Multisurf: MITM detection using multiple HTTP clients

Marcela Melara, Annie Edmundson, Ohad Fried

1 Introduction

An increasing number of popular websites support the SSL/TLS protocol, the current standard for encrypting web traffic. Most commonly seen as part of the HTTPS protocol, SSL/TLS provides data and message confidentiality to protect users browsing the web from malicious attackers attempting to eavesdrop or tamper with traffic. Nonetheless, about 48% of popular websites remain insecure by only supporting HTTP connections [8], which are vulnerable to man-in-the-middle (MITM) attacks. Because HTTP traffic is not encrypted nor authenticated, an unsuspecting user may be visiting a specific website without realizing that an adversary has modified the contents of these web pages while in transit.

Indeed, such in-flight web page modifications occur in practice with a surprising frequency for various reasons, often resulting in undesirable effects such as injected advertisements, broken pages, and exploitable security vulnerabilities [7]. While the vast majority of changes to web pages in transit have economic incentives for website publishers, [7] also found that a portion of their measured in-flight modifications was due to injected malware. To provide web servers with a practical, more affordable alternative to HTTPS, Reis *et al.* [7] proposed *web tripwires*, client-side scripts that can detect most modifications to unencrypted web pages.

However, this solution requires that website publishers modify their pages to contain these scripts, and users unaware of the installed web tripwires may disregard warning messages as spam. Furthermore, helping website publishers understand and react to any changes made *en route* does not necessarily help users protect themselves from injected malware. This paper presents *Multisurf*, a browser extension which checks the integrity of unencrypted web pages helping users detect when their HTTP traffic has been hijacked, and without requiring support from the administrator or owner of the affected website. Multisurf's collaborative integrity checks detect in-flight changes to websites through a system of trusted peers, end-hosts run by persons or institutions that a user running the Multisurf client trusts. By gathering the peers' versions of requested web content, the client can verify whether the visited web page was tampered with in transit. Because many changes to web pages *en route* are not of malicious nature, the Multisurf client displays the result of the integrity check and gives the user the option of viewing the peers' versions to help her determine if she accepts the in-flight modifications or if she considers the modifications to be malicious, blacklisting the site. Thus, Multisurf leverages out-of-band communication, inter-personal trust and takes user preferences into account giving control to the end user in whether she would like to surf the web in a more aware way.

This paper is organized as follows. In Section 2 we outline our system model. Section 3 details Multisurf's system design and the collaborative integrity check protocol; we evaluate the efficacy and accuracy of Multisurf in Section 4. Section 5 describes some related work, and we discuss directions for future work in Section 6. We conclude in Section 7.

2 System Model

In Multisurf, a user runs the Multisurf browser extension which communicates with a native client application in the background. The extension gathers all outgoing HTTP requests and incoming HTTP responses, including the retrieved page content, and passes this information on to the client. Upon a request for a web page, the Multisurf client engages a user-defined set of trusted peers, trusted end-hosts which run a specialized Multisurf peer application that mirrors the browser's web requests on demand and return the served web content to the client. To ensure that the client-to-peer connections are not compromised, these parties establish a secure communications channel using HTTPS when participating in the Multisurf protocol. Once all peers have finished the protocol, the client performs the integrity check by comparing the page content from each peer with each other as well as with its own retrieved page content on various levels of granularity. We elaborate on the specific comparison vectors used in Section 3.

The client then returns the result of the integrity check back to the browser extension so that it can reflect the result to the user. Multisurf considers a site to be safe if either no discrepancies are found, or if the client and peers all differ in the exact same page element in the content. The latter case could indicate a peer-specific server-side modification to the page based on factors such as geographic location, or an in-flight change by an advertising company inserting a different ad for each peer in the same element of the web page, neither of which are malicious

in nature. Should the client detect discrepancies between some of the peers and itself that are not consistent across all checked content, it will mark the web page as unsafe. Lastly, if the browser extension or the client detect that the requested web page supports HTTPS, querying the peer becomes unnecessary as we assume the site cannot be compromised by a man-in-the-middle. Thus, the extension displays a third kind of message to the user when a site is HTTPS-only.

2.1 Goals

We aim to achieve the following properties in Multisurf:

Lightweight. Multisurf should not pose an unreasonable computational or storage burden on the browser, the client host, or the trusted peers. Since the client and peers communicate via HTTPS, Multisurf must only engage the peers when necessary, *i.e.,* when the user requests an unencrypted web page.

Rapid Detection of Modifications. Because there are no guarantees about the integrity of the traffic sent over an HTTP connection, the client must be able to easily detect any discrepancies between its version of a web page and the peer's version to quickly notify the user of the status of the page. To ensure rapid detection of changes to pages *en route*, Multisurf's comparison vectors must depend only on the served page contents.

2.2 Threat Model

We assume an active man-in-the-middle (MITM) attacker who can only control a Multisurf user's Internet access link, and is distant enough from any web server's link to not control all server-to-client traffic; for instance, the adversary resides on the same insecure WiFi network as the user. However, Multisurf cannot prevent passive nor active MITM attacks against users from occurring, but instead provides integrity checks to help users detect when they have become the victim of an *active* MITM attack.

We rely on the fact that it is hard for the adversary to perform a MITM attack at multiple locations at the same time.[1] Therefore, the client's and peers' access links cannot be compromised by a single adversary simultaneously.

Detecting an eavesdropper on the user's access link as well as discovering and protecting against a compromised web server link remain out of the scope of this work.

3 Design

The Multisurf system has three main components:

- The browser extension which interacts with the user.
- The native client application which is responsible for communicating with the peers and for performing the integrity checks.
- The peer application which mirrors any requests from the Multisurf client.

The browser extension gathers HTTP request and response data while the user browses the web normally. In the background, the Multisurf extension passes this information to the native client, which solicits all received HTTP requests via SSL to one or more trusted peers. These also issue the same request for a web page to the web server. The peer returns its version of the serverâĂŹs served content to the client. Figure 1 details the client-peer protocol.

To perform the integrity check, the Multisurf client compares the original server web page content with the peers' versions of the request web page. We have designed various comparison vectors which the client can employ to verify the integrity of the received content for the request web page:

- A line-by-line comparison of the content. This is a rather coarse-grained comparison metric.
- The number of script tags in the content. Any discrepancies in this could indicate a malicious script injection.
- The links contained within `<script>` tags and `<a>` tags. This algorithm finds matching elements in the web content being compared. Differing links in the same element may indicate a malicious in-flight injection.

The client sends the result of the consistency check back to the browser extension, which displays the results to the user. In the case of an unsafe website, the browser extension allows the user to view the differing web pages to help her determine if she deems the differences malicious or not. Should the user decide that an unsafe website is malicous, the extension allows her to blacklist the site for future browsing sessions.

3.1 Implementation

We have implemented a working Multisurf prototype. The browser extension is implemented for the Google Chrome browser in Javascript and Python. The native client and peer applications are implemented in Python. We have made our code available on Git Hub.[2]

4 Evaluation

In order to evaluate Multisurf, we ask the following questions:

[1] More specifically, by multiple locations we mean networks at distinct geographic locations.

[2] `http://github.com/ohadf/multisurf`

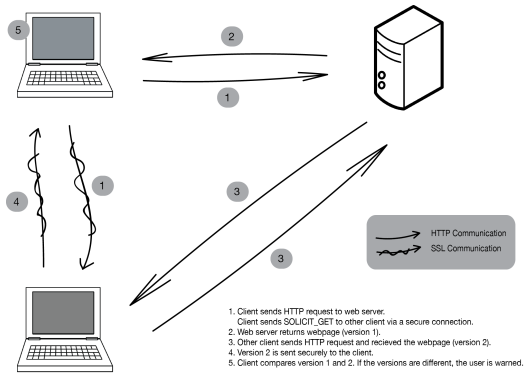

Figure 1: The Multisurf client-peer protocol.

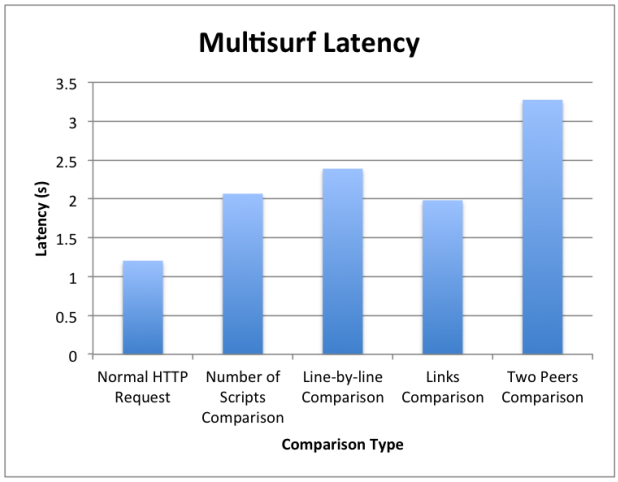

Figure 2: Latency of Multisurf.

- How much does it cost to use Multisurf in terms of performance?
- How accurately does Multisurf classify "safe" and "unsafe" sites?

We answer these questions by measuring the latency of our system, as well as analyzing the results of running our system using different comparison algorithms.

Our measurements are taken using the Alexa Top 100 [1] websites as our dataset. All of our experiments were performed using a laptop with a 2 GHz Intel Core 2 Duo processor.

4.1 Latency

For each web page, we measure the latency from the client; this is measuring the time for the client to make her own request, the time for the client to ask her peer to make the same request, and the time for the peer to make the request and return her response to the client. Using the Alexa Top 50 websites, we measure the latency 50 times and take the average. This is repeated for the four different comparison algorithms that Multisurf can use, as well as a base line measurement of the latency solely for the client to make a request. The results are shown in Figure 4.1.

The latency of the different methods in comparison to one another are what was expected. When the client sends just her own request, without interacting with her peer, she has the lowest latency. On the other hand, the highest latency occurs when the client uses two peers to determine if a site is safe; she must send her own request in addition to asking two peers to send their requests. Overall, the latencies were relatively similar and the additional time it took to use Multisurf does not outweigh the benefits.

4.2 Accuracy

We measure accuracy in terms of the number of websites that are classified as "safe" and "unsafe." Once the client recieves her own response as well as the response her peer sends, both responses go through a comparison, and are then placed in one of six categories:

1. "Safe"
2. "Unsafe"
3. Identical responses, but no content to compare
4. Different responses, but no content to compare
5. HTTPS-Only
6. Error

In determining accuracy, we only consider the first two categories. A "safe" site is one that was determined by the comparison algorithm to not have a man in the middle attack; an "unsafe" site is one that was determined by the comparison algorithm to possibly have a man in the middle attack. For our measurements, we assume that there were no man in the middle attacks and 100% accuracy would be no "unsafe" sites. We calculated the accuracy for our four different comparison methods: comparing the number of scripts, comparing line by line, comparing the links, and comparing line by line with two peers.

4.2.1 Number of Scripts

Figure 4.2.1 shows the results of using Multisurf with the comparison algorithm that compares the number of scripts in each response. The accuracy is 34.39%. This shows that there are a significant number of websites that send a different number of scripts in each response.

4.2.2 Line by Line

When comparing the responses line by line, Multisurf exhibited only 8.72% accuracy. The amount of dynamic web

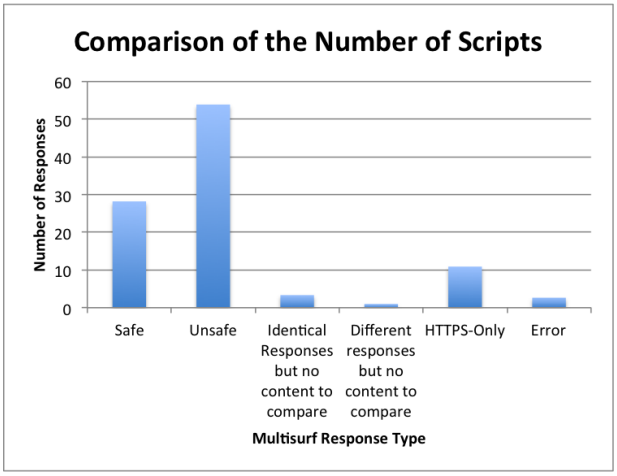

Figure 3: Responses from Comparing the Number of Scripts.

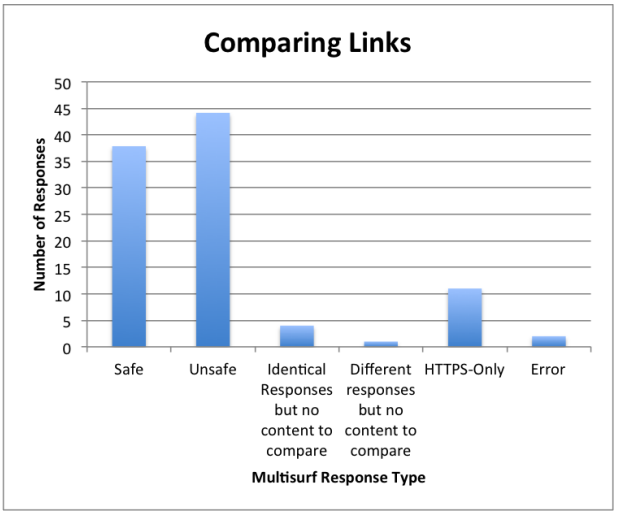

Figure 5: Responses from Comparing Links.

content on the Internet is a large factor in the low accuracy rate of this method. The results are shown in Figure 4.2.2.

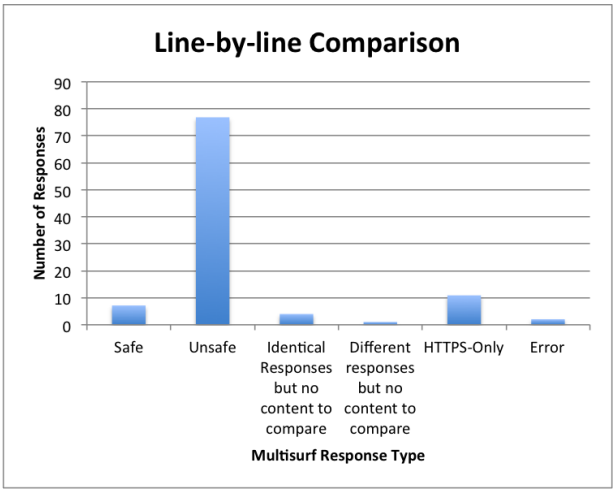

Figure 4: Responses from Comparing Line by Line.

4.2.3 Links

A possible man in the middle attack is to inject or change a URL in the response to take the client to a malicious site. The Multisurf response types for comparing which links are in each response are shown in Figure 4.2.3; this method has an accuracy of 46.16%.

4.2.4 Line by Line with Two Peers

The accuracy rates for the three comparison methods above are relatively low; this can be improved by adding a second peer. In this case, we check to see if the response of the client is different from either of the peers' responses. If the peers' responses are the same (as measured line by line) and the client's response is different (as measured line by line), then the site is classified as "unsafe." Otherwise, the site is deemed "safe." The accuracy for this method is 100% and can be seen in Figure 4.2.4.

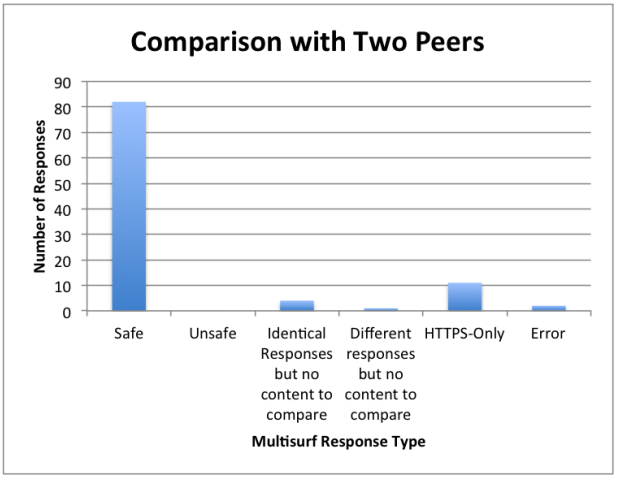

Figure 6: Responses from Comparing with Two Peers.

5 Related Work

While there has been much research into man in the middle attacks, their attack surfaces, as well as their detection and prevention, our work takes a different approach and is specific to HTTP. Parts of Multisurf were inspired by

Web Tripwires; other work that is related includes Tor, Perspectives, and a variety of research on SSL.

Web Tripwires. Using client-side javascript code, web tripwires are a way to detect many in-flight page modifications over HTTP [7]. Our system differs in a couple of ways: (1) We attempt to solely detect man-in-the-middle attacks, whereas web tripwires detects modifications from pop-up blockers, ad blockers, ad injectors, as well as from other sources. (2) Our system will not have access to any servers that hold web data. Instead of comparing the requested page to a known-good representation of the requested page, we compare our requested page to a trusted peer/organization's request of the same page.

Tor. The goal of Tor is to hide the origin of a web request [3]. However, the use of Tor is orthogonal to this work for a few reasons: (1) Tor is not suitable for everyday web browsing because it is slow and many commonly used services are not compatible with Tor, and (2) Users may not be seeking the kind of secure web browsing environment that Tor offers, i.e. they may still want to have a locale-dependent/personalized experience. Thus, our work provides the compatibility with common web services, is light-weight, and does not significantly change a user's every-day browsing experience.

Perspectives. Rather than validating an SSL certificate by checking for certificate authority approval, with Perspectives the browser validates a certificate by checking for consistency with the certificates observed by the network notaries over time [9]. Our system uses a similar approach to check whether web content is being served consistently across the web. Instead of having a group of trusted notary servers, the user designates a group of trusted peers/organizations that mirror the HTTP request on her behalf and relay the response they receive from the web server back to the user. The user's browser then compares the received responses to verify that the various versions of the requested web content are consistent.

SSL. There has been a variety of research in detection and prevention of man in the middle attacks over SSL [2, 4, 5, 6, 10]. There has been related work, such as this, on the topic of MITM attacks over SSL, but few that cover these attacks simply over HTTP. Xia and Brustoloni study methods of prevention in a user-friendly manner [10]. While our system also aims for the goal of being user-friendly, Multisurf is only applicable to HTTP traffic.

6 Future Work

Comparison Algorithms. We currently have four different comparison algorithms, and the majority of them are not relatively accurate. Some future work includes more fine-grained comparison algorithms that can better handle dynamic content and provide better accuracy. Another possibility is the combination of different comparison algorithms.

Learning Algorithm. It would be useful for Multisurf to learn which responses look "safe" and which look "unsafe" while a client is using it. This would also improve accuracy and potentially performance.

User-friendly Display. A graphical display in the browser would make Multisurf more usable. The challenge for this work is in deciding what is most understandable to a user. While showing the different lines of html are not understandable (and therefore not useful), showing what parts of a web page are different between the client's response and the peer's response would be helpful.

Availability. Our future work involves new ways to be able to run our system without depending on a peer or peers to always be available.

Optimizations. There is plenty of room in Multisurf for performance optimizations. For example, in the comparison algorithm with two peers, each peer request is sent consecutively, but the performance can be optimized if the peer requests were sent simultaneously.

Other Browsers. Adding extensions for other browsers would make Multisurf available to more users.

7 Conclusions

HTTP provides little security for users, and is susceptible to man in the middle attacks. We presented Multisurf, a system for detecting and notifying a user of a potentially malicious website. Multisurf is a Chrome browser extension that allows for easy access and usability.

Using the Alexa Top 100 sites, we have shown that Multisurf can use a variety of comparison algorithms to detect differences in web page responses. This shows that our system can detect an active man in the middle attack, where the attacker is carrying out a targeted attack on the client. Multisurf does this in a user-friendly way and without presenting obstacle to the user.

References

[1] Alexa: The Web Information Company. Top Sites. URL `http://www.alexa.com/topsites`.

[2] F. Callegati, W. Cerroni, and M. Ramilli. Man-in-the-middle attack to the https protocol. *Security & Privacy, IEEE*, 7(1):78–81, 2009.

[3] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[4] J. JIA and Z. XUE. The principle and prevention of ssl man-in-the-middle attack [j]. *China Information Security*, 4, 2007.

[5] Y. Joshi, D. Das, and S. Saha. Mitigating man in the middle attack over secure sockets layer. In *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*, pages 1–5. IEEE, 2009.

[6] R. Oppliger, R. Hauser, and D. Basin. Ssl/tls session-aware user authentication–or how to effectively thwart the man-in-the-middle. *Computer Communications*, 29(12):2238–2246, 2006.

[7] C. Reis, S. D. Gribble, T. Kohno, and N. C. Weaver. Detecting in-flight page changes with web tripwires. In *NSDI*, volume 8, pages 31–44, 2008.

[8] TIM Trustworthy Internet Movement. SSL Pulse. `https://www.trustworthyinternet.org/ssl-pulse/`, Retrieved Jan. 2014.

[9] D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving ssh-style host authentication with multi-path probing. In *USENIX Annual Technical Conference*, pages 321–334, 2008.

[10] H. Xia and J. C. Brustoloni. Hardening web browsers against man-in-the-middle and eavesdropping attacks. In *Proceedings of the 14th international conference on World Wide Web*, pages 489–498. ACM, 2005.