Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 24, 2014

V. Narayanan
Google

October 21, 2013

Explicit Proxying in HTTP - Problem Statement And Goals
draft-vidya-httpbis-explicit-proxy-ps-00

Abstract

This document describes the issues with HTTP proxies for TLS protected traffic and motivates the need for explicit proxying capability in HTTP. It also presents the goals that such a solution would need to satisfy and some example solution directions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Web proxies that are present in the communication path between clients and servers are fairly common practice. There are many reasons one may employ a proxy, but the commonly deployed scenarios today are at odds with client to server privacy. While in almost all cases, some kind of user consent is received to carry traffic through proxies, such consent is fairly vague and the user is often unaware about the extent to which proxies have visibility into their communications. In some cases, such visibility may be construed as an unacceptable violation of privacy.

This document describes the types of proxies and the issues with the currently used models. It makes a case for legitimizing the use of proxies while providing sufficient transparency to the endpoints. Towards that end, it also sets goals towards designing such an explicit proxying mechanism for http communication. Note that the scope of this document is limited to HTTPS only - HTTP communication not protected by TLS is out of scope for this discussion.

2. Motivation

The move to securing http connections is often to provide a confidential channel for exchange of information between a client and server. In the presence of a proxy, a secure channel may perceived to be end-to-end when it is in fact not the case. To fix this, proxies must be visible to the communicating endpoints. However, without an interoperable solution, explicit proxying of connections becomes an issue. The motivations to make proxying explicit include:

- Making secure communications possible for users

- Allowing endpoints to choose not to communicate in the presence of an intermediary

- Ensuring that a proxy is authorized to be in the path

- Allowing detection of modified content

- Allowing the ability to cache content in intermediate entities

3. Proxying Needs Today

There are number of reasons proxies have been deployed in networks. Some of these reasons include:

- Policy Enforcement

  Authentication and bandwidth policies may be often enforced using a proxy. This allows the model of having conditional connectivity or limits on connectivity such as may be observed in a hotel or an airport hotspot, among other places. For TLS protected connections, this type of policy enforcement becomes difficult (the connections just fail until the user finds a way to authenticate with the proxy). But, it may be useful to support this using explicit proxying techniques for a better experience.

- Caching

  Very widely used on the Internet, caching proxies allow serving of content from topologically closer sources in order to preserve network resources and improve user experience.

- Content Modification

  Certain networks employ content modifying proxies that generally modify content to improve network and overall user experience. For instance, proxies in mobile networks may need to modify content to change the codec for sake of older devices. However, sometimes, content modification proxies have also been known to make high definition content unavailable - while arguably this may be in the best interest in balancing the overall health of the network (a bad network isn’t good for any user), this is also highly debatable.

- Content Filtering

  Content filtering proxies may prevent malware, but may also prevent access to sites that are in violation of the administrative policies. This filtering may be by size, type or subject matter.

- Content Inspection

  Some proxies may be installed with a need to inspect and flag inappropriate content (e.g., in schools).

As can be seen, some of the proxies need access to the content, while others do not.

4. Proxy Configurations Today

Proxies used to be configured manually by having the users specify the proxy information in the browser settings. However, due to the difficulty in effectively implementing this approach (specifically that the user needed to be involved when he/she moved out of the proxy’s network), two proxying models have evolved. One is the group configuration policies that are used in enterprises, where a device that is part of an enterprise gets subjected to the enterprise proxying policies by pre configuration.

Another is the model of "interception" or "transparent" proxies became widely popular and is also in a fairly large use today. In the "transparent" proxy model, neither endpoint knows about the interception. Transparent proxies can be employed in the presence of TLS (HTTPS) as well, when certificate pinning is disabled. Most deployments end up disabling certificate pinning so that proxying can be accomplished. The client machines are often configured with root certs that will allow it to accept the proxy generated ephemeral certificate for the server. Future configurations of proxies continues to be a problem and explicit mechanisms of configuring proxies may be necessary to motivate the move away from interception proxies.

There are also tunneling proxies, where HTTP CONNECT may be used to tunnel the client requests to the server.

5. Problems With Proxies Today

The use of proxies leads to a number of privacy issues. To summarize:

- The user often has no knowledge that their data exchanges are passing through an interception proxy that potentially has visibility to the actual content exchanged.

- The server has no knowledge of the presence of the proxy and hence, cannot refuse to serve sensitive content over a proxied connection.

- The weakened security model, when certificate pinning is disabled at a general level, allows inspection of content potentially by entities other than legitimate proxies that the user may be willing to give access to. This is especially true in enterprises with a multitude of platforms, devices and browsers, where explicit configuration of proxy certificates is an administrative burden.

- The client can no longer authenticate the real server. Hence, the client has no influence on certificate revocation checks and any visible warnings to the user are made infeasible.

- Client authentication (and hence mutual authentication) is made infeasible. Any server relying upon mutual authentication to establish a TLS connection cannot work with transparent proxies.

- The user has no way of detecting whether content has been modified en route.

- The client is susceptible to downgrade attacks, as it cannot do any policy enforcement on versions or algorithms.

With privacy becoming more and more important, it is important for us to support solutions that allow awareness of a privacy breach to both users and the servers, when that happens. To this effect, it is important that proxies be explicitly supported and detected.

6. Explicit HTTP Proxying

The problems illustrated in this document call for explicit knowledge of on-path proxies to the user and the content provider. The key assumptions and goals driving this target are stated below.

6.1. Assumptions

6.1.1. On Users, Intermediaries And Content Providers

- Users configure proxies and forget about the configuration.
- Users have limited control over provider installed certificates.
  - Often, the user’s only choice is to not sign up for service at all.
- Users may not wish to have some or any of their communications intercepted, even when they are on a network for which they previously configured a proxy.
- Intermediaries have various legitimate reasons for wanting to inspect traffic:
  - Cache content
  - Implement network traffic policies (e.g. legal compliance, malware detection, etc)
- Content providers may not wish to serve certain content in anything less than an end-to-end secure fashion.

6.1.2. On Today’s Practices

- Many networks seem to leave the traffic on port 443 untouched and unblocked today, likely as a result of both the importance of the data and the relative rarity of communications using TLS. It is unclear how this might change as TLS protected traffic increases, if we continued to not have a solution for explicit proxying.
- Entities which need to inspect traffic on port 443 today are forced to either block port 443 or to deploy an intercepting proxy and install root certs on all devices which may use the network. In the latter case, the deployed proxy impersonates both the content-provider to the user-agent, and the user-agent to the content-provider. Though there is work to allow users to detect these situations [DANE], support is not widespread.
- Many, if not most, mobile devices using cellular networks use proxies and several of them act as transforming proxies.

- Users and sites have only one mechanism for specifying point-to-point security policy for HTTP [RFC2616], which is the scheme of the URI identifying any particular resource.

6.2. Goals

These are the goals of a solution aimed at making proxying explicit in HTTP.

- In the presence of a proxy, users’ communications SHOULD at least use a channel that is point-to-point encrypted.

- Users MUST be able to opt-out of communicating sensitive information over a channel which is not end-to-end private.

- Content-providers MAY serve certain content only in an end-to-end confidential fashion.

- Interception proxies MUST be precluded from intercepting secure communications between the user and the content-provider.

- User-agents and servers MUST know when a transforming proxy is interposed in the communications channel.

- User-agents MUST be able to detect when content requested with an https scheme has been modified by any intermediate entity.

- Entities other than the server or user-agent SHOULD still be able to provide caching services.

- User agents MUST be able to verify that the content is in fact served by the content provider.

- A set of signaling semantics should exist which allows the content-provider and the user to have the appropriate level of security or privacy signaled per resource.

- Ideally, HTTP URI semantics SHOULD NOT change, but if it does, it must remain backwards-compatible.

- Configuration and deployment of proxies should be as easy as currently used solutions.

- Introduction of explicit proxying MUST NOT require a flag day upgrade - in other words, it should work with existing client and content provider implementations during the transition.

7. Potential Solution Directions

There are several potential directions this work can take, some of which are clearly undesirable and some that are more viable. This section is not intended for an exhaustive description of each solution - rather, it is aimed at serving as a starting point for discussions. Note that more than one of these directions may need to be adopted and brought together for a complete solution - so, each section here is not intended to be standalone and complete.

7.1. Do Nothing

This is a scenario that continues to support interception proxies in current modes. The fundamental premise of this document is based on the fact that this is bad.

7.2. Signed Policy Per Origin

RFC6454 specifies a method by which there can be a signed policy per origin. However, such coarse granularity of providing a policy for an entire domain is often not useful and hence, this is rarely used in practice.

7.3. Explicit Proxy Detection using HTTP/TLS

Means of:

- Explicit signaling of presence of proxy from user agent to server.

- Signaling to indicate user preference for end-to-end secure communication

- Signaling to indicate content unavailability via proxies

- Verification of proxy identity to detect untrusted proxies

- Serving interstitial pages to manage portals that enforce bandwidth, connectivity times, etc.

The above can be accomplished in a variety of ways, including HTTP/TLS error codes, HTTP2.0 proxy signaling semantics and HTTP/TLS exchange of proxy identities.

7.4. Object Level Security in HTTP

The ability to detect modified content is needed. Specifically:

- Object level integrity protection of content by content provider

- Object level encryption by content provider (optionally)

7.5. TLS Origin Cert Exchanges

The ability to exchange the true certificate chain of the server in TLS exchanges so that clients can make better decisions about servers.

8. Security Considerations

TBD

9. IANA Considerations

TBD

10. Acknowledgments

List of names here - TBD

11. Normative References

[1]   Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Author’s Address

Vidya Narayanan
Google
1600 Amphitheatre Pkwy
Mountain View, CA
USA

Email: vn@google.com