Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 05, 2015

M. Nottingham

July 04, 2014

Problems with Proxies in HTTP
draft-nottingham-http-proxy-problem-01

Abstract

This document discusses the use and configuration of proxies in HTTP, pointing out problems in the currently deployed Web infrastructure along the way. It then offers a few principles to base further discussion upon, and lists some potential avenues for further exploration.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

HTTP/1.1 [RFC7230] was designed to accommodate proxies. It allows them (and other components) to cache content expansively, and allows for proxies to break "semantic transparency" by changing message content, within broad constraints.

As the Web has matured, more networks have taken advantage of this by deploying proxies for a variety of reasons, in a number of different ways. Section 2 is a survey of the different ways that proxies are used, and Section 3 shows how they are interposed into communication.

Some uses of proxies cause problems (or the perception of them) for origin servers and end users. While some uses are obviously undesirable from the perspective of an end users and/or origin server, other effects of their deployment are more subtle; these are examined in Section 4.

These tensions between the interests of the stakeholders in every HTTP connection - the end users, the origin servers and the networks they use - has led to decreased trust for proxies, then increasing deployment of encryption, then workarounds for encryption, and so forth.

Left unchecked, this escalation can erode the value of the Web itself. Therefore, Section 5 proposes straw-man principals to base further discussion upon.

Finally, Section 6 proposes some areas of technical investigation that might yield solutions (or at least mitigations) for some of these problems.

Note that this document is explicitly about "proxies" in the sense that HTTP defines them. Intermediaries that are interposed by the server (e.g., gateways and so-called "Reverse Proxies", as used in Content Delivery Networks) are out of scope.

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Why Proxy?

HTTP proxies are interposed between user agents and origin servers for a variety of purposes; some of them are with the full knowledge and consent of end users, to their benefit, and some are solely for the purposes of the network operator - sometimes even against the interests of the end users.

This section attempts to identify the different motivations networks have for deploying proxies.

2.1. Application Layer Gatewaying

Some networks do not have direct Internet connectivity for Web browsing. These networks can deploy proxies that do have Internet connectivity and then configure clients to use them.

Such gatewaying between networks were some of the first uses for proxies.

2.2. Caching

An extremely common use of proxies is to interpose a HTTP cache, in order to save bandwidth, improve end-user perceived latency, increase reliability, or some combination of these purposes.

HTTP defines a detailed model for caching (see [RFC7234]); however, some lesser-known aspects of the caching model can cause operational issues. For example, it allows caches to go into an "offline" mode where most content can be served stale.

Also, proxy caches sometimes fail to honor the HTTP caching model, reusing content when it should not have been. This can cause interoperability issues, with the end user seeing overly "stale" content, or applications not operating correctly.

2.3. Network Policy Enforcement

Some proxies are deployed to aid in network policy enforcement; for example, to control access to the network, requiring a login (as allowed explicitly by HTTP's proxy authentication mechanism), bandwidth shaping of HTTP access, quotas, etc. This includes so-called "Captive Portals" used for network login.

Some uses of proxies for policy enforcement cause problems; e.g., when a proxy uses URL rewriting to send a user a message (e.g., a "blocked" page), they can make it appear as if the origin server is sending that message - especially when the user agent isn't a browser (e.g., a software update process).

2.4. Content Filtering (a.k.a. Content Policy Enforcement)

Some networks attempt to filter HTTP messages (both request and response) based upon network-specific criteria. For example, they might wish to stop users from downloading content that contains malware, or that violates site policies on appropriate content, or that violates local law.

Intermediary proxies as a mechanism for enforcing content restrictions are often easy to circumvent. For example, a device might become infected by using a different network, or a VPN. Nevertheless, they are commonly used for this purpose.

Some content policy enforcement is also done locally to the user agent; for example, several Operating Systems have machine-local proxies built in that scan content.

Content filtering is often seen as controversial, often depending on the context it is used within and how it is performed.

2.5. Content Modification

Some networks modify HTTP messages (both request and response) as they pass through proxies. This might include the message body, headers, request-target, method or status code.

Motivation for content modification varies. For example, some mobile networks interpose proxies that modify content in an attempt to save bandwidth, improve perceived performance, or transcode content to formats that limited-resource devices can more easily consume.

Modifications also include adding metadata in headers for accounting purposes, or removing metadata such as Accept-Encoding to make virus scanning easier.

In other cases, content modification is performed to make more substantial modifications. This could include inserting advertisements, or changing the layout of content in an attempt to make it easier to use.

Content modification is very controversial, often depending on the context it is used within and how it is performed. Many feel that, without the explicit consent of either the end user or the origin server, a proxy that modifies content violates their relationship, thereby degrading trust in the Web overall.

However, it should be noted that Section 5.7.2 of [RFC7230] explicitly allows "non-transparent" proxies that modify content in certain ways. Such proxies are required to honor the "no-transform" directive, giving both user agents and origin servers a mechanism to

"opt out" of modifications ([RFC7234], Section 5.2.1.6); however, it is not technically enforced.

[W3C.NOTE-ct-guidelines-20101026] is a product of the W3C Mobile Web Best Practices Working Group that attempts to set guidelines for content modification proxies. Again, it is a policy document, without technical enforcement measures.

3. How Proxies are Interposed

How a proxy is interposed into a network flow often has great affect on perceptions of its operation by end users and origin servers. This section catalogues the ways that this happens, and potential problems with each.

3.1. Manual Configuration

The original way to interpose a proxy was to manually configure it into the user agent. For example, most browsers still have the ability to have a proxy hostname and port configured for HTTP; many Operating Systems have system-wide proxy settings.

Unfortunately, manual configuration suffers from several problems:

- o Users often lack the expertise to manually configure proxies.
- o When the user changes networks, they must manually change proxy settings, a laborious task. This makes manual configuration impractical in a modern, mobile-driven world.
- o Not all HTTP stacks support manual proxy configuration. Therefore, a proxy administrator cannot rely upon this method.

3.2. proxy.pac and WPAD

The limitations of manual configuration were recognized long ago. The solution that evolved was a format called "proxy.pac" [proxypac] that allowed the proxy configuration to be automated, once the user agent had loaded it.

Proxy.pac is a JavaScript format; before each request is made, it is dispatched to a function in the file that returns a string that denotes whether a proxy is to be used, and if so, which one to use.

Discovery of the appropriate proxy.pac file for a given network can be made using a DHCP extension, [wpad]. WPAD started as a simple protocol; it conveys a URL that locates the proxy.pac file for the network.

Unfortunately, the proxy.pac/WPAD combination has several operational issues that limit its deployment:

- The proxy.pac format does not define timeouts or failover behavior precisely, leading to wide divergence between implementations. This makes supporting multiple user agents reliably difficult for the network.

- WPAD is not widely implemented by user agents; some only implement proxy.pac.

- In those user agents where it is implemented, WPAD is often not the default, meaning that users need to configure its use.

- Neither proxy.pac nor WPAD have been standardized, leading to implementation divergence and resulting interoperability problems.

- There are DNS-based variants of WPAD, adding to to confusion.

- DHCP options generally require tight integration with the operating system to pass the results to HTTP-based applications. While this level of integration is found between O/Ses and their provided applications, the interface may or may not be available to third parties.

- WPAD can be spoofed, allowing attackers to interpose a proxy and intercept traffic. This is a blocking issue for implementation.

3.3. Interception

The problems with manual configuration and proxy.pac/WPAD have led to the wide deployment of a third style of interposition; interception proxies.

Interception occurs when lower-layer protocols are configured to route HTTP traffic to a host other than the origin server for the URI in question. It requires no client configuration (hence its popularity over other methods). See [RFC3040] for an example of an interception-related protocol.

Interception is also strongly motivated when it is necessary to assure that the proxy is always used, e.g., to enforce policy.

Interception is problematic, however, because it is often done without the consent of either the end user or the origin server. This means that a response that appears to be coming from the origin server is actually coming from the intercepting proxy. This makes it difficult to support features like proxy authentication, as the

unexpected status code breaks many clients (e.g., non-interactive applications like software installers).

Furthermore, interception is a "layer violation"; i.e., misusing lower-layer protocols to enforce a higher-layer (often expressed as "layer 8") requirement.

In addition, as adoption of multi-path TCP (MPTCP) [RFC6824] increases, the ability of intercepting proxies to offer a consistent service degrades.

3.4. Configuration As Side Effect

More recently, it's become more common for a proxy to be interposed as a side effect of another choice by the user.

For example, the user might decide to add virus scanning - either as installed software, or a service that they configure from their provider - that is interposed as a proxy. Indeed, almost all desktop virus scanners and content filters operate in this fashion.

This approach has the merits of both being easy and obtaining explicit user consent. However, in some cases, the end user might not understand the consequences of use of the proxy, especially upon security and interoperability.

4. Second-Order Effects of Proxy Deployment

4.1. Proxies and HTTP

Deployment of proxies has an effect on the HTTP protocol itself. Because a proxy implements both a server and a client, any limitations or bugs in their implementation impact the protocol's use.

For example, HTTP has a defined mechanism for upgrading the protocol of a connection, to aid in the deployment of new versions of HTTP (such as HTTP/2) or completely different protocol (e.g., [RFC6455]).

However, operational experience has shown that a significant number of proxy implementations do not correctly implement it, leading to dangerous situations where two ends of a HTTP connection think different protocols are being spoken.

Another example is the Expect/100-continue mechanism in HTTP/1.1, which is often incorrectly implemented. Likewise, differences in support for trailers limits protocol extensions.

4.2. Proxies and TLS

It has become more common for Web sites to use TLS [RFC5246] in an attempt to avoid many of the problems above. Many have advocated use of TLS more broadly; for example, see the EFF's HTTPS Everywhere [https-everywhere] program, and SPDY's default use of TLS [I-D.mbelshe-httpbis-spdy].

However, doing so engenders a few problems.

Firstly, TLS as used on the Web is not a perfectly secure protocol, and using it to protect all traffic gives proxies a strong incentive to work around it, e.g., by deploying a certificate authority directly into browsers, or buying a sub-root certificate.

Secondly, it removes the opportunity for the proxy to inform the user agent of relevant information; for example, conditions of access, access denials, login interfaces, and so on. User Agents currently do not display any feedback from proxy, even in the CONNECT response (e.g., a 4xx or 5xx error), limiting their ability to have informed users of what's going on.

Finally, it removes the opportunity for services provided by a proxy that the end user might wish to opt into. For example, consider when a remote village shares a proxy server to cache content, thereby helping to overcome the limitations of their Internet connection. TLS-protected HTTP traffic cannot be cached by intermediaries, removing much of the benefit of the Web to what is arguably one of its most important target audiences.

It is now becoming more common for a proxy to man-in-the-middle TLS connections (see [tls-mitm] for an overview), to gain access to the application message flows. This represents a serious degradation in the trust infrastructure of the Web.

Worse is the situation where proxies provide a certificate where they inure the user to a certificate warning that they then need to ignore in order to receive service.

5. Principles for Consideration

Every HTTP connection has at least three major stakeholders; the user (through their agent), the origin server (possibly using gateways such as a CDN) and the networks between them.

Currently, the capabilities of these stakeholders are defined by how the Web is deployed. Most notably, networks sometimes change content. If they change it too much, origin servers will start using

encryption. Changing the way that HTTP operates therefore has the potential to re-balance the capabilities of the various stakeholders.

This section proposes several straw-man principles for consideration as the basis of those changes. Their sole purpose here is to provoke discussion.

5.1. Proxies Have a Legitimate Place

As illustrated above, there are many legitimate uses for proxies, and they are a necessary part of the architecture of the Web. While all uses of proxies are not legitimate - especially when they're interposed without the knowledge or consent of the end user and the origin - undesirable intermediaries (i.e., those that break the reasonable expectations of other stakeholders) are a small portion of those deployed used.

Note that while proxies have a legitimate place, it does not imply that they are an equal stakeholder to other parties in all ways; e.g., they do not have a natural right to access encrypted content, for example.

5.2. Security Should be Encouraged

Any solution needs to give all stakeholders - end users, networks and origin servers - a strong incentive towards security.

This has subtle implications. If networks are disempowered disproportionately, they might react by blocking secure connections, discouraging origin servers (who often have even stronger profit incentives) from deploying encryption, which would result in a net loss of security.

On the other hand, if networks are given carte blanche, it can destroy trust in the Web altogether. In particular, making it too easy to interpose a proxy (even if the user is "informed" by clicking through a dialogue) degrades the infrastructure in an unacceptable way.

5.3. Users Need to be Informed of Proxies

When a proxy is interposed, the user needs to be informed about it, so they have the opportunity to change their configuration (e.g., attempt to introduce encryption), or not use the network at all.

Proxies also need to be strongly authenticated; i.e., users need to be able to verify who the proxy is.

5.4. Users Need to be able to Tunnel through Proxies

When a proxy is interposed, the user needs to be able to tunnel any request through it without its content (or that of the response) being exposed to the proxy.

This includes both "https://" and "http://" URIs.

5.5. Proxies Can say "No"

A proxy can refuse to forward any request. Currently, the granularity of that "no" is per-URI for unencrypted requests, and per-IP (perhaps per-SNI) for encrypted requests.

5.6. Changes Need to be Detectable

Any changes to the message body, request URI, method, status code, or representation header fields of an HTTP message need to be detectable by the origin server or user agent, as appropriate, if they desire it.

This allows a proxy to be trusted, but its integrity to be verified.

5.7. Proxies Need to be Easy

It must be possible to configure a proxy extremely easily; the adoption of interception over proxy.pac/WPAD illustrates this very clearly.

5.8. Proxies Need to Communicate to Users

There are many situations where a proxy needs to communicate with the end user; for example, to gather network authentication credentials, communicate network policy, report that access to content has been denied, and so on.

Currently, HTTP has poor facilities for doing so. The proxy authentication mechanism is extremely limited, and while there are a few status codes that are defined as being from a proxy rather than the origin, they do not cover all necessary situations.

The Warning header field ([RFC7234], Section 5.5) was designed as a very limited form of communication between proxies and end users, but it has not been widely adopted, nor exposed by User Agents.

Importantly, proxies also need a limited communication channel when TLS is in use, for similar purposes.

Equally as important, the communication needs to clearly come from the proxy, rather than the origin, and be strongly authenticated.

5.9. Users Require Simple Interfaces

While some users are sophisticated in their understanding of Web security, they are in a vanishingly small minority. The concepts and implications of many decisions regarding security are subtle, and require an understanding of how the Web works; describing these trade-offs in a modal dialogue box that gets in the way of the content the user wants has been proven not to work.

Similarly, while some Web publishers are sophisticated regarding security, the vast majority are not (as can be proven by the prevalence of cross-site scripting attacks).

Therefore, any changes cannot rely upon perfect understanding by these parties, or even any great effort upon their part. This implies that user interface will be one of the biggest challenges faced, both in the browser and for any changes server-side.

Notably, the most widely understood indicator of security today is the "lock icon" that shows when a connection is protected by TLS. Any erosion of the commonly-understood semantics of that indicator, as well as "https://" URIs, is likely to be extremely controversial, because it changes the already-understood security properties of the Web.

Another useful emerging convention is that of "Incognito" or "private" mode, where the end user has requested enhanced privacy and security. This might be used to introduce higher requirements for the interposition of intermediaries, or even to prohibit their use without full encryption.

5.10. User Agents Are Diverse

HTTP is used in a wide variety of environments. As such there can be no assumption that a user is sitting on the other end to interpret information or answer questions from proxies.

5.11. RFC2119 Doesn't Define Reality

It's very tempting for a committee to proclaim that proxies "MUST" do this and "SHOULD NOT" do that, but the reality is that the proxies, like any other actor in a networked system, will do what they can, not what they're told to do, if they have an incentive to do it.

Therefore, it's not enough to say that (for example), "proxies have to honor no-transform" as HTTP/1.1 does. Instead, the protocol needs to be designed in a way so that either transformations aren't possible, or if they are, they can be detected (with appropriate handling by User Agents defined).

5.12. It Needs to be Deployable

Any improvements to the proxy ecosystem MUST be incrementally deployable, so that existing clients can continue to function.

6. Potential Areas to Investigate

Finally, this section lists some areas of potential future investigation, bearing the principles suggested above in mind.

6.1. Improving Proxy.Pac

Many of the flaws in proxy.pac can be fixed by careful specification and standardization, with active participation by both implementers and those that deploy it.

6.2. TLS Errors for Proxies

HTTP's use of TLS [RFC2818] currently offers no way for an interception proxy to communicate with the user agent on its own behalf. This might be necessary for network authentication, notification of filtering by hostname, etc.

The challenge in defining such a mechanism is avoiding the opening of new attack vectors; if unauthenticated content can be served as if it were from the origin server, or the user can be encouraged to "click through" a dialog, it has severe security implications. As such, the user experience would need to be carefully considered.

6.3. HTTP Errors for Proxies

HTTP currently defines two status codes that are explicitly generated by a proxy:

- 504 Gateway Timeout ([RFC7231], Section 6.6.5) - when a proxy (or gateway) times out going forward

- 511 Network Authentication Required ([RFC6585], Section 6) - when authentication information is necessary to access the network

It might be interesting to discuss whether a separate user experience can be formed around proxy-specific status codes, along with the definition of new ones as necessary.

6.4. TLS for Proxy Connections

While TLS can be used end-to-end for "https://" URIs, support for connecting to a proxy itself using TLS (e.g., for "http://" URIs) is spotty. Using a proxy without strong proof of its identity introduces security issues, and if a proxy can legitimately insert itself into communication, its identity needs to be verifiable.

6.5. Improved Network Information

Many of the use cases for proxies that modify content is transcoding or otherwise adapting that which is too "heavy" for the network it is transiting through.

If network operators made better, more fine-grained and timely information about their operational characteristics freely available, endpoints (server and client) could adapt requests and responses to reflect it, thereby removing the need for intermediation.

6.6. Improving Trust

Currently, it is possible to exploit the mismatched incentives and other flaws in the CA system to cause a browser to trust a proxy as authoritative for a "https://" URI without full user knowledge. This needs to be remedied; otherwise, proxies will continue to man-in-the-middle TLS.

6.7. HTTP Signatures

Signatures for HTTP content - both requests and responses - have been discussed on and off for some time.

Of particular interest here, signed responses would allow a user-agent to verify that the origin's content has not been modified in transit, whilst still allowing it to be cached by intermediaries.

Likewise, if header values can be signed, the caching policy (as expressed by Cache-Control, Date, Last-Modified, Age, etc.) can be signed, meaning it can be verified as being adhered to.

Note that properly designed, a signature mechanism could work over TLS, separating the trust relationship between the UA and the origin server and that of the UA and its proxy (with appropriate consent).

There are significant challenges in designing a robust, widely-deployable HTTP signature mechanism. One of the largest is an issue of user interface - what ought the UA do when encountering a bad signature?

7. Security Considerations

Plenty of them, I suspect.

8. Acknowledgements

This document benefits from conversations and feedback from many people, including Amos Jeffries, Willy Tarreau, Patrick McManus, Roberto Peon, Guy Podjarny, Eliot Lear, Brad Hill, Martin Nilsson and Julian Reschke.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[I-D.mbelshe-httpbis-spdy]
Belshe, M. and R. Peon, "SPDY Protocol", draft-mbelshe-httpbis-spdy-00 (work in progress), February 2012.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3040] Cooper, I., Melve, I., and G. Tomlinson, "Internet Web Replication and Caching Taxonomy", RFC 3040, January 2001.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, April 2012.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7234] Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014.

[W3C.NOTE-ct-guidelines-20101026] Rabin, J., "Guidelines for Web Content Transformation Proxies 1.0", World Wide Web Consortium NOTE NOTE-ct-guidelines-20101026, October 2010, <http://www.w3.org/TR/2010/NOTE-ct-guidelines-20101026>.

[https-everywhere] EFF, ., "HTTPS Everywhere", 2013, <https://www.eff.org/https-everywhere>.

[proxypac] various, ., "Proxy Auto-Config", 2013, <http://en.wikipedia.org/wiki/Proxy_auto-config>.

[tls-mitm] Jarmoc, J., "SSL/TLS Interception Proxies and Transitive Trust", 2012, <https://www.grc.com/miscfiles/HTTPS_Interception_Proxies.pdf>.

[wpad] Cohen, J., "Web Proxy Auto-Discovery Protocol", 1999, <http://tools.ietf.org/html/draft-ietf-wrec-wpad-01>.

Author's Address

Mark Nottingham

Email: mnot@mnot.net
URI: http://www.mnot.net/