



# **A Week to Remember: The Impact of Browser Warning Storage Policies**

Joel Weinberger and Adrienne Porter Felt, *Google*

<https://www.usenix.org/conference/soups2016/technical-sessions/presentation/weinberger>

**This paper is included in the Proceedings of the  
Twelfth Symposium on Usable Privacy and Security (SOUPS 2016).**

**June 22–24, 2016 • Denver, CO, USA**

ISBN 978-1-931971-31-7

**Open access to the Proceedings of the  
Twelfth Symposium on Usable Privacy  
and Security (SOUPS 2016)  
is sponsored by USENIX.**

# A Week to Remember

## The Impact of Browser Warning Storage Policies

Joel Weinberger  
Google, Inc.  
jww@chromium.org

Adrienne Porter Felt  
Google, Inc.  
felt@chromium.org

### ABSTRACT

When someone decides to ignore an HTTPS error warning, how long should the browser remember that decision? If they return to the website in five minutes, an hour, a day, or a week, should the browser show them the warning again or respect their previous decision? There is no clear industry consensus, with eight major browsers exhibiting four different HTTPS error exception storage policies.

Ideally, a browser would not ask someone about the same warning over and over again. If a user believes the warning is a false alarm, repeated warnings undermine the browser's trustworthiness without providing a security benefit. However, some people might change their mind, and we do not want one security mistake to become permanent.

We evaluated six storage policies with a large-scale, multi-month field experiment. We found substantial differences between the policies and that one of the storage policies achieved more of our goals than the rest. Google Chrome 45 adopted our proposal, and it has proved successful since deployed. Subsequently, we ran Mechanical Turk and Google Consumer Surveys to learn about user expectations for warnings. Respondents generally lacked knowledge about Chrome's new storage policy, but we remain satisfied with our proposal due to the behavioral benefits we have observed in the field.

### 1. INTRODUCTION

An HTTPS error warning might be the last defense between an activist and an active network attacker. As a community, we need security warnings to be effective: clear, trustworthy, and convincing. Prior research has focused on warning comprehension, design, and performance in the field (e.g., [2, 9, 10, 17, 18, 19]). We look at a new angle: storage policies.

Network attackers and benign misconfigurations both cause HTTPS errors. Users may *perceive* warnings as false positives if they do not believe they are under attack. In such a situation, Alice can override the warning and proceed to the website. What happens the next time Alice visits the same website with the same warning? She won't see the warning again until her error exception expires, the length of which depends on her browser's *storage policy*. E.g., Edge saves exceptions until the browser restarts.

Copyright is held by the author/owner. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee.

*Symposium on Usable Privacy and Security (SOUPS) 2016, June 22–24, 2016, Denver, Colorado, USA.*

A browser's exception storage policy has profound usability and security effects. On one hand, consider a user under attack who overrides a warning because she incorrectly believes it to be a false positive. Saving the exception forever puts that user at risk of a persistent compromise. On the other hand, consider a user who repeatedly visits a website with an expired but otherwise valid certificate. Showing the second person a warning every time they visit the site would undermine the warning's trustworthiness without providing much security benefit. Over time, that user will become jaded and might override a real warning.

We are unaware of any prior research into the effects of exception storage policies. As a result, browser engineers have selected storage policies without knowing the full trade-offs. In the case of Google Chrome, the original storage policy was chosen entirely for ease of implementation. Our goal is to provide user research and a security analysis to inform browsers' storage policies in the future.

In this paper, we evaluate various storage policies. Ideally, an optimal policy should maximize warning adherence and minimize the potential harm that could come from mistakenly overriding a warning. We ran a multi-month field experiment with 1,614,542 Google Chrome warning impressions, followed by Mechanical Turk and Google Consumer Surveys (GCS). Based on our findings, we propose a new storage policy that has been adopted by Google Chrome 45.

**Contributions.** We make the following contributions:

- To our knowledge, we are the first to study warning storage policies. We define the problem space by identifying goals, constraints, existing browser behavior, and metrics for evaluating storage policies.
- Using a large-scale, multi-month field experiment, we demonstrate that storage policies substantially affect warning adherence rates. Depending on the policy, adherence rates ranged from 56% to 70%.
- We propose a new storage policy, grounded in our experimental results and a security analysis of available policies. We implemented and deployed this strategy as part of Google Chrome 45.
- We ran surveys about storage policies. Respondents generally did not have strong beliefs or accurate intuitions about storage policies, suggesting that changing a browser's policy would not negatively surprise users.

- We show that researchers need to account for storage policies when comparing adherence rates across browsers. We find that Chrome’s adherence rate could be significantly higher or lower than Firefox’s depending solely on the selected storage policy.

## 2. BACKGROUND

We explain the role of HTTPS errors and why the false alarm effect is a concern for HTTPS warnings.

### 2.1 Purpose of HTTPS errors

HTTPS ensures that web content is private and unalterable in transit, even if a man-in-the-middle (MITM) attacker intercepts the connection. In order to do this, the browser verifies the server’s identity by validating its public-key certificate chain. Browsers show security warnings if the certificate chain fails to validate.

**Threat model.** MITM attackers range in skill level and intent. An attacker could be a petty thief taking advantage of an open WiFi hotspot, or it might be a wireless provider trying (fairly benignly) to modify content for traffic shaping [15]. On the more serious end, governments are known to utilize MITM attacks for censorship, tracking, or other purposes [7, 12, 13]. The attacker might be *persistent*, meaning the target user is continuously subject to attack over a long period of time. Governments and ISPs are examples of entities that have the technical means for persistent attacks.

**False positives.** Many HTTPS errors are caused by benign misconfigurations of the client, server, or network [1]. When people encounter these situations, they want to ignore the error. Although the actual attack rate is unknown, we believe that false positives are much more common than actual attacks. Unfortunately, false positives and actual attacks seem very similar to non-expert end users.

**Warnings.** If there is an HTTPS error, the browser will stop the page load and display an HTTPS error warning (for examples, see Figure 1). Typically, users are able to override the warning by clicking on a button, although this may be disabled if the website serves the HTTP Strict Transport Security (HSTS) or HTTP Public Key Pinning (HPKP) headers<sup>1</sup>. If the error is caused by an actual attack, overriding the warning allows the attack to proceed.

**Storage policy.** Once a user has overridden an HTTPS warning on a website, the browser must persist the user’s exception for some amount of time. The browser’s *storage policy* determines how long the exception is saved for.

<sup>1</sup>The HSTS header specifies that the host is only loaded over valid HTTPS. HPKP allows the server to specify a set of public keys of which at least one is required to be in the certificate chain on any future loads of the server in the browser. User agents generally assume that the presence of these headers imply stronger requirements by the server about the importance of valid HTTPS, and thus HTTPS warnings on such sites are generally made non-overridable.

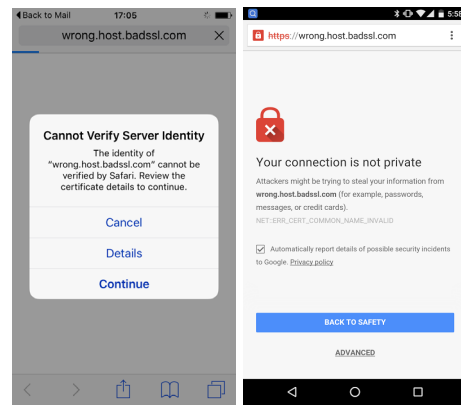


Figure 1: HTTPS error warnings in Safari for iOS (left) and Chrome for Android (right).

### 2.2 The false alarm effect

“Each false alarm reduces the credibility of a warning system,” cautioned Shlomo Breznitz in 1984 [6]. “The credibility loss following a false alarm episode has serious ramifications to behavior in a variety of response channels. Thus, future similar alerts may receive less attention... they may reduce their willingness to engage in protective behavior.” Breznitz was describing the *false alarm effect*, a theory that humans heed warnings less after false alarms. The false alarm effect is long known to decrease attention and adherence to non-computer warnings (e.g., [14, 20]).

Many prior researchers have observed evidence of the false alarm effect for computer security warnings. Nearly all of these researchers have urged industry vendors to decrease their false positive rates to mitigate the effect. Unfortunately, HTTPS errors are still commonly false alarms [1].

In one study of simulated spear phishing, researchers observed a correlation between recognizing a warning and ignoring it [8]. For example, one of their participants said the phishing warning that would have protected him/her “looked like warnings I see at work which I know to ignore” [8]. In a similar study of PDF download warnings, “55 of our 120 participants mentioned desensitisation to warnings as a reason for disregarding them” [16]. Bravo-Lillo et al. found, in two related studies, that participants quickly learned to ignore spurious security dialogs [5, 4].

The false alarm effect is a psychological process that can happen quickly. Anderson et al. watched participants view repeated security dialogs in an fMRI machine [3]. Their participants did less visual processing of the dialogs after only one exposure, with a large drop after thirteen exposures.

Outside of the lab, researchers have seen evidence of the false alarm effect in Chrome users in the field. Chrome users clicked through 50% of SSL warnings in 1.7 seconds or less, which “is consistent with the theory of warning fatigue” [2].

### 3. PROPOSAL

We argue that a browser’s exception storage policy should be chosen with care (rather than for ease of implementation) because it affects end user security and warning effectiveness. We propose a new policy based on desired usability and security properties of HTTPS warnings.

#### 3.1 Goals

Our goals for a storage policy are:

- Reduce the false alarm effect by avoiding unnecessary warnings. The longer the storage policy, the less likely it is that a user will see a repeat warning that they consider a false alarm. In the long run, this should yield increased attention to actual attacks.
- Reduce the cost of a mistake if someone misidentifies an actual attack as a false alarm. If a user fails to heed a warning during an actual attack, we do not want that user to be permanently compromised. The shorter the storage policy, the less time that an attacker has to intercept the client’s connection.
- Avoid unpleasantly surprising users.

A keen reader may notice that the first two goals are diametrically opposed. To reduce the false alarm effect, we should *increase* how long exceptions are stored; to reduce the cost of a mistake, we should *decrease* how long exceptions are stored. In Section 4, we perform a study to find a policy that satisfies both constraints as much as possible, while acknowledging that neither can be completely satisfied.

#### 3.2 Analysis of existing options

There is no industry consensus for how long HTTPS error exceptions should be stored, and existing browser exception storage policies do not meet our goals. We tested browser storage policies as of February 2016 (Figure 2). With the notable exception of Firefox, browser vendors appear to have selected their storage policies based on ease of implementation, which sometimes results in the same browser having inconsistent policies across platforms.

**Browser session.** The most common storage policy is to save exceptions until the browser restarts, either by closing the browser or closing all window instances of the current

Browser	OS	Storage policy
Chrome 44	Windows	Browser session
Safari 9	Mac	Browser session
UC Browser 10	Android	Browser session
Edge 20	Windows	Browser session
Firefox 44	Windows	User choice (browser session or permanent)
Safari 9	iOS	Permanent
UC Browser 2	iOS	Permanent
UC Mini 10	Android	Overriding not allowed

**Figure 2: Browser exception storage policies. This covers Google Chrome, Apple Safari, Microsoft Edge, and Mozilla Firefox, as well as UC Web’s three browsers, which are popular in South and East Asia.**

profile. A session-based policy yields unpredictable but typically short storage lengths. Although a browsing session can last anywhere from five minutes to a month, we know that the average Chrome browsing session lasts slightly less than a day. False alarms could therefore still be daily occurrences for people who need to interact with misconfigured websites.

From a technical perspective, this is the simplest policy to implement: the user’s decision is saved as an in-memory map of hostnames to exception state. For Chrome, engineers chose this policy in large part because it was very easy to implement and the trade-offs associated with the storage policy were unknown; we guess the same decision making process might have been used by other browsers.

**Permanently.** The next most common storage policy is to always save exceptions permanently. This policy is also easy to implement in browsers that store other per-website preferences permanently in a preferences file. Permanently storing exceptions reduces the false alarm effect, but the cost of a mistake is also permanent.<sup>2</sup>

**A choice.** Firefox is the only browser to explicitly give users a choice between two storage policies. By default, an exception is stored permanently. However, the user has the option to store the exception only until browser restart. Firefox users choose the shorter option 21% of the time [2]. Although we like the idea of a choice, browser vendors still need to decide what the options and default are.

**Not applicable.** UC Mini for Android doesn’t let users override HTTPS errors, so there is no need for storage. This prevents people from accessing misconfigured websites at all.

#### 3.3 Our proposal

In contrast to the above existing options, we propose a new, time-based storage policy:

- Store exceptions for a fixed amount of time that is not forever. The amount of time should empirically minimize the cost of a mistake and false alarm frequency.
- Delete stored exceptions when we think users will expect it, for example when clearing browser history or closing a private browsing session.
- If a user ever encounters a valid certificate chain for a website, forget any previously stored exceptions for that website. This can occur when someone proceeds through a warning in the presence of a transient attacker and then later reconnects from a safe network. Forgetting the exception in this situation should reduce harm without increasing the false alarm rate.

In Section 4, we test this policy with several different configurations and compare it to other, existing strategies.

<sup>2</sup>Browsers that provide a “permanent” storage strategy do generally provide a way to remove an exception once granted, but the difficulty in undoing this decision depends on the browser. In Firefox 44, for example, it requires going to a special “Certificates” menu several levels into Preferences under “Advanced” settings. Then one must manually curate a list of server certificates to find the one for which there an exception was earlier granted, and then the user must explicitly choose to delete it.



## 4. FIELD EXPERIMENT

We ran a large-scale field experiment to determine whether the time-based storage policy has merit and, if so, the ideal length of time for a time-based storage policy.

### 4.1 Measurement

We want to know whether there is a length of time that minimizes both the false alarm effect and cost of a mistake. We cannot measure either property directly because we do not know which HTTPS errors are false alarms or mistakes. However, we can use the warning adherence rate and regret rate as proxies of our desired properties.

**Adherence rate.** Chrome already uses telemetry to record important warning metrics in aggregate, including adherence. *Adherence* is the rate at which people heed the warning’s advice to not proceed to the page. We desire high adherence rates. A low adherence rate is a sign that users are experiencing warnings that they consider false alarms.

**Regret rate.** How often do users change their mind about whether it’s safe to override a warning? If someone repeatedly overrides the same warning, then we should stop showing them that warning. On the other hand, consider someone who overrides a warning on Tuesday but then adheres to that warning on Thursday. We view this as an indication of regret — that the user’s original decision to override the warning was a mistake. We don’t want to store mistakes for any longer than necessary. If a long time period has a high regret rate, it is inferior because it perhaps is preventing users from changing their minds sooner. We acknowledge that our regret rate is an imperfect metric because it does not actually measure users’ *feelings*. However, it is meaningful when applied as a comparison tool across experimental conditions because it allows us to see changes in behavior.

Thus, we deem a storage policy as superior if it has a high adherence rate and low regret rate. A strictly superior strategy would be one that did not change the regret rate at all, but increased the adherence rate.

### 4.2 Experiment structure

**Groups.** We tested six policies: one session-based policy, three short time periods (one day, three days, one week), and two long time periods (one month and three months). In the first round of our experiment, we tested only the session-based and short time policies. After that round was successful, we added the long time periods. Our groups and metrics only apply to overridable HTTPS error warnings. Errors that cannot be overridden (due to HSTS or HPKP) are excluded from our experiment.

**Assignment.** We set the number of Chrome users in each experimental group to the same small percentage. The experiment was done across all Chrome platforms<sup>3</sup>. Clients were randomly assigned into experimental groups, and their pseudonymous telemetry data was tagged with the group name. Telemetry data was collected only from Chrome users who opted in to Chrome user metrics.

**Length.** Regret rates cannot be measured until exceptions

<sup>3</sup>Windows, Mac, Linux, ChromeOS, Android, and iOS.

```
host string : {
  fingerprint string
  decision_expiration_time uint64
  guid string
}
```

Figure 3: Decision memory structure

begin to expire, which happens at the time determined by the policy length. So to collect useful data, we let the experiment run for three months on Chrome’s stable release channel, but discarded the data. This warm-up gave the longest strategy time for decisions to expire initially so we could measure changes in user behavior and regret rates. At this point, we collected warning impressions for 28 days.

### 4.3 Implementation

We describe how we implemented the storage policies.

#### 4.3.1 Session storage policy

Chrome’s original implementation is an in-memory map from hostname to a map of certificate and policy decision. The decision is an **enum** of 3 possible values: **ALLOWED**, **DENIED**, **UNKNOWN**. They respectively represent a certificate error that was allowed by the user, one that was denied, or one in an unknown state. In practice, the saved state is either **ALLOWED** or **DENIED**.

If a certificate error is encountered, the networking stack asks the warning manager for the user’s preference. If the user has already allowed the error for this particular host, the warning manager tells the networking stack to allow the connection to continue. Otherwise, a warning is shown. If the user overrides the warning, the warning manager will add the decision to the map.

All profiles receive their own map, so decisions do not carry between profiles. Since this map is in-memory, it is reset when Chrome completely shuts down. On restart, users will be re-asked for any decisions they previously granted.

#### 4.3.2 Time-based storage policies

With a time-based storage policy, exceptions need to persist through browser restarts. This means that exceptions need to be saved on disk. We used an internal Chrome API named “Content Settings,” which stores persistent preferences on a per-profile basis. For example, user preferences about geolocation use and plug-ins are stored in Content Settings. We consider an error exception for a website to be a type of website preference.

Content Settings are stored and retrieved by hostname. We thus map a given hostname to a set of structures containing metadata about individual exceptions granted by the user. Figure 3 shows the Content Setting structure for storing certificate exceptions. It contains:

- `host` is the hostname where the exception was made.
- `fingerprint` is a SHA-256 hash of the certificate and full certificate chain that contained the error. We save the hash to individually identify each error.
- `decision_expiration_time` is the Epoch Time in seconds of when the decision expires. If the certificate in question is checked again, we check the expiration time to see whether the previous exception is still valid.
- `guid` is a globally unique identifier set at browser session start. We use this to address a complexity that arose from sharing code between the time-based policies and the session-based policy. The Content Settings API doesn't allow the caller to know whether a setting was made this session or a previous session. Since Chrome may not cleanly shutdown (for example, if it is force killed or if the machine resets), the per-session exceptions cannot be reliably cleaned up when the session is over. It is tempting to clean up the settings on browser start, but this is problematic for measuring the regret rates: if the settings are cleaned up, and then the same certificate is received, there is no way to know if the user previously made an exception. The solution is to create a per-session globally unique identifier (GUID) which is stored with all the exceptions stored in Content Settings. Then, if the browser session restarts for any reasons, all of the old exceptions are still stored, but they will reflect an old GUID, so it is known that they are "expired" (i.e. they were created in a previous session).

### 4.3.3 History

When stored on disk, exceptions contain hostnames and certificate fingerprints. They are potentially privacy-sensitive since they reveal information about the user's browsing habits. Our implementation therefore needs to take care with how exceptions are handled.

Profiles are the mechanism for storing settings, state, and history of the current user. However, Incognito profiles, also known in other browsers as "Private Browsing," do not record state about the user past the current session. In general, Chrome does a best effort to not store user history in permanent storage. Thus, Chrome does not save any Content Settings for Incognito profiles to disk.

Additionally, there is a general expectation that history-resetting activities should delete site visiting activities. Since this is an indirect type of history recording, it is necessary to make sure certificate exception Content Settings are erased when history is cleaned up. Chrome internally provides a `BrowsingDataRemover` API which is called whenever browsing data or history is cleaned up. The certificate exceptions Content Settings are cleared whenever this API is invoked.

```
EXPIRED_AND_PROCEED
EXPIRED_AND_DO_NOT_PROCEED
NOT_EXPIRED_AND_PROCEED
NOT_EXPIRED_AND_DO_NOT_PROCEED
```

Figure 4: Events when a certificate exception is encountered, used to calculate the regret rate.

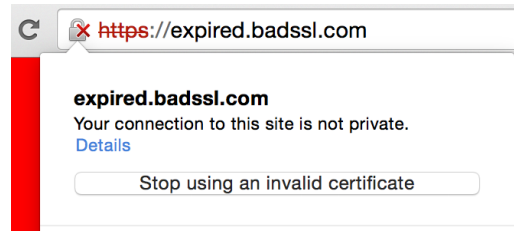


Figure 5: The button we added to let experiment participants revoke exceptions.

### 4.3.4 Analytics

To do the study, we must measure the adherence and regret rates. The adherence rate is already recorded by Chrome, so we had to add the regret rate. This is calculated by recording (a) the decision made when a certificate error is encountered, and (b) the prior state of that certificate exception. Figure 4 shows the recorded events. The `EXPIRED_*` events indicate that the identical certificate error had been encountered in the past, while the `NOT_EXPIRED_*` events indicate the opposite. The `*_PROCEED` events indicate an ultimate decision to create an exception for the error, while the `*_DO_NOT_PROCEED` indicate the opposite. These measurements are all taken in relation to the user's interaction with the HTTPS warning page.

## 4.4 Ethics

Running a security field experiment inherently has risks, as do all security engineering changes. In this case, the primary risk was that adherence or regret rates could suffer in undesirable ways. A secondary risk was that saving a preference to a local file might have an impact on user expectations of local privacy. However, all of our experimental treatments fell within the bounds of other browsers' behavior (since other major browsers have both very short and very long storage policies). We believed the small risk was worth the potential benefit of a new, improved policy.

Still, we were cautious. We took steps to limit any potential harm that could come from the experiment:

- We monitored key statistics as we ran the experiment. For example, we monitored the average number of warning impressions to watch for any sudden large increases, which could be an indication of accidentally desensitizing users to Chrome's warnings. We also observed how often users utilized the "Revoke" button in the page info bubble to make sure users were not explicitly changing their minds often. We could have immediately stopped the experiment via server-side controls if we had believed it necessary.
- We slowly rolled out the experiment to progressively larger groups of users, beginning with pre-release ver-

sions of Chrome. Pre-release Chrome users are developers, power users, and other people willing to trade inconvenience for cutting edge features. When we progressed to stable, we slowly ramped up group sizes.

- We started with three short time periods that were similar to the average Chrome user session: one day, three days, and one week. Initially, we did not do long groups in case the regret rates were too high. Once we saw that regret rate changes were small in the first three groups, we added the two longer groups.
- Previously, users could force Chrome to revoke an exception by restarting. We didn't want to take away this control, so we added a button to the page info bubble (Figure 5) to let users revoke an exception. Additionally, it resets all socket connections for the current browser session to make sure that any exceptions already granted in the networking layer are reset.
- Our implementation provides an additional local history entry for websites with exceptions, but it is similar to regular history. According to our proposal, clearing history now also deletes error exceptions.
- Incognito mode should not persist anything new to disk, so exceptions made in Incognito mode are forgotten once the last Incognito tab is closed.

We did not debrief study participants. Given the low level of risk and our cautious experimental rollout, we did not feel that debriefing was necessary. Furthermore, debriefing notices are infeasible for small, low-risk field experiments. We run many in-product experiments in the course of improving and rolling out new security features, so debriefing notices would be frequent and tiresome. Instead, we prefer to design our experiments to be low-risk. If we had felt that the potential for harm was great enough to merit debriefing, we would have run a lab study instead of a field experiment.

Our experiment was internally reviewed prior to launch in a process that included security experts, a privacy expert, and an experimental research expert.

## 4.5 Limitations

We believe that our data is representative and well-defined. However, there are limitations and potential sources of bias.

**Sample bias.** Since our metrics were collected via Chrome's user metrics analytics (UMA) opt-in program, our data is biased towards users who have chosen to have events anonymously collected. This could mean, for example, that there is a bias towards users who are more or less privacy sensitive, affecting the rate that they adhere to warnings as compared to the general population. However, a large fraction of the population opts in, and we examined millions of warning impressions during the full course of the experiment.

**Metrics.** We are using adherence and regret rates as proxies for actual human desires and intent. It is impossible for our large-scale metrics to precisely capture actual human meaning. For example, imagine that Alice views a warning, gets distracted by her dog, and then overrides a new impression of the same warning once she returns to her task. The initial adherence does not mean the warning worked. In the

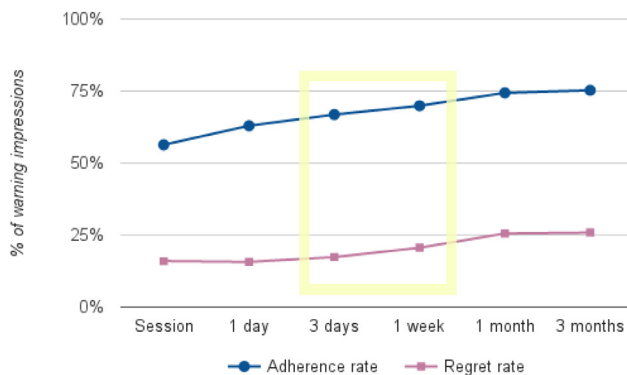


Figure 6: Results of different storage policies.

reverse, imagine that Alice overrode a warning on Tuesday but got distracted when she saw it again on Thursday. She is not actually expressing regret for her Tuesday action. This same limitation holds true across all of our conditions, and we expect the same amount of noise for all conditions.

**Continuous measurement.** Once we choose a strategy for deployment, we can keep our metrics in place to continuously measure adherence and regret to look for unexpected changes. However, we cannot continuously run a full experiment for *all groups* to know if our initial experimental results permanently hold. Because real users are affected, we must choose a system that we think is safest and most usable for our users. Unfortunately, this means that while we can see if our initial results remain for our chosen strategy, we cannot know if they would remain for the other groups.

**Over-representation.** We do not differentiate between users who see many warnings and users who see few warnings. Our statistics are averaged across all users within a treatment group, so users who see many warning impressions will be over-represented when averaging across impressions. Given the scale of our experiment, we do not expect a confound because different types of users should be evenly distributed across experimental groups.

## 4.6 Results

Table 1 shows the impact of different storage policies on 1,614,542 warning impressions. Our experimental data substantiates two hypotheses:

- Storage policies matter. We see large differences in adherence and regret rates across policies.
- Storage length correlates with both adherence rates and regret rates.

We see the biggest difference by comparing the two extremes. Participants in the three-month group saw an increase in adherence from 56% to 75%, as compared to the session-based policy. At the same time, the three-month group's regret rate increased from 16% to 26%.

While we were pleased to see the adherence rate increase with the longer storage policies, we recognize the cost. The longer Chrome stores exceptions, the more likely the user is to reverse their decision once the exception expires.

	Session (baseline)	One Day	Three Days	One Week	One Month	Three Months
Adherence rate	56.35%	62.96%	66.82%	69.88%	74.38%	75.28%
Regret rate	15.98%	15.67%	17.35%	20.59%	25.56%	25.86%
Difference in regret from baseline	-	-0.31	1.37	4.61	9.58	9.88

**Table 1: Results of different exception storage policies. The difference in regret from baseline is simply the baseline’s regret rate subtracted from the policy’s regret rate.**

## 4.7 Choosing a new policy

Following the experiment, we needed to select a new policy for Chrome. Figure 6 highlights the most promising candidates. We ultimately chose the one-week policy.

Our main aim is to raise the adherence rate for Chrome’s HTTPS error warnings. With only this constraint in mind, we would select the three-month policy. The results show, as expected, that the longer the policy, the greater the adherence. However, the increase in adherence also brings an increase in regret rate: the three-month policy yielded a 9.88 point increase in the regret rate (Table 1). We are not willing to accept such a large increase to the regret rate.

To strike a balance between the two conflicting constraints, we decided that we would accept up to a 5-point increase from the baseline’s regret rate. Of the policies that meet this requirement, the one-week policy has the greatest adherence gains. Both the one-month and three-month policies have much larger regret rate increases, while the one-day and three-day policies have lower adherence rate gains.

We do not assert that this is the objectively best choice for a storage policy. All of the policy choices require a trade-off, and different companies may weigh adherence and regret rates differently. For example, someone who is willing to tolerate a higher regret rate would likely choose the three-month policy. Going forward, as we monitor Chrome’s metrics, we plan to re-evaluate this trade-off.

## 4.8 Deployment

We launched the one-week policy as part of Google Chrome 45, in September 2015. Post-launch, the policy is working well for the general population. Looking at 9,318,975 warning impressions, we see an adherence rate of 71.79% and a regret rate of 18.20%. To our pleasant surprise, the policy yielded a slightly higher adherence rate and slightly lower regret rate for the general population as compared to the one-week experimental group.

## 5. USER EXPECTATIONS

One of our initial goals was to avoid unpleasant surprises, which requires understanding user expectations. Several months after Chrome adopted our week-long storage policy in Chrome, we collected user feedback to either confirm or question our decision. Do users *have* expectations? Does our newly adopted proposal meet those expectations?

### 5.1 Method

We surveyed 1,327 people about Chrome’s exception storage policy. First, we asked 100 Mechanical Turk workers to tell us about the storage policy in their own words. Based on those responses, we designed multiple-choice questions and gathered 1,227 Google Consumer Survey responses.

#### 5.1.1 Mechanical Turk

**Questions.** The survey contained three questions, which intentionally did not mention security:

1. Which Internet browsers do you use at least once a week?
2. Imagine that you saw this error page while trying to open a website in Chrome. [Image.] If you clicked ‘Proceed’ on the error page, how long would Chrome remember your decision for?
3. Have you ever seen this error page before, in Chrome?

A screenshot of the questions is available in Appendix A.1.

**Screening.** We limited the survey to Mechanical Turk workers in the US, and we screened for Chrome usage. The survey was advertised as “Chrome users - Survey about error pages, takes about 4 minutes,” with the goal of attracting survey respondents who use Chrome. We ran the survey until we collected 100 responses from people who said they use Chrome at least weekly according to the first question. We paid other respondents but discarded their responses. We did not receive any nonsense or garbage responses.

**Coding.** One researcher coded the short answer responses. The researcher did one round of open coding, developed a codebook, and then applied a fixed codebook to the responses. Since the responses are short and straightforward, we did not have a second researcher duplicate the codes.

**Payment.** We paid respondents \$0.80 to complete a survey that took between one and four minutes. This amount was chosen to reflect a minimum hourly wage of \$12.

#### 5.1.2 Google Consumer Surveys

**Questions.** We ran two questions as separate surveys, each accompanied by an image of an HTTPS warning:

- If you clicked ‘Proceed’ on this error page, how long would Chrome remember that decision for? (Response options: Once, while I’m using the website; A week; Until I clear my history; Forever; I don’t know)
- If you clicked ‘Proceed’ on this error page, how long would you WANT Chrome to remember that decision for? (Response options: Once, while I’m using the website; A few hours or days; Until I clear my history; Forever)

Response options were randomly reversed, with “I don’t know” pinned as the bottom answer for the first question. Screenshots of the questions are available in Appendix A.2.



Session	58%
Period of time	19%
Browser cleared	5%
Forever	13%
Don't know	5%

**Table 2: How long would Chrome remember your decision for? Mechanical Turk short answers.**

	AU	US
Once, while I'm using the website	20%	9%
A few hours or days	6%	4%
Until I clear my history	16%	17%
Forever	9%	10%
I don't know	49%	60%

**Table 3: How long would Chrome remember your decision for? GCS multiple choice responses.**

**Screening.** We requested 600 responses for each question, split evenly between Australian and American respondents. We received 300 for each category except for the first one from Australia, where we received 327 responses.

**Payment.** Respondents were not directly paid. Google Consumer Surveys on desktop are displayed on websites in lieu of paywalls. Respondents received free access to website content after completing the survey.

## 5.2 Ethics

We did not ask for any personally identifiable or sensitive information. Participants were compensated for their time in a way suitable for each survey platform.

## 5.3 Results

We conclude that respondents do not have strongly held beliefs about Chrome's exception storage policy, and preferences are split between session-based and longer policies.

### 5.3.1 Beliefs about current behavior

**Categories.** The short answer responses fell into five categories, which we used for the multiple choice questions:

- *Session.* The response specified a period of time that's similar to a session-based policy. This includes saving it once, for a very short period of time, until restarting, or until closing the window.
- *Period of time.* The response talked about a period of time that lasts longer than a typical browsing session on a website. For example, "a week" or "30 days."
- *Browser cleared.* The respondent mentioned "clearing" something (for example, "until your browsing history is cleared," or "until you clear your cache").
- *Forever.* A synonym of "forever," like "always."
- *Don't know.* The respondent couldn't answer the question. For example, "not sure," or "I don't know."

**Correctness.** Few respondents correctly identified Chrome's current storage policy, even though it had been in place for

Once, while I'm using the website	58%	AU	51%
A few hours or days	12%	US	13%
Until I clear my history	22%		18%
Forever	8%		18%

**Table 4: How long would you want Chrome to remember your decision for? GCS responses.**

several months. We find that most respondents lack preconceived beliefs about Chrome's storage policy, although they can make reasonable guesses when incentivized.

A majority of the Mechanical Turk respondents incorrectly said the storage policy is session-based (Table 2). Although incorrect, this is a reasonable guess; Chrome exhibited this behavior until several months prior to the survey, and other browsers have session-based storage policies. 95% of the Mechanical Turk responses matched feasible potential policies. We therefore conclude that non-expert browser users are capable of reasoning about exception storage policies — when paid to pay attention to a survey.

In reality, browser users are not paid to pay attention to our question. Warnings interrupt people who are trying to complete another task. As a result, their attention is split between the warning and the other task. Consumer Surveys are similar because they interrupt respondents en route to a desired website. In this context, people struggled to answer the question. Approximately half of GCS respondents said they didn't know the answer, and the response rate was low (2.4% in Australia, 6.8% in the United States). This suggests to us that respondents found this question too difficult to answer quickly, meaning they have no strongly held, preconceived belief about current exception storage policies.

**Defining a session.** Mechanical Turk respondents had varying definitions of a "session." We assigned one or more secondary codes to the session-related responses, depending on the type of session the respondent described. Of the 58 session-related responses:

- 26 referred to storing the exception once ("1 time")
- 14 explicitly used the word "session" ("that session only")
- 10 talked about the lifetime of a tab ("till I close the window")
- 6 listed very short time periods
- 5 mentioned restarting ("until I restarted the browser")

All of these responses relate to the lifetime of a browsing session, but they each have different properties in practice. For example, tab lifetimes are generally much shorter than the time between browser restarts.

### 5.3.2 Policy preferences

We asked GCS respondents to choose their preferred storage policy, and their answers were split (Table 4). Half of respondents preferred a session-based storage policy, but the other half expressed a preference for the current time-based

strategy or longer. This leaves us with no clear consensus, although favoring the previous session-based policy.

Notably, a fifth of respondents expected clearing their history to revoke an exception. Chrome's previous strategy did not do this, nor do other browsers. We are glad we added it because it appears to be a common expectation.

## 6. IMPLICATIONS

We discuss the main lessons learned from our experiment and surveys, and give suggestions for future work.

### 6.1 Storage policies matter

Changing a warning's storage policy has almost as large an effect on adherence as completely changing the warning's UI. In our experiment, we saw a 19 percentage point difference between different storage policies (56% to 75%), which is huge! For comparison, Chrome researchers raised adherence by 25 percentage points with a full text and design overhaul [10]. Our work demonstrates that exception storage policies are an important part of warning interaction design, and we see enormous potential in this line of research.

We hope to motivate further research into storage policies. Historically, research into warning effectiveness has primarily focused on the warning's content or effectiveness [10, 17, 19, 18]. Storage policies have received little attention aside from brief mentions in two of our recent projects [2, 11]. Other warnings' storage policies might also benefit from changes, or there might be more clever storage policies that outperform the ones we tested.

### 6.2 Our proposed policy works

Our proposed storage policy reduced the number of likely-unnecessary warnings. Warnings should be meaningful, justified, and rare. If the browser knows with a good degree of certainty that a user will not adhere to a warning, and there is a reasonable chance that the user's decision is correct, then the browser should not show the warning. When we reduced the number of unnecessary warnings, the overall adherence rate improved significantly with little cost to the regret rate.

We believe in removing unnecessary warnings because it increases the salience and trustworthiness of the warnings that remain. Over time, we hope that showing fewer unnecessary warnings will mitigate the false alarm effect and increase confidence in HTTPS error warnings.

We encourage other browser vendors to experiment with and adopt similar policies for storing (and forgetting) certificate error decisions. We would be interested to learn whether other browsers find similar benefits and side effects.

### 6.3 Warning adherence across browsers

Researchers need to account for storage policies when testing browser UI or otherwise comparing adherence rates. It is tempting to attribute differences in adherence to obvious differences in UI across browsers or experimental treatments. However, our findings demonstrate that one must first control for differences in storage policies.

Consider our efforts to improve Chrome's HTTPS error warning. In 2013, we learned that Firefox users were twice as likely to adhere to warnings as Chrome users (66% vs 30%) [1]. We initially attributed this to the obvious differences in UI

between the browsers and thus began experimenting with design changes. Was Firefox's text easier to understand? Did the background color matter?

We were partly right: the design did matter. However, it was not the only factor. Chrome's HTTPS warning adherence rate remained lower than Firefox's even after a full redesign [10]. In fact, Chrome's adherence rate remained lower even when we tried using Firefox's exact warning UI in Chrome [11]. At the time, we hypothesized that this surprising finding might be due to demographics or storage policies [2, 11]. Our findings now support the storage policy hypothesis; Firefox's longer storage policy should give it a higher adherence rate. As Table 1 shows, Chrome's adherence rate could be higher or lower than Firefox's depending on our choice of storage policy.

Our findings also have two implications for laboratory studies. First, the effect of storage policies makes it difficult to compare adherence rates in the field to rates in a laboratory setting. A controlled laboratory study will include a fixed number of repeat warning exposures over a short period of time, whereas field data might include an unknown number of repeat warning exposures over a long period of time. This is not an apples-to-apples comparison. Second, we also recommend that researchers control for the number of repeat exposures when comparing experimental treatment groups, either across experiments or within the same experiment.

### 6.4 Storage policies are confusing

Our survey respondents were not familiar with Chrome's storage policy. Few of the survey respondents correctly identified Chrome's current storage policy, and most Google Consumer Survey respondents couldn't guess at all.

We find this confusion unsurprising. The eight browsers that we examined (Section 3.2) have four different storage policies, and we added a fifth policy. Browsers made by the same company have different policies across platforms, so people who use multiple devices will see different behaviors over time. Furthermore, storage policies are not well documented. Firefox is the only browser to mention the storage policy in the warning UI, and the policies are not mentioned on most browsers' help pages. Given this, how would people learn about storage policies?

We do not conclude that browser vendors should immediately embark on an education campaign. End users are not responsible for learning all of the technical details of their browsers. Instead, we think that browsers should act in the user's best interest and try to meet user expectations as much as possible without explicitly teaching people about storage policies. However, future work could explore whether people's behavior changes once they learn about different storage policies. (For example, people might be more cautious if they learn that exceptions last forever.) If that were the case, then comprehension of storage policies might be important enough to merit UI changes.

## 7. CONCLUDING SUMMARY

How long should a browser store a user's decision to override an HTTPS error warning? There is no clear industry consensus — eight major browsers have four different policies — and little research exists to guide the choice. We defined the usability and security requirements of an ideal policy,

and then we proposed a policy that meets our constraints.

We performed a large-scale field experiment to test different storage policies. After comparing adherence and regret rates between experimental groups, we concluded that error exceptions should be forgotten after one week. A one-week storage policy raised the adherence rate from 56% to 70% with little cost to the regret rate. Google Chrome 45 adopted our proposal, which brought the overall adherence rate to 72% as of February 2016.

To learn more about user beliefs and preferences, we ran Mechanical Turk and GCS surveys that asked about Chrome's storage policy. Most respondents did not know Chrome's current storage policy, and preferences were split between Chrome's old policy and our proposal. We remain satisfied with our proposal because respondents did not appear to have strong enough opinions to negate the clear benefit that we observed in our field experiment.

We encourage future work into the usability and security of different error storage policies. Would other browsers benefit from our policy? Are there changes to our policy that would improve it? How important is comprehension?

## 8. ACKNOWLEDGMENTS

We thank Ryan Sleevi and Chris Palmer for their expert input into the security trade-offs and experimental design. We also thank SOUPS reviewers for their suggestions.

## 9. REFERENCES

- [1] D. Akhawe, B. Amann, M. Vallentin, and R. Sommer. Here's my cert, so trust me, maybe? Understanding TLS errors on the Web. In *World Wide Web Conference (WWW)*, 2013.
- [2] D. Akhawe and A. P. Felt. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *Proceedings of Usenix Security*, 2013.
- [3] B. B. Anderson, C. B. Kirwan, J. L. Jenkins, D. Eargle, S. Howard, and A. Vance. How polymorphic warnings reduce habituation in the brain: Insights from an fMRI study. In *Proceedings of CHI*, 2015.
- [4] C. Bravo-Lillo, L. F. Cranor, S. Komanduri, S. Schechter, and M. Sleeper. Harder to ignore? Revisiting pop-up fatigue and approaches to prevent it. In *Proceedings of SOUPS*, 2014.
- [5] C. Bravo-Lillo, S. Komanduri, L. F. Cranor, R. W. Reeder, M. Sleeper, J. Downs, and S. Schechter. Your attention please: Designing security-decision UIs to make genuine risks harder to ignore. In *Proceedings of SOUPS*, 2013.
- [6] S. Breznitz and C. Wolf. *The psychology of false alarms*. Lawrence Erlbaum Associates, NJ, 1984.
- [7] P. Eckersley. A Syrian man-in-the-middle attack against Facebook. <https://www.eff.org/deeplinks/2011/05/syrian-man-middle-against-facebook>. Accessed June 2016.
- [8] S. Egelman, L. F. Cranor, and J. Hong. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of CHI*, 2008.
- [9] S. Egelman and S. Schechter. The importance of being earnest [in security warnings]. In *Financial Cryptography and Data Security, Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013.
- [10] A. P. Felt, A. Ainslie, R. W. Reeder, S. Consolvo, S. Thyagaraja, A. Bettles, H. Harris, and J. Grimes. Improving SSL warnings: Comprehension and adherence. In *Proceedings of CHI*, 2015.
- [11] A. P. Felt, R. W. Reeder, H. Almuhiemedi, and S. Consolvo. Experimenting at scale with Google Chrome's SSL warning. In *Proceedings of CHI*, 2014.
- [12] E. Hjelmvik. Analysis of Chinese MITM on Google. <http://www.netresec.com/?page=Blog&month=2014-09&post=Analysis-of-Chinese-MITM-on-Google>. Accessed June 2016.
- [13] E. Hjelmvik. Forensics of Chinese MITM on GitHub. <http://www.netresec.com/?page=Blog&month=2013-02&post=Forensics-of-Chinese-MITM-on-GitHub>. Accessed June 2016.
- [14] S. Kim and M. S. Wogalter. Habituation, dishabituation, and recovery effects in visual warnings. In *Proceedings of Human Factors and Ergonomics Society Annual Meeting*, 2009.
- [15] A. Kingsley-Hughes. Gogo in-flight wi-fi serving spoofed ssl certificates. <http://www.zdnet.com/article/gogo-in-flight-wi-fi-serving-spoofed-ssl-certificates/>, January 2015.
- [16] K. Krol, M. Moroz, and M. A. Sasse. Don't work. Can't work? Why it's time to rethink security warnings. In *Proceedings of the International Crisis on Risk and Security of Internet and systems (CRiSIS)*, 2012.
- [17] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The emperor's new security indicators: An evaluation of website authentication and the effect of role playing on usability studies. In *Proceedings of IEEE Symposium on Security and Privacy*, 2007.
- [18] H. K. Sotirakopoulos, A. and K. Beznosov. On the challenges in usable security lab studies: Lessons learned from replicating a study on SSL warnings. In *Proceedings of SOUPS*, 2011.
- [19] J. Sunshine, S. Egelman, H. Almuhiemedi, N. Atri, , and L. F. Cranor. Crying wolf: An empirical study of SSL warning effectiveness. In *Proceedings of USENIX Security*, 2009.
- [20] P. Thorley, E. Hellier, and J. Edworthy. Habituation effects in visual warnings. *Contemporary Ergonomics*, 2001.

## APPENDIX

### A. SURVEY SCREENSHOTS

#### A.1 Mechanical Turk

**Error pages**

This survey asks questions about an error page that you might see while browsing the Internet.

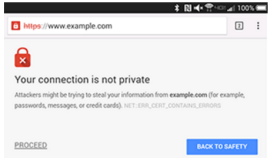
**\* Required**

**Your Mechanical Turk worker ID \***

**Which Internet browsers do you use at least once a week? \***

- Google Chrome
- Opera / Opera Mini
- Internet Explorer
- Safari
- UC Browser
- Firefox

**Imagine you saw this error page while trying to open a website in Chrome:**



**If you clicked 'Proceed' on the error page, how long would Chrome remember your decision for? \***

**Have you ever seen this error page before, in Chrome? \***

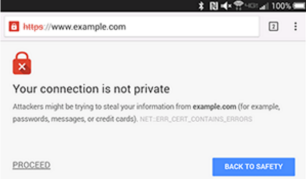
- Yes
- No
- I don't remember

**Submit**

*Never submit passwords through Google Forms.*

Please complete the following survey to access this premium content.

If you clicked 'Proceed' on this error page, how long would you WANT Chrome to remember that decision for?



Select an answer

OR

Show me a different question

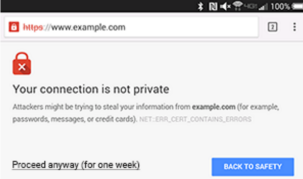
Skip survey

Google [INFO](#) [PRIVACY](#)

#### A.2 Google Consumer Survey

Please complete the following survey to access this premium content.

If you clicked 'Proceed' on this error page, how long would Chrome remember that decision for?



Select an answer

OR

Show me a different question

Skip survey

Google [INFO](#) [PRIVACY](#)