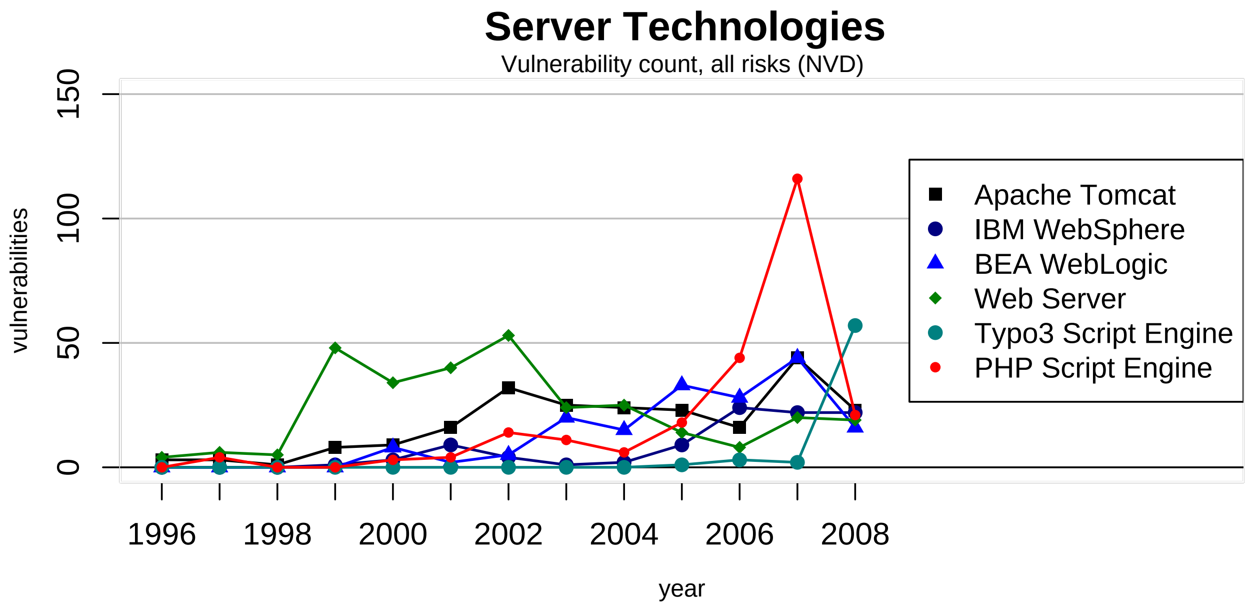

Figure 2: Number of vulnerabilities reported per year from 2000 to 2008 for Web servers and various Web application servers. Source: National Vulnerability Database (NVD)

- install the update: modify local installation with changes introduced by the update (after checking integrity of the update)

- apply the update: execute the software with the changes by the update in effect

While each step looks trivial, the underlying question to ask is how the delay from update availability to when the update is applied by end users is influenced by each step in any specific update mechanism in use. This ultimately reflects in the effectiveness of the update mechanism as measured by the share of users that have updated their software at a given time after the release of a new patch. There are other factors as well such as update policies (i.e. compatibility requirements with existing (e.g. intranet) applications, or a company wide schedule for updates etc.), willingness to update (i.e. a new user interface might let people block an update) and possibly licence cost (not applicable for most Web browsers as they are available for free). We'll briefly summarize how the five most popular Web browsers get updated. Given that each of them uses a different system and update strategy, it's clear that industry has not yet settled on a single best practice yet.

Google Chrome Web browser checks for updates every five hours. It is using the recently open sourced Google update component code-named Omaha [7], which keeps polling for updates even when Google Chrome is not running. To control speed of update distribution and to throttle download resource usage, the update servers report newly available updates only with a certain probability, which is set on the server side per release. Once a new update is found to be available on the server, the client automatically downloads and installs it in the background without prompting the user. The new version of Google Chrome gets applied at the next restart of the browser. At the time of this writing, in April 2009, the user was not even prompted to restart the browser after a new update was ready. Given that the whole update process happens without any user interruption, Google Chrome is said to have a "silent update" mechanism. As of April 2009, the user could not disable update checks. A manual update can be initiated by choosing "About Google Chrome" in Google Chrome's settings menu.

Mozilla Firefox Web browser checks for updates periodically in a frequent schedule while the browser is running. The user can also check for updates manually with the menu command "Help" - "Check for Updates ...". When and how often Mozilla Firefox checks for updates can be set by typing "about:config" in the address bar and changing the variable browser.cache.check_doc_frequency. The default value is 3. The user can choose between once-per-session (0), each-time (1), never (2) and when-appropriate/automatically (3). It's not further specified how often a setting of 3 checks for updates but our daily update share measurements presented in this paper indicate that it is least once a day. In the preferences dialog box, the user can disable update checks for Mozilla Firefox, installed add-ons and search engines. It can be specificed whether Mozilla Firefox should ask what to do if an update is found or if it should be automatically downloaded and installed (the default) and if there should be a warning if any add-ons will get disabled by the update (the default). Even though it says "automatic", the user will still be prompted to accept a newly downloaded Mozilla Firefox version before it is actually installed and applied by a restart of the browser.

Apple Safari Web browser is updated through Apple's "Software Update" service integrated in OS X, which also takes care of other system and application updates. The user can choose to check for updates daily, weekly, monthly or not at all. When updates are available, the user is prompted to initiate the downloads and get them installed. Updates that Apple considers "important" can be chosen to be down-

Web browser	discover update automatically	download update	install update	apply update
Google Chrome	every 5 hours	automatically	automatically; browser keeps running during installation; user is not prompted to restart the browser when new version is installed	with next browser start
Mozilla Firefox	once per session, each-time, when-appropriate, or never	automatically, manually, or never	manually with one mouse click; browser must be closed during installation and will restart afterwards	with next browser start
Apple Safari	daily, weekly, monthly, or never	manually, optionally automatically for important updates, or never	manually (often together with other updates); browser sometimes must be closed during installation	with next browser start
Opera	weekly, or never	manually or never	manually (like a fresh browser install from scratch); browser must be closed during installation	with next browser start
Microsoft Internet Explorer	daily, weekly, or never	automatically, manually. or never	automatically or manually; browser sometimes must be closed during installation	with next browser start

Table 1: Comparison of Web browser update mechanisms.

loaded automatically. Important and new updates are not discovered instantly after availability as the update polling schedule is as set by the user with at most one daily check. During an update, affected applications sometimes have to be closed, which is an annoyance to users. After installation of the update, the next time Apple Safari is started, the new version will be used.

Opera Software's Opera Web browser, by default, checks for updates every week and notifies the user when a new update is available. A user choosing to update his browser is then forwarded to the Opera download Web site, where the update follows the same procedure as if the user were to install Opera for the first time. This update procedure requires serious user activity and typically about ten user decisions on different dialogs, such as choosing the install/update location, clicking the license agreement, closing the active browser, etc. The frequency Opera checks for updates is fixed. The weekly checks can be disabled in the "opera:config" window or directly in the configuration file setting the parameter "Check For New Opera" to 0. The user can manually check for pending updates at any time through an option in the "Help" menu. The upcoming major version of Opera, version 10 now in alpha testing, will update itself automatically as new versions are released [8].

Microsoft Internet Explorer Web browser gets updated through the Automatic Updates service integrated in the Windows operating system since Windows 98 and more recent versions. In Windows XP, the user can choose in the Security Center under "Automatic Updates" settings whether Windows should check for updates daily at a given time or weekly on a specified week day. The user can choose between four options: (1) automatic download and install, (2) automatic download and prompting the user for when to install, (3) automatic notification about updates without downloading and installing them , and (4) turning automatic updates off. The recommended setting is daily automatic update checks, downloads and installs. Windows installations in companies are often configured to only accept updates from a company internal update server at their own schedule. Since October 2003, Microsoft has been following the "Patch Tuesday", the second Tuesday of each month, to release security patches. In rare cases, out-of-band patches are made available in between the patch Tuesdays for highly critical vulnerabilities such as to fix a vulnerability in Microsoft Internet Explorer [9] on Wed, December 17th, 2008, which was believed to have been exploited to infect more than two million computers. After installation of the update, the next time Microsoft Internet Explorer is started, the new version will be used.

We show an overview of the update features in Table 1. Each Web browser software listed here also allows to manually check for updates. All of the described systems poll for updates from update servers and none of them gets updates pushed directly. This is mainly for security reasons, which often prevent Internet servers from contacting clients directly. However, for the user, this does not make a difference besides an additional delay introduced for discovering a new update.

After publication of our browser study [1] in July 2008, we got many write-ins of users explaining to us why they prefer not to update their Web browser. Some users simply don't want to update because updating can be very inconvenient. Most update mechanisms interrupt the user working, some require him to wait for the download, prompt for installation at the most inconvenient time (i.e. during a public presentation), some let the user busily wait during installation as the browser must be closed during this process, and finally some updates break the system or expose the user to a new user interface, which requires time to get used to it or which the user simply does not like better than the old user interface. In addition, there's no guarantee that an update can always be undone without side effects. Especially with larger updates, the benefits of installing a new version sometimes do not outweigh all the troubles an update could cause to the user. If update mechanisms were designed with ease of use and convenience in mind, users would be much more willing to get updates deployed.

In terms of user interaction, only Google Chrome's silent update mechanism does not disturb the user at all. The user is neither disturbed working by getting prompted for download or installation of an update, nor does the user have to quit the browser during installation of the update. The update will simply be applied the next time the user decides to restart the browser. While this totally silent update causes no disruption to the user, it has the disadvantage that the user is not actually made aware of the update and might keep running an outdated version for days or weeks, even though a newer version is already installed on the local system. At least for security bugs, it would seem advantageous to make the user aware of the installed update and ask for a browser restart at the next convenience.

5. MEASURING UPDATE EFFECTIVENESS

To measure update effectiveness, we looked at the percentage of daily active users that use the newly released Web browser version; with 100% being all users of the same major version seen on the same day. By tracking the usage shares over three weeks after a new release, we could determine how fast users update to the latest version and compare the update performance between different releases of the same and other browsers.

We used anonymized logs from Google's world wide distributed Web servers and parsed the HTTP user agent string, which each browser sends to any Web server, when requesting a Web page. The user agent string contains the browser's name and version number as well as some other information. Here are two sample user agent strings of Mozilla Firefox 3.0.8 and Google Chrome 1.0.154.53:

1) `Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.9.0.8) Gecko/2009032609` **Firefox/3.0.8**
2) `Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/525.19 (KHTML, like Gecko)` **Chrome/1.0.154.53** `Safari/525.19`

For eliminating duplicate visits on the same day, we counted only the first Web request by each Web browser with the same Google PREF cookie. We used the Pacific Daylight (Savings) Time timezone as day reference because all but one vendor of the considered Web browsers have their head quarters in this time zone (with the exception of Opera, which is based in Norway). We split the version number in major and minor versions, e.g. Mozilla Firefox 3.0.8 has a major version of 3. Upgrading between major versions is a much bigger hurdle for users as the changes in the software versions are larger and sometimes introduce incompatibilities and user interface changes. For this paper, we focused on the update effectiveness *within the same major version* of various Web browsers. We did not consider "upgrades" between different major versions in our study as they have different characteristics than minor version updates. Microsoft Internet Explorer only reports the major version number and omits the minor version number in the user agent string. The often stated reason for this omission is to reduce information leakage and make it harder for an attacker to select a working exploit for the actual browser version in use. As we have seen drive-by download Web sites trying many different exploits at once, it's unclear how much additional protection this omission really gives. Therefore, based solely on our Web server logs, we cannot determine the update speed of minor versions within the Microsoft Internet Explorer population. However, for four other popular browsers, namely for Google Chrome, Mozilla Firefox, Apple Safari, and Opera, the minor version gets reported.

Some inaccuracies in our measurements are possible, namely due to Web browsers not sending any cookies, which we dropped from our analysis, and those sending a fake user agent string to disguise as a different Web browser. However, the majority of ordinary Internet users are known not to change the default settings of their software. Finally, some geographic regions are covered less as Google has varying popularity in different regions and we only measured visits to Google servers. In first quarter 2009, Google servers performed Web searches for 81% of global users according to Hitlinks [10]. We consider the effects of the mentioned measurement inaccuracies to be negligible, given the global scale and size of our data set.

6. UPDATE CHAMPIONS AND LAGGARDS

We first analyzed the update effectiveness of the four latest non beta releases for Google Chrome, Mozilla Firefox, Apple Safari, and Opera, which were released before mid April 2009, and plotted them against releases of the same Web browser. The plots in Figure 3 show the update effectiveness as percentage of daily active users of a new Web browser version. We plot the shares for the first 21 days after the release and state the Web browser version in the graph legend together with the date of its release. We compared a new release of a Web browser within Web browsers of the same type and the *same major version number.*

After 21 days of releasing **Google Chrome** 1.0.154.48, an exciting 97% share of active Google Chrome 1.x users were using the latest Google Chrome 1.x version. This is by far the best update effectiveness measured for any of the four investigated Web browsers. It's striking how similar the shares increase for different releases of Google Chrome, indicating the statistical robustness of the process measured. The sudden decrease in the usage share of Google Chrome 1.0.154.46 at the end of the first week after release is explained by the availability of the successor Google Chrome 1.0.154.48.

Mozilla Firefox usage share increases are also strikingly similar across different releases. The usage shares increase slightly faster in the first few days compared to Google Chrome but than flatten out way earlier, never reaching more than 85% usage share for the latest version within 21 days of the release. Frequent checking for updates and the rather obtrusive user prompt to install the new Mozilla Firefox update by restarting the browser most likely help to get users to update earlier. Another factor for such a shift is the actual hour of the release on the publicized release date, which can also shift results of daily measurements. The fast initial uptake of a new Mozilla Firefox release is a very positive fact of Mozilla Firefox' update mechanism.

A mere maximum 53% share of **Apple Safari** 3.x Web browser users benefit from an update within three weeks of its release. With newer releases of Apple Safari 3.2.x versions, the update effectiveness drops considerably lower. The reason is that Apple put the bar higher to who is eligible for updates to Apple Safari 3.2.x by requiring Mac OS X Tiger 10.4.11 or higher or Mac OS X Leopard 10.5.5 or higher with Security Update 2008-007 installed. Given that Apple Safari 3.2.1 reaches only 33% on day 21 after release, that's an additional 20% of Apple Safari 3.x users that were left behind since Apple Safari 3.2.x came out. It's not the first time that installation requirements prevent users from

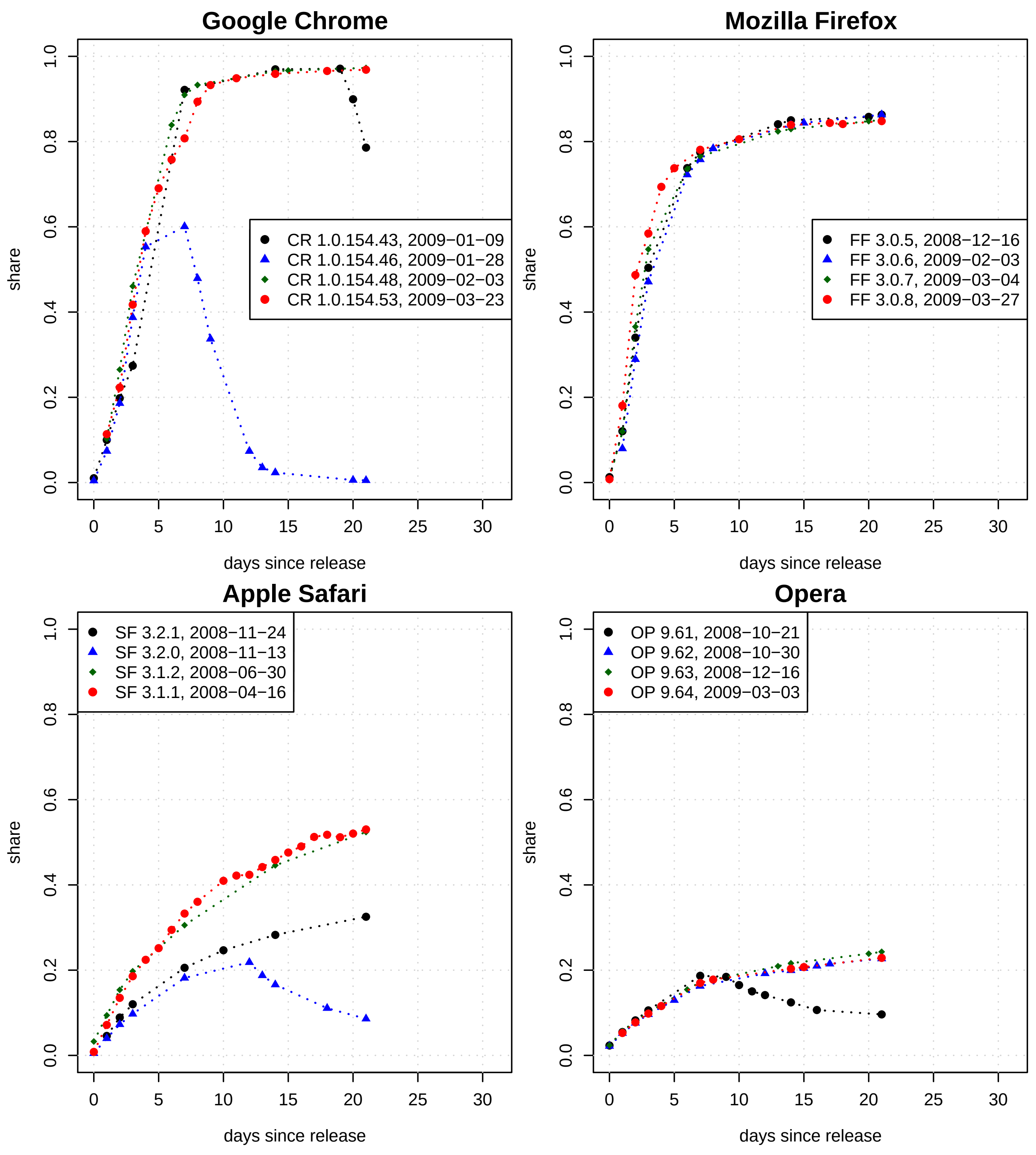

Figure 3: Usage shares of new browser versions within the same browser major version for the first 21 days after release.

updating browsers: users of OS X Panther 10.3, the most recent OS X until OS X Tiger 10.4 was released on April 29, 2005, are limited to Apple Safari 1.3 and Mozilla Firefox 2. Similarly, Windows 9x users have to stick with Mozilla Firefox 2 and Microsoft Internet Explorer 6 and Win 2000 users are limited to Microsoft Internet Explorer 6, the effect of which is measured in [3].

Opera browser users apparently don't update frequently. After three weeks of a new release, a disappointing maximum of 24% active daily users of Opera 9.x have the newest Opera browser installed. It's a pity that 76% of Opera 9.x users currently don't benefit from the security improvements and new features of new Opera versions within three weeks of its release. If some engineering time were spent on increasing update effectiveness instead of working on new features, this would eventually benefit many more users. We also recognize an outlier, namely Opera 9.61, which got replaced after nine days of its release. The upcoming major version of Opera, version 10 now in alpha testing, will update itself automatically as new versions are released [8].

All in all, the poor update effectiveness of Apple Safari and Opera gives attackers plenty of time to use known exploits to attack users of outdated browsers. Figure 4 shows the version mix of Opera 9.x browsers in use relative to all Opera 9.x browsers, 21 days after Opera 9.63 was released. This version reached the highest measured update effectiveness of the last four Opera releases with 24% usage share. Similarly, we give in Figure 5 the shares of Apple Safari 3.x minor versions relative to all Apple Safari 3.x versions in use, 21 days after Apple Safari 3.2.1 got released.

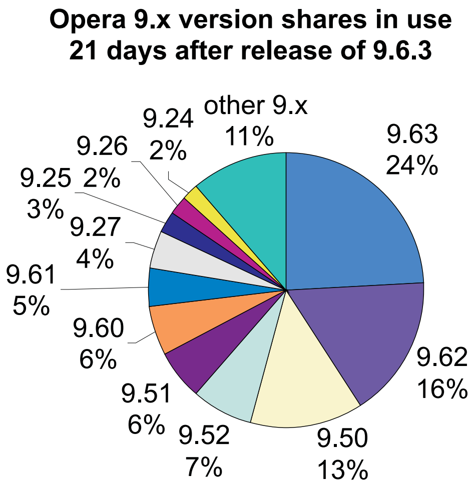

Figure 4: Distribution of Opera 9.x minor versions (relative to all Opera 9.x versions in use) 21 days after release of Opera 9.63. "Other 9.x" summarizes versions with a less than 2% share each.

7. DISCUSSION OF UPDATE EFFECTIVENESS

Comparing the best of each of the four most recent non beta browser releases until mid April 2009 in terms of update effectiveness as shown in Figure 6 gives the following results:

- Google Chrome with silent updates performed best (97% for Google Chrome 1.0.154.48 on day 21).
- Mozilla Firefox with one click updates performed second best (85% for Mozilla Firefox 3.0.8 on day 21) and had the fastest initial update effectiveness in the first five days after release.
- Apple Safari with OS based update checks performed second worst (53% for Apple Safari 3.1.1 on day 21). The more recent versions Apple Safari 3.2.0 and 3.2.1 (at most 33% on day 21) introduced a high dependency on the underlying operating system by requiring a high Mac OS X patch-level for the installation of the update to succeed. This resulted in a considerably worse update performance than Apple Safari 3.1.1 had.
- Opera with manual re-install performed worst (24% for Opera 9.63 on day 21).

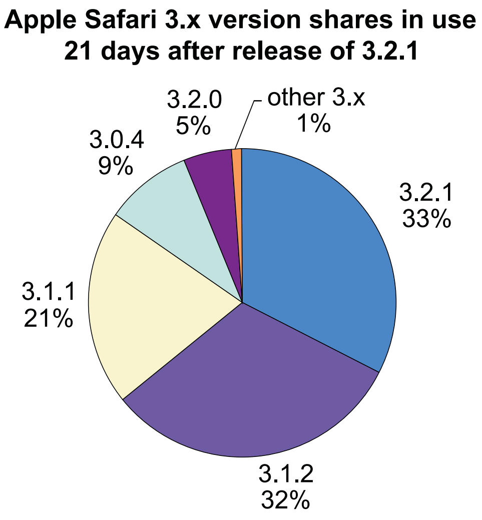

Figure 5: Distribution of Apple Safari 3.x minor versions (relative to all Apple Safari 3.x minor versions in use) 21 days after release of Apple Safari 3.2.1. "Other 3.x" summarizes versions with a less than 2% share each.

As previously discussed, we were not able to measure Microsoft Internet Explorer's minor version update dynamics for technical reasons. Given that Microsoft Internet Explorer is updated through the operating system, much like Apple Safari but with optional auto-download of any browser update (and not just important ones as in OS X), we would expect Microsoft Internet Explorer's update performance to be between that of Apple Safari and Mozilla Firefox.

One intriguing question remains. Why is Google Chrome not reaching 100% usage share with new releases even though it silently performs updates without user interaction and currently without letting the user disable updates? This observation is attributed to a combination of the following effects:

- Google Chrome updates only become effective after installation once the user restarts the browser. Currently, there's no reminder shown the to user that he should restart and apparently, a significant population share does not restart their browser withing three weeks of a new release. Showing a non disruptive notice for making the user restart rather sooner than later could help.
- Some installations (e.g. Internet cafes, test labs, remote desktops) use read-only software images in virtual machines, where a software cannot update itself. After a restart, the same old browser version will still be installed.

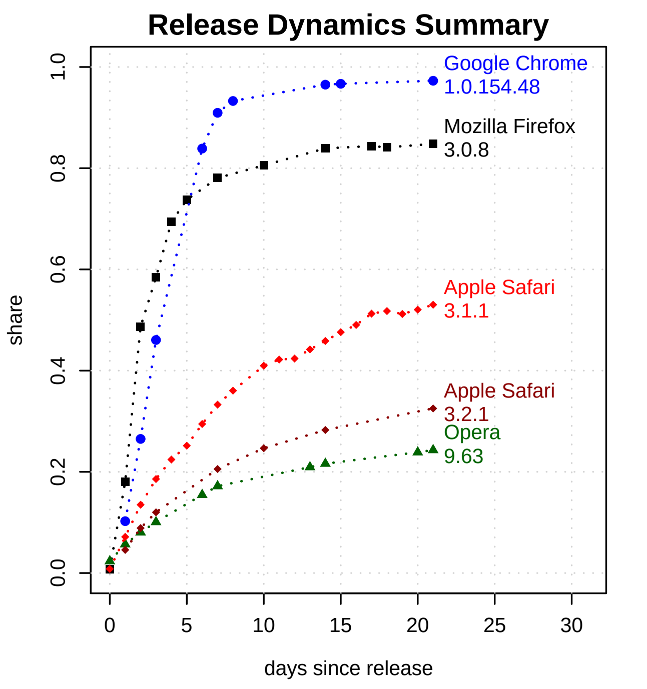

Figure 6: Shares of active users visiting Google Web servers with the indicated Web browser versions measured relative to users of the same browser major version over 21 days after each update was released.

- Insufficient user privileges can prevent the installation of an update. For Google Chrome, no root or administrative privileges are required, so this case should be rare.
- Some firewalls or network filtering devices might block Google Chrome updates.
- Some users might remove googleupdate.exe or prevent it from starting.
- Some tools and tweaked Web browsers might use an outdated Google Chrome user agent string.

8. DISCUSSION OF PATCHING STRATEGIES

Patch management in general (not only of Web browsers) is a complex undertaking – especially in the context of large, business critical infrastructures typically found in organizations. Thus, in the last decade the increasing number of patches has led corporate customers request software vendors for a more predictable schedule to plan the patching of their complex IT environments. As a result, several big software vendors like Oracle or Microsoft today typically release *all* their security patches on a predictable schedule (e.g. every month or quarter). There are no out-of-band patches except for rare, highly critical emergency ones. We consider it suboptimal if patches for heavily attack exposed and highly prevalent applications such as Web browsers and plug-ins with more than a billion users worldwide, are delayed due to the need for a fixed patch schedule for some business users. A fixed patch schedule mainly benefits the patch management processes of larger corporations - organizations which are typically better protected against Internet threats than the masses of individual users. Based on our measurements and the evolution of the threats towards end-users we suggest that software vendors release patches for attack exposed applications, such as Web browsers and plug-ins, as soon as they are available - while keeping a patch schedule for less attack exposed applications. We believe that there is room for a better trade-off to benefit overall security.

9. CONCLUSIONS

Our global measurements have empirically proven that Web browser Google Chrome's silent auto update mechanism is the most effective compared to those of Mozilla Firefox, Apple Safari, and Opera. With 97% latest version share among daily active users three weeks after a new release, it clearly reached the best result. We think that Google Chrome's update effectiveness could be further improved by gently notifying the user about the needed browser restart for changes to take effect after the installation of a new release is done. Furthermore, our results show that Mozilla Firefox' update mechanism has the fastest initial update effectiveness in the first five days after release. Being able to deploy security patches for Web browsers quickly to all end users greatly increases end system security as known vulnerabilities can't be exploited anymore.

In the case of Apple Safari 3.2.1, we have noticed that coupling browser and operating systems and consequently requiring the user to have a recent operating system patch level in order to be eligible to install a browser update should be avoided. Apple left an additional 20% of Apple Safari 3.x users behind with an outdated browser version compared to the previous update to Apple Safari 3.1.2, which did not have these requirements.

Given that today's best performing update mechanism, Google's Updater code-name Omaha [7], was recently open sourced and is free to use for anyone, we encourage others to try it out for their own software. With silent updates, the user does not have to care about updates and system maintenance and the system stays most secure at any time. We think this is a reasonable default for most Internet users. Furthermore, silent updates are already well accepted for Internet Web applications.

10. ACKNOWLEDGMENTS

We would like to thank Mark Larson, Carl Nygaard, Mike Smith, and Linus Upson (listed in alphabetical order) for their feedback on this paper.

11. REFERENCES

[1] S. Frei, T. Duebendorfer, G. Ollmann, and M. May. Understanding the Web browser threat. Technical Report 288, TIK, ETH Zurich, June 2008. presented at DefCon 16, Aug 2008, Las Vegas, USA. `http://www.techzoom.net/insecurity-iceberg`

[2] Finjan. How a cybergang operates a network of 1.9 million infected computers. MCRC Blog - 2009, April 2009. `http://www.finjan.com/MCRCblog.aspx?EntryId=2237`

[3] S. Frei, T. Duebendorfer, and B. Plattner. Firefox (In)security update dynamics exposed. *SIGCOMM*

Comput. Commun. Rev., 39(1):16–22, 2009.
`http://doi.acm.org/10.1145/1496091.1496094`

[4] Google Chrome Web browser.
`http://www.google.com/chrome`.

[5] NIST. National Vulnerability Database (NVD).
`http://nvd.nist.gov`.

[6] Common Vulnerability Scoring System (CVSS) Calculator. `http://nvd.nist.gov/cvss.cfm?calculator&version=2`.

[7] Omaha, the open source Google Updater.
`http://code.google.com/p/omaha/`.

[8] Opera auto update. `http://my.opera.com/desktopteam/blog/index.dml/tag/auto-update`.

[9] Microsoft Security Bulletin MS08-078.
`http://www.microsoft.com/technet/security/bulletin/ms08-078.mspx`, December 2008.

[10] NetApplications.com. Search Engine Worldwide Market Share. `http://marketshare.hitslink.com/report.aspx?qprid=4`, March 2009.