BEHAVE Working Group
Internet-Draft
Intended status: Informational
Expires: September 7, 2010

D. Wing
Cisco

March 6, 2010

Coping with IP Address Literals in HTTP URIs with IPv6/IPv4 Translators

draft-wing-behave-http-ip-address-literals-02

Abstract

A small percentage of HTTP URIs contain an IPv4 address literal as the hostname which is not accessible to IPv6-only HTTP clients using an IPv6/IPv4 translator and DNS64. This document proposes a workaround for this problem using an HTTP proxy to handle that traffic.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Two of the translation scenarios involve IPv6-only hosts accessing IPv4-only hosts (Scenario 1, "an IPv6 network to the IPv4 Internet" and Scenario 5, "an IPv6 network to an IPv4 network" [I-D.ietf-behave-v6v4-framework]). For this to work, the IPv6-only host sends a DNS AAAA query and receives a synthesized AAAA response. While it is common practice to use domain names in many application protocols, most applications do not require using domain names. IPv4 address literals occur in HTTP URI hostname fields (e.g., http://192.0.2.1) on the Internet (see Appendix B) and on some corporate networks. Such occurrences do not cause problems today, because both IPv4 hosts and dual-stack hosts can connect to those addresses just fine. However, those IPv4 address literals are inaccessible to IPv6-only clients using an IPv6/IPv4 translator and DNS64.

This document proposes a workaround to this problem when an HTTP browser, running on an IPv6-only host, encounters an HTTP URI containing an IPv4 address literal (instead of containing a domain name).

2. Proposed Workaround

Nearly all modern web browsers can be configured with a Proxy auto-config (PAC) [PAC] file. A PAC allows a browser to have an HTTP proxy handle traffic to a host with an IPv4 address literal, while allowing direct access (through the IPv6/IPv4 translator) for the majority of traffic, as shown in Appendix A. With this workaround, an IPv6-only HTTP client can access HTTP URIs that contain IPv4 address literals. The HTTP proxy needs to be able to send packets to the IPv4 Internet, and can be located parallel to the translator (as shown in Figure 1) or on either side of the IPv6/IPv4 translator, including in the host itself. To be located on the IPv6-only side of the translator, the proxy needs to understand how to formulate an IPv6 address from an IPv4 address [I-D.wing-behave-learn-prefix].

```
                                      |
          <----------IPv6------------>|<---------IPv4----------->
                                      |
          +-----------+ proxied  +----------+
          |           +--------->|HTTP Proxy+-
          |IPv6-only  |          +----------+ \     ,-----------.
          |    web    |               |        >-->(IPv4 Internet)
          |  browser  | direct   +----------+ /     `-----------'
          |           +--------->|IPv6/IPv4 +-
          +-----------+          |Translator|
                                 +----------+
                                      |
```

Figure 1: Network Diagram showing HTTP proxy and Translator

The following diagram shows the translator located on the IPv4 Internet:

```
                                      |
                                      |
          <----------IPv6------------>|<---------IPv4----------->
                                      |             +----------+
                                      |          +->|HTTP Proxy+
          +-----------+               |         /   +----+-----+
          |           |               |        /         V
          |IPv6-only  |          +----------+ /     ,-----------.
          |    web    +--------->|IPv6/IPv4 ++---->(IPv4 Internet)
          |  browser  |          |Translator|      `-----------'
          |           |          +----------+
          +-----------+               |
```

Figure 2: Network Diagram showing HTTP proxy on IPv4 network

The Web Proxy Autodiscovery Protocol (WPAD) [I-D.ietf-wrec-wpad] is useful to autoconfigure web browsers for non-technical users or for a large community of users (e.g., inside of an enterprise).

3. Disadvantages of Workaround

While the workaround is helpful, the PAC and WPAD workarounds have several disadvantages:

- o operating an HTTP proxy, even for the relatively small amount of the HTTP traffic that contains IPv4 address literals, is generally considered more resource-intensive than operating an IPv6/IPv4 translator because the HTTP proxy has to terminate a TCP connection and originate a separate TCP connection and shuffle

data between them. For this reason alone, the PAC workaround described in this document is inferior to the web browser handling IPv4 native addresses itself [I-D.wing-behave-learn-prefix].

- The client's IPv4 address will be different for traffic going through the IPv6/IPv4 translator versus going through the HTTP proxy. While not yet seen in practice, it is anticipated that some HTTP servers use the IP addresses for AAA (authentication, authorization and accounting) purposes, such as encoding the IP address into a cookie or URI. Thus, the client's different IPv4 address may break interaction with those servers. Also see Appendix C.

- The workaround only provides assistance to IPv4 address literals in hostnames. It does not help IPv4 address literals that appear, for various reasons, in the URL path or query string (e.g., http://www.example.com?host=1.2.3.4>, Java, or Javascript.

- Interworking an existing PAC file with the new functionality described in this document may be difficult.

- WPAD increases the attack surface.

- Both PAC and WPAD are a de facto standards.

4. Security Considerations

WPAD increases the attack surface, because of how WPAD uses unauthenticated DHCP or DNS to find the PAC file, searches domain names for PAC files, and because the PAC file is retrieved via unauthenticated HTTP.

5. IANA Considerations

This document requires no IANA actions.

6. Acknowledgements

Thanks to Stig Venaas and Andrew Yourtchenko for their review comments. Thanks to Shin Miyakawa for suggesting this same technique is useful for IPv4-only hosts to connect to IPv6 address literals. Thanks to Cameron Byrne for highlighting the problem with the client's apparent IPv4 address.

7. Informative References

[Alexa] Alexa, "Top 1,000,000 Global Sites", September 2009, <http://www.alexa.com/topsites>.

[I-D.ietf-behave-v6v4-framework] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-07 (work in progress), February 2010.

[I-D.ietf-wrec-wpad] Gauthier, P., Cohen, J., Dunsmuir, M., and C. Perkins, "Web Proxy Auto-Discovery Protocol", draft-ietf-wrec-wpad-01 (work in progress), July 1999.

[I-D.wing-behave-learn-prefix] Wing, D., "Learning the IPv6 Prefix of a Network's IPv6/IPv4 Translator", draft-wing-behave-learn-prefix-04 (work in progress), October 2009.

[PAC] Wikipedia, "Proxy auto-config", September 2009, <http://en.wikipedia.org/wiki/Proxy_auto-config>.

Appendix A. Example PAC files

A simple example of a PAC file that causes HTTP or HTTPS URIs containing IPv4 address literals to be proxied to v6v4-proxy.example.net on port 8080. This would be useful for an IPv6-only client that needs to access content with IPv4 address literals in the HTTP URI:

```
function FindProxyForURL(url, host) {
var regexpr = /^https?:\/\/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/
if (regexpr.test(url))
    return "PROXY v6v4-proxy.example.net:8080";
else
    return "DIRECT";
}
```

Figure 3

A simple example of a PAC file that causes HTTP or HTTPS URIs containing IPv6 address literals to be proxied to v6v4-proxy.example.net on port 8080. This would be useful for an IPv4-only client that needs to access content with IPv6 address literals in the HTTP URI.

```
function FindProxyForURL(url, host) {
var regexpr = /^https?:\/\/\[[0-9a-fA-F:\.\]*\]/
if (regexpr.test(url))
    return "PROXY v4v6-proxy.example.net:8080";
else
    return "DIRECT";
}
```

Figure 4: Example PAC for IPv4-only client

Appendix B. HTTP IPv4 Address Literals on the Internet

There has been some doubt that HTTP URIs on the Internet contain hostnames with IPv4 address literals. This section provides some scripts which demonstrate the relatively low -- but prevalent -- existence of IPv4 address literals.

An examination of Alexa's top 1 million domains [Alexa] at the end of August, 2009, showed 2.38% of the HTML in their home pages contained IPv4 address literals. This can be verified with:

```
wget http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
unzip top-1m.csv.zip
cat top-1m.csv |
  cut -d "," -f 2 |
  xargs -I % -n 1 -t wget -nv % -O - --user-agent="Mozilla/5.0" |
  grep -E "https?://[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
```

Of the top 1 million websites at the end of August, 2009, 3455 (0.35%) of them are IPv4 address literals (e.g., http://192.0.2.1). This can be verified with:

```
wget http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
unzip top-1m.csv.zip
grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
  top-1m.csv | wc
```

Appendix C. Same IP address for proxied HTTP

By placing the HTTP proxy behind the same NAT64 that the handset is

using, it is possible -- although fragile -- to cause the HTTP proxied traffic to have the same IPv4 address as non-proxied HTTP traffic:

```
                             |
   <----------IPv6------------->|<---------IPv4----------->
                +-----+         |
                |HTTP |         |
   +-------+    |Proxy|         |
   |IPv6-  |   /+-----+\        |
   |only   |  /         \  +----------+ /     ,-----------.
   |web    +-+-----------\>|IPv6/IPv4 ++---->(IPv4 Internet
   |browser|               |Translator|       `-----------'
   +-------+               +----------+
                             |
```

Figure 5: Network Diagram showing HTTP proxy behind 6/4 Translator

The IPv6 host is configured with a PAC file. The HTTP proxy changes IPv4 address literals into IPv6 address literals, but it sends and receives IPv6 packets on the wire. The HTTP proxy could do that by sticking a fixed string in front of the IPv4 address literal to generate the IPv6 destination address.

Then, if the source address of the HTTP proxy is hashed like the source address of the host, the NAT64 will chose the same egress IPv4 address.

Author's Address

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
USA

Email: dwing@cisco.com