W-Bad: Interception, Inspection, and Interference with Web Proxy Auto-Discovery (WPAD)

Casey Deccio
Computer Science Department
Brigham Young University
Provo, UT
casey@byu.edu

***Abstract*—The Web Proxy Auto-Discovery Protocol (WPAD) was developed decades ago as a way to automatically configure Web proxies for clients in a given network environment. However, almost since its inception it has been identified as being vulnerable—both in design and implementation. Yet even today it is found in popular operating systems and browsers. In this paper, we chronicle the history of the domain name `wpad.domain.name`, which has caused grief for users worldwide, due to router firmware bugs and efforts to intercept, inspect, and interfere with Web requests. We measure its delegation history, the manner in which it was opportunistically used for abuse, and the set of vulnerable clients over time.**

I. Introduction

The Web Proxy Auto-Discovery Protocol (WPAD) was drafted over 20 years ago as a mechanism for allowing operating systems (OSes) and Web browsers alike to automatically identify the server designated for proxying Hypertext Transfer Protocol (HTTP) requests in a network environment [1]. While the protocol was never formally adopted by standards bodies (i.e., remains only an "Internet Draft" in the Internet Engineering Task Force [IETF] archives), it is implemented in almost all major OSes and browsers, and it is enabled by default in some.

While the goals of WPAD were useful—especially in the day when HTTP proxy use was pervasive—its very design has long been criticized as vulnerable. The protocol can be exploited in multiple ways. One particular exploit is based on the following principles. WPAD behavior is based on configuration information provided by the network on which the system currently resides. While the network itself might not be malicious, it might be misconfigured in a way that is not noticeable under normal circumstances. A third party might take advantage of this unnoticed misconfiguration to interfere with user HTTP traffic.

This is the story of `wpad.domain.name`, which is perhaps the first widespread exploit of WPAD. As early as 2010, home routers were inadvertently distributing the Domain Name System (DNS) suffix `domain.name` to systems on their networks. This suffix was the basis for WPAD implementations to determine HTTP proxy location; specifically, they would use the DNS name `wpad.domain.name` for proxy discovery, per the protocol. For years, that domain name did not resolve, such that no ill effect was experienced by users. But in 2017, `wpad.domain.name` began resolving to an IP address. This effectively allowed the registrants of `wpad.domain.name` to dictate proxy settings for home users around the globe. In turn, this control opened the door for interception, inspection, and interference with user HTTP traffic—in a clear violation of user privacy and security.

This paper documents the history of `wpad.domain.name`. Using historical passive DNS databases, active DNS and HTTP measurements, and anecdotal experiences from Internet forums, we create a timeline of events that collectively tell the story of how the network changed over time to create an unsafe environment for vulnerable clients and end users. Using active DNS and HTTP measurements, we describe the nature of the attack, and the impact on end users.

II. Background

A. Domain Name System

The Domain Name System (DNS) is the system whereby domain name are translated to IP addresses. This process involves a *stub resolver*, which issues queries to a *recursive resolver*. The recursive resolver, in turn, issues queries to one or more *authoritative servers* to find the answer, which it then returns to the stub resolver.

A DNS query consists of a domain name and a query type. For example, a query for `www.example.com` with type `A` (address) yields an IP address, e.g., 192.0.2.1. A query for `example.com` with type `NS` (name server) yields a set of names corresponding to the servers authoritative for `example.com`, i.e., where `example.com` has been delegated. For example, these servers might included `ns1.example.com` and `ns2.example.com`.

B. HTTP Proxies and Proxy Auto-Configuration

HTTP proxies have historically been (and in many cases continue to be) useful for enterprises for localizing download of Web content and/or the monitoring of incoming and outgoing HTTP traffic. When an HTTP proxy is used by a client, the client sends its request to the HTTP proxy, and the proxy issues the request to the HTTP server (i.e., the target) on behalf of the client. This allows the proxy to cache content or redirect clients in the case that remote content is malicious or otherwise

978-3-903176-58-4

should be blocked. With HTTPS requests, a proxy can still be used, but instead of issuing the full request to the proxy over HTTP, the client uses the `CONNECT` method, with which the proxy establishes (only) a TCP connection with the target server. At this point, the proxy simply relays data between client and target server. The client then establishes a secure connection with the target server (through the proxy) using TLS, so all the proxy sees of the communication between client and server is ciphertext.

In Web browsers and OSes, the use of an HTTP proxy is dictated in one of two ways: either a specific HTTP proxy is always used, or the HTTP proxy (if any) is determined at the time of the HTTP request using a proxy auto-configuration (PAC) script. Designating a specific proxy is very rigid and often unsuitable when proxy use is dependent on the request. For example, it might be desirable for the client operating in a corporate network to only use a proxy when accessing sites external that network. A PAC script introduces additional flexibility in this regard. A PAC script is retrieved over HTTP at the Uniform Resource Locator (URL) with which it is associated, e.g., `http://www.example.com/proxyconfig.pac`. With each HTTP request, the URL and host associated with the request are run through the `FindProxyForURL()` function contained in the script. For example, the following would instruct a client to use 192.0.2.10 as a proxy whenever the host in the URL is not a subdomain of `example.com` and does not resolve to something within 192.0.2.0/24:

```
function FindProxyForURL(url, host) {
  if (dnsDomainIs(host, ".example.com") ||
          isInNet(host, "192.0.2.0",
                  "255.255.255.0")) {
    return "DIRECT";
  } else {
    return "PROXY 192.0.2.10:8080";
  }
};
```

It is notable to mention that the PAC specification is outdated, and it only has recently starting to meet more updated protocol expectations, such as support for IPv6 [2], [3].

C. Web Proxy Auto-Discovery (WPAD)

While a PAC script can provide flexibility for HTTP clients, several issues remain unaddressed. For example, a mobile system might need to use a different proxy configuration in one environment than it uses in another environment. Also, in any environment, there is a question about how to discover the proxy settings without having to hard-code them. One solution for this is the use of the Web Proxy Auto-Discovery Protocol (WPAD) to find a PAC script. With WPAD, the URL of the PAC script can be dynamically discovered, using a technique that is described hereafter. Even so, a client configured with WPAD still ultimately uses a PAC script and therefore inherits some of the challenges associated with explicitly-configured PAC script (e.g., lack of IPv6 support).

WPAD was proposed in an Internet Draft that dates back to 1999 [1]. While the draft was never formalized into a Request for Comments (RFC)—the de facto standard vehicle for many Internet protocols—it was integrated into nearly every popular Web browser. At the time of writing, the most current versions of Mozilla Firefox (v113.x) and Google Chrome (v113.x) support WPAD. Additionally, the latest versions of operating systems such as MacOS 13 and Windows 11 offer system-wide proxy settings that include WPAD, which settings can be used by both system-specific browsers (e.g., Safari and Edge) and third-party browsers. While WPAD is not currently the default in many implementations, it is the default system-wide proxy setting on Windows systems.

With WPAD, a browser or operating system discovers an HTTP proxy configuration by systematically issuing DNS queries, according to the following pattern. The software retrieves the DNS suffix configured on a given system—usually the suffix associated with the organization in which it operates. The suffix is often provided by the Dynamic Host Control Protocol (DHCP). Using that suffix, it forms a DNS domain name by prepending the `wpad` label. For example, the domain name made from the suffix `foo.example.com` would be `wpad.foo.example.com`. An attempt is made to resolve the resulting domain name to an IP address. If not successful (i.e., because the name does not exist or there are no `A` or `AAAA` records at the domain name—for IPv4 or IPv6 addresses, respectively), then the left-most label is removed from the suffix and `wpad` prepended again. For example, a failed attempt at resolving `wpad.foo.example.com` results in an attempt to resolve `wpad.example.com`. This process continues until resolution succeeds. At the point that resolution succeeds, the software opens a connection to the IP address to which the name resolved and issues an HTTP `GET` request for the path `/wpad.dat`. The Web server returns a PAC script containing the proxy configuration that should be used by clients.

D. Abuse of WPAD

The contents of the PAC script discovered and retrieved using WPAD can direct end-user Web clients to a proxy designed by the authors of the PAC script. The HTTP requests of Web clients using that script can then be potentially observed, intercepted, manipulated, redirected, or dropped. This is effectively a man-in-the-middle (MITM) attack. Obtaining control of a client's HTTP requests, therefore, becomes a matter of controlling the PAC content. Controlling the PAC content requires control of the Web server corresponding to the WPAD domain name. Leading clients to retrieve the PAC content from the Web server involves controlling the DNS responses for the domain names being resolved as part of WPAD.

In the attack discussed herein, third parties registered a domain name actively queried by Web clients around the world, in connection with WPAD. They configured their authoritative DNS servers to return an `A` record for a Web server under their control, which returned PAC content directing clients to use their server as an HTTP proxy. The domain name was `wpad.domain.name`. The third party was Gransy s.r.o.

Those affected included users of the D-Link home routers, and possibly others.

III. Previous Work

Avenues to opportunistically exploit WPAD have been explored for the past 20 years. In 2003, research from Wessels et al. showed that `wpad` was ranked sixth in queries observed at the root servers for top-level domains (TLDs) that do not exist [4]. While these queries alone do not directly correlate to exploits, they are indicative of the greater WPAD problem. There is no good reason why HTTP clients should be looking for proxy settings via WPAD from the root domain, i.e., using the query name `wpad`. Similarly, HTTP clients should not generally be looking for proxy settings via WPAD from a TLD (e.g., `com`, `net`, or `cn`) [5].

In 2016, Chen et al. measured the potential for opportunistic exploit using WPAD in new generic TLDs [6]. They found that `wpad` ranked fourth in terms of first (i.e., left-most) label of queries for nonexistent TLDs observed at the root servers, yielding approximately 20 million queries daily. They also found that two-thirds of the new gTLDs [7] delegated prior to August 2015 showed WPAD activity prior to their delegation, increasing the potential for opportunistic exploit. This work resulted in an advisory by the United States Government's Cybersecurity and Infrastructure Security Agency (CISA) [8]. In 2017, Chen et al. performed a more general study related to client vulnerability with discovery protocols, including WPAD, within gTLDs [9]. They found that out of the 48 services they identified as potentially exploitable, 45 (94%) exposed vulnerabilities in popular clients.

Kristoff et al. studied the potential for opportunistic exploit of the intra-site automatic tunnel addressing protocol (ISATAP), a discovery protocol very similar to WPAD, but used for IPv6 transition [10]. They conduct a measurement of legacy IPv6 transition mechanisms, including end hosts, servers, and network infrastructure, and they discover attack vectors for exploiting millions of Internet-connected hosts.

Several works have directly addressed the `wpad.domain.name` incident described in this paper. In 2023, Boulila et al. published a survey on the different research related to WPAD and its security issues [11]. They detail the WPAD protocol itself, describe attack scenarios, provide an overview of research publications on the topic, and briefly summarize the `wpad.domain.name` incident. Two other documents refer to the `wpad.domain.name` incident: a contribution to Akamai's Spring 2018 State of the Internet by Kulberg and a blog article by Persch [12], [13].

We supplement the analysis provided by the collective reports on `wpad.domain.name` with additional measurements, insights, and perspectives. For example, the State of the Internet Report indicated that the `/wpad.dat` file was removed between the time of the CISA alert (2016) [8] and their report (2018) [12], and Boulila reports that the proxy was unavailable after July 2021 [11]. However, we found both the WPAD content and the proxy service to be active through (at least) October 2021 (see Section VI-A and Section VI-B). Finally, Persch observed that the content at `http://wpad.domain.name/wpad.dat` was different when requested from Germany vs. from Malaysia [13]. The analysis in this paper involves issuing HTTP requests from 56 probes in 26 distinct countries to investigate the issue in greater depth (see Section VI-B3).

IV. Problematic Home Router Configuration

Home routers typically run a DHCP service for their clients. In addition to distributing IP addresses, these routers also typically provide a DNS suffix. The D-Link DIR 615 is one home router that provides such services. It also includes a Web-based configuration console wherein the DNS suffix can be set. The default domain for some versions of this router was `domain.name`, no doubt intended as an (innocuous) example text for the administrator, providing a description of a value that might appropriately go into the field [14].

Of course, neither the routers nor the clients behind them are affiliated with the DNS suffix `domain.name`. Yet because this is the DNS suffix provided to clients behind the routers, applications and protocols that require the DNS suffix, including WPAD, will use `domain.name` as that suffix. Thus, for software using WPAD protocol, `wpad.domain.name` becomes the target of resolution name for HTTP proxy discovery.

For affected systems, as long as `wpad.domain.name` does *not* exist in the DNS, the fact that this DNS suffix is being unwittingly distributed to end-user systems is harmless. However, if `wpad.domain.name` *were* to resolve to an IP address, then WPAD software would issue an HTTP `GET` request for the following path at that IP address: `/wpad.dat`. Assuming a PAC script exists at that URL, any systems having retrieved it would be subject to the HTTP proxy rules found therein. Thus, the combination of a system that uses WPAD and a home router that hands out a DNS domain for which the `wpad` subdomain exists in the DNS creates the perfect configuration for a security and privacy vulnerability.

V. Phase 1: Initial Delegation of `wpad.domain.name`

In June 2012 `wpad.domain.name` was registered and delegated from `name`. This discovery was detected using DNSDB, a database of historical domain name-to-IP address mappings created from passive DNS [15]. In essence, every DNSDB entry corresponds to a query from recursive resolver to authoritative server for which the response contained *some* mapping—as opposed to a name error (`NXDOMAIN`) response. Such queries are typically the result of some end-system application (e.g., Web browser) making a DNS query. In the case of `wpad.domain.name`, these queries are often associated with WPAD systems behind vulnerable routers. This is discussed more in Section VII. DNSDB records do not contain any identifying information of the resolvers that make the observed DNS query. However, they do include the aggregate query count for a given domain name and type over a given time period. A summary of delegations associated with `wpad.domain.name` is found in Table I.

	Dates	NS Name(s)	Count
Phase 1	06/2012–07/2012	{a,b,c}.gandi.net	554
	06/2012–06/2016	ns{1,2}.wpad.domain.name	15M
	06/2016–09/2017	(No delegation)	
Phase 2	09/2017–10/2017	ns{,2}.parktons.com	118K
	11/2017–12/2021	ns{1,2}.anycastdns.cz	8.1M

TABLE I: Historical delegation information for wpad.domain.name.

The initial delegation of wpad.domain.name was to a set of three servers having suffix gandi.net. However, these NS records only existed for about 10 days, and only about 500 queries were observed. From June 2012 to June 2016, wpad.domain.name was delegated to ns1.wpad.domain.name and ns2.wpad.domain.name. During this four-year period 15 million recursive-to-authoritative DNS queries were observed. We refer to this time period as "Phase 1".

While there is no evidence to identify a motive for registering wpad.domain.name during Phase 1, there is also no evidence to suggest that there was any threat to security and privacy. Though wpad.domain.name was delegated continuously between June 2012 and June 2016, the delegation was mostly harmless because—with only a small exception—wpad.domain.name did not resolve to an *IP address* during this time (i.e., no A record existed). As such, no WPAD client would attempt to receive an HTTP request from which they might receive a potentially malicious PAC script. The exception to this was the 10 days in June 2012 when wpad.domain.name resolved to an IP address used by Gandi—a domain registrar—for domain "parking" [16]. Based on the NS names used during this time (a,b,c.gandi.net), the domain was likely registered through Gandi, and domain parking was simply the default until changed. The A record at wpad.domain.name resulted in the resolution of wpad.domain.name by WPAD clients, followed by HTTP requests to the Web server at that IP address. However, because this Web server was dedicated to domain parking, there was no PAC file returned, and the results were mostly innocuous, even if the activity was noticed by some users (see Section VI-B2).

In June 2016, responses with NS records for wpad.domain.name stopped being observed. We note that this does not mean that *queries* for wpad.domain.name stopped (see Section VII). Because domain name registrations are renewed on an annual basis, the fact that responses corresponding to a single set of NS records were first observed in June 2012 and stopped being observed in June 2016 is consistent with the original registration lapsing and the delegation being removed from name.

VI. Phase 2: Abuse of wpad.domain.name

In September 2017 wpad.domain.name was delegated to a new set of servers, suggesting a new registration of wpad.domain.name. For the first six days of the new registration, an initial set of NS records were observed: ns.parktons.com and ns2.parktons.com. Approximately 117K responses containing these records were observed over just six days. However, from October 2017 onward, only the following NS records were observed for wpad.domain.name: ns1.anycastdns.cz and ns2.anycastdns.cz. Approximately 15 million queries were observed in connection with these records through December 2021. We refer to this as "Phase 2".

A. Opportunistic WPAD Name Resolution

Unlike Phase 1, historical A records existed for wpad.domain.name throughout the entirety of Phase 2. We use evidences from both passive and active analysis to support this finding.

1) Historical Passive DNS: DNSDB entries showed that during the first six days of the Phase 2 delegation (i.e., corresponding to NS records ending in parktons.com), A records mapped wpad.domain.name to IP addresses 31.192.228.197, 159.253.25.197, and 159.253.28.197. Using Routeviews and py2asn, we determined that all of these IP addresses were historically associated with AS43948 ("GleSYS-AS"). A total of about 9K DNS queries were observed for which those IP addresses were returned. While it is unclear whether or not these IP addresses are used for parking, reports indicate that a simple PAC file was being returned from http://wpad.domain.name/wpad.dat, unlike the case with the initial 2012 delegation (see Section V and Section VI-B).

From November 2017 to May 2019, wpad.domain.name resolved to IP addresses in AS16276 ("OVH"). While the IP addresses observed during that 18 months changed twice, the 16-bit prefixes were consistent throughout: 37.187.0.0/16 and 91.121.0.0/16.

For the six-month period between July 2019 and January 2020, wpad.domain.name resolved to the IP address 95.168.185.183, which is associated with AS205544 ("LEASEWEB UK LIMITED"). During this time, approximately 1.5M query DNS queries were observed resulting in that IP address. For the three-month period that followed (January to April 2020), wpad.domain.name resolved to 127.0.0.1. This was possibly a precautionary measure, to interrupt and prevent any malicious activity, but we cannot confirm this with the data. Following that, for a brief six days, wpad.domain.name resolved to 94.130.18.141, an IP address associated with AS24940 ("Hetzner Online GmbH"). From this latest mapping, about 36K DNS responses were observed.

From October 2020 through December 2021, wpad.domain.name resolved to 185.38.111.1, an IP associated with AS60592, "Gransy s.r.o.". From October

2020 to December 2021 2.3M queries were observed associated with this IP address.

2) Active DNS Queries: We supplemented our analysis of the historical passive DNS with active DNS queries for `wpad.domain.name`. Using the Ark platform [17] made available by the Center for Applied Internet Data Analysis (CAIDA), we issued a DNS lookup of type `A` for `wpad.domain.name` from 56 vantage points (Ark probes) located in 26 different countries in September and October of 2021. Each DNS lookup was performed by issuing a recursive DNS query to the recursive resolver with which each probe was locally configured. The results of the DNS lookup were consistent across all vantage points: in every case, `wpad.domain.name` resolved to the IP address 185.38.111.1. This is the same IP address to which `wpad.domain.name` was observed in the DNSDB history from October 2020 through December 2021.

B. Opportunistic WPAD HTTP Response

While DNS resolution is potentially problematic, the privacy and security concerns with actually responding to a WPAD HTTP request are even higher. Throughout Phase 2, there is evidence not only that `wpad.domain.name` resolved to an IP address, but also that an HTTP server was accepting requests and providing PAC scripts in response to requests for `/wpad.dat`. We describe evidences using historical HTTP records, online forums, and active HTTP requests.

1) Historical HTTP: The closest thing to an HTTP equivalent for DNSDB is the Internet Archive or "Wayback Machine" [18]. The Internet Archive has just two records for `http://wpad.domain.name/wpad.dat`: one dated March 21, 2021, and one dated September 24, 2021 [19]. The response content for the entries at both dates is empty. This empty content is consistent with receiving an HTTP response with empty response body. While this analysis does not reveal HTTP content, let alone PAC scripts, it reinforces findings in subsequent sections, where such content was found (see Section VI-B3).

2) Online Forums: While the Internet Archive has little historical HTTP data related to `http://wpad.domain.name/wpad.dat`, there are other data sources. Web-accessible mailing list archives and support forums show reports of interference related to `wpad.domain.name` as early as 2012 and as recent as September 2021. We reference 12 such reports (in a total of 11 websites), summarized in Table II and Table III.

The only report of problems during Phase 1 was during the first days of delegation, when `wpad.domain.name` resolved to an IP address hosting a parking page for a domain registrar. The report indicated that the HTTP response code was a 404 error, from which we infer that no PAC script would have been returned.

However, there are 11 reports of problems during Phase 2, all of which involve PAC content being returned. Most of the reports were from Brazil or South Africa. Of the 11 reports, 8 include the actual PAC content returned by the WPAD HTTP server. These give a history of historical responses for HTTP requests for `http://wpad.domain.name/wpad.dat`. The earliest and simplest PAC contents were noted in a mailing list post dated September 27, 2017:

```
function FindProxyForURL(url, host) {
    return 'PROXY 185.82.212.95:8080; DIRECT';
}
```

This configuration directs browsers and other HTTP clients using WPAD to use the HTTP proxy at 185.82.212.95 port 8080 for all HTTP requests. Not surprisingly, this IP address is associated with the same AS as 185.38.111.1, which is the IP address to which `wpad.domain.name` most recently resolved: AS60592 ("Gransy s.r.o."). See Section VI-A1 and Section VI-A2.

The contents of the PAC file at `http://wpad.domain.name/wpad.dat` have changed over time, according the reports. At least four variants have been reported, with the changes most often adding more exceptions to the proxy conditional. The most recent version (June 8, 2021) was the following:

```
function FindProxyForURL(url, host) {
 if (isPlainHostName(host) ||
   dnsDomainIs(host, ".windowsupdate.com") ||
   dnsDomainIs(host, ".microsoft.com") ||
   dnsDomainIs(host, ".baidu.com") ||
   dnsDomainIs(host, ".kaspersky.com") ||
   dnsDomainIs(host, ".axaltacs.net") ||
   dnsDomainIs(host, ".live.com") ||
   dnsDomainIs(host, ".drivergenius.com") ||
   isInNet(host, "10.0.0.0", "255.0.0.0") ||
   isInNet(host,
     "172.16.0.0", "255.255.224.0") ||
   isInNet(host,
     "192.168.0.0", "255.255.0.0") ||
   isInNet(host,
     "127.0.0.0", "255.0.0.0"))
  return "DIRECT";
 else
  return 'PROXY 185.38.111.1:8080';
 }
}
```

This effectively tells the HTTP client that except for a handful of destination domains (e.g., `windowsupdate.com`) and local or private addresses (e.g., 127.0.0.0/8), proxy HTTP requests through 185.38.111.1 port 8080.

3) Active HTTP Requests: The interference and exploit reported by users and administrators around the world indicated that the HTTP server at `wpad.domain.name` returns PAC content directing HTTP clients to use a designated proxy. However, our initial experimentation with issuing HTTP requests for `http://wpad.domain.name/wpad.dat` resulted in empty HTTP responses:

```
HTTP/1.1 200 OK
Date: Fri, 08 Oct 2021 20:06:23 GMT
Content-Length: 0
```

These empty responses are consistent with the behavior observed by the Internet Archive (see Section VI-B1) but at

Date	Country	URL
2012-06-27	Unknown	https://www.wilderssecurity.com/threads/please-help-with-this-outbound-connection-problem.327034/
2017-09-27	Brazil	https://eng.registro.br/pipermail/gter/2017-September/071659.html
2017-09-28	Brazil	http://mm.icann.org/pipermail/gnso-newgtld-wg-wt4/2017-September/000182.html
2017-09-28	Unknown	https://www.reddit.com/r/networking/comments/732r5n/anybody_else_having_issues_with_wpaddomainname/
2017-11-24	Unknown	https://social.technet.microsoft.com/Forums/windowsserver/en-US/e49a45f0-6875-4285-a1d4-5d7de0c63c53/wpad-entry-cannot-browse-websites-using-edge-and-chrome?forum=win10itpronetworking
2017-11-24	Brazil	https://medium.com/@thiago.palmeira/domain-name-wpad-name-collision-exploit-86df7f61d5e5
2021-01-05	Unknown	https://www.bleepingcomputer.com/forums/t/740178/was-my-router-compromised-wpad-attack/
2021-01-09	Italy	https://www.hwupgrade.it/forum/showthread.php?t=2931491
2021-01-26	South Africa	https://mybroadband.co.za/forum/threads/internet-browsing-on-telkom-adsl--not-working-when-check-for-proxy-automatically-is-enabled.1121074/
2021-06-08	South Africa	https://mybroadband.co.za/forum/threads/pure-dsl-internet-on-laptop-slow-but-fast-on-android.1140307/
2021-03-24	Morocco	https://forum.kaspersky.com/topic/malicious-object-detected-wpaddat-wpaddomainname-trojanscriptagentdc-merged-16171/
2021-09-17	Unknown	https://forum.kaspersky.com/topic/malicious-object-detected-wpaddat-wpaddomainname-trojanscriptagentdc-merged-16171/

TABLE II: Date and country of reports of problems associated with `wpad.domain.name`.

	Number of Reports	
Country	**Phase 1**	**Phase 2**
Unknown	1	4
Brazil	0	3
South Africa	0	2
Italy	0	1
Morocco	0	1
	1	11
Total	**12**	

TABLE III: Counts of reports of problems associated with `wpad.domain.name` during Phase 1 (see Section V) and Phase 2 (see Section VI). All reports associated with Phase 2 involve a PAC script being returned by `wpad.domain.name`, whereas the only report associated with Phase 1 returned a 404 error.

Country	PAC	Country	PAC	Country	PAC
Argentina	○	Israel	○	South Africa	●
Bangladesh	○	Japan	●	Spain	○
Brazil	○	Madagascar	○	Switzerland	○
Canada	○	Mauritius	○	Tanzania	●
China	○	Mexico	●	Ukraine	○
Costa Rica	○	Netherlands	○	United Kingdom	○
Czech Republic	○	New Zealand	○	United States	○
Germany	○	Paraguay	○	Zambia	●
Hungary	○	Serbia	○		

TABLE IV: Result of issuing HTTP requests for `http://wpad.domain.name/wpad.dat` from 56 vantage points in 26 countries. Each country is marked according to whether its HTTP request returned PAC content (●) or not (○).

odds with the online reports (see Section VI-B2). To reconcile these different HTTP response behaviors, we analyzed the behavior of `wpad.domain.name` when HTTP requests are made from different vantage points.

Using CAIDA's Ark platform, we issued an HTTP request for `http://wpad.domain.name/wpad.dat` from each of the same probes that we used for DNS lookups (see Section VI-A2)—56 probes in 26 distinct countries. These HTTP lookups were all made in September and October 2021. While the DNS resolution was consistent from all vantage points (see Section VI-A2), the HTTP response behavior varied. From 50 (89%) of the 56 probes, representing 21 (81%) of the 26 countries, the HTTP response consisted of empty content.

The remaining six probes, from five countries, received an HTTP response with PAC content that varied only slightly from that most recently reported on public forums (see Section VI-B2). Specifically, it includes two additions to the list of domain exceptions: `googlevideo.com` and `youtube.com`.

These five countries for which this content was returned were Japan (2 probes), Mexico, Zambia, South Africa, and Tanzania. The entire list of countries from which HTTP requests were made are shown in Table IV. The results show that the sever at `wpad.domain.name` was clearly returning PAC content, but it was dependent on origin of the HTTP request.

We note that the probes selected were all that were available to us on the Ark platform at the time of analysis. While a comprehensive study might have included HTTP requests from more probes at additional locations, the purpose of this analysis is to show that HTTP requests for `http://wpad.domain.name/wpad.dat` resulted in different content, based on the origin of the request.

C. HTTP Interception, Inspection, and Interruption

We now turn our attention to the HTTP proxy itself, i.e., the one designated in the PAC script returned by the WPAD HTTP server at `http://wpad.domain.name/wpad.dat`. We wanted to answer the following questions about the proxy. Was it receiving and acting on HTTP requests? Was it modifying HTTP requests? Was it modifying HTTPS requests? To answer these questions, we issued HTTP requests to the top 500 sites from the Alexa Top Sites list, as of October 2021 [20]. The

HTTP requests consisted of the following for each top domain (i.e., the target server):

- An HTTP request issued directly from our client
- An HTTP request issued through the proxy
- An HTTPS request issued directly from our client
- An HTTPS request issued through the proxy

We issued each request using the command-line program `curl` [21], all from a single vantage point within the United States. We make several observations about the results in the paragraphs that follow.

The proxy received and handled both HTTP and HTTPS requests. For each HTTP or HTTPS request, our client was able to establish a TCP connection with the proxy at the designated port and make the appropriate request (as documented in Section II-B). For each such request, it received an HTTP response. Thus, as far as we could observe, both HTTP and HTTPS requests were handled by the proxy.

Where the proxy received an HTTP response, the request was returned unmodified. We compared the content returned with the response received in conjunction with an HTTP request made directly with that received from a proxied request. The only differences we observed in content appeared to be related to the fact that one request came from the proxy's IP address while the other came from the client's IP address. For example, for some sites, the URLs for a given embedded image differed slightly—indicative of two different versions of the page, served either deliberately because of differing request origin or because upstream changes had not propagated to both pages. Other sites explicitly included the IP address of the client in the page content, and the IP addresses were obviously different from the target server's perspective. While our analysis does not mean that the proxy might tamper with some content, we could find no evidence of such tampering with our limited testing.

Where an HTTP request was made, and the proxy did not receive an HTTP response, it returned an artificially-generated HTTP response. When a target server exhibited either of the following conditions, the proxy created and returned its own, artificial HTTP response:

- The domain name of the target server did not resolve to an IP address (i.e., resulted in a a NODATA or `NXDOMAIN` response). One example was `microsoftonline.com`; although subdomains of `microsoftonline.com` resolve to an IP address, `microsoftonline.com` itself does not.
- The TCP connection to the target server times out, or is refused (i.e., with a TCP RST). For example, `163.com` timed out, and the connection to `godaddy.com` port 80 was refused.

In the case that the proxy generated a response—due to either of these conditions—the response code was 200, and the entire body consisted of the following content:

```
<html><meta http-equiv="refresh"
  content="0;url=http://proxy.domain.name">
  </html>
```

This has the effect of redirecting the client to the URL `http://proxy.domain.name/`. At the time of analysis, this URL further redirected the client to `https://net.domain.name`. The Web page at `https://net.domain.name` included just three major links: "Web hosting", "Create Website", and "Email Account". Each link directs the user to a list of ads related to the description of the respective link. This behavior is the HTTP equivalent of Verisign's Site Finder [22], with which a wildcard record was introduced into the `com` and `net` zones in 2003. With the Site Finder wildcard records in place, otherwise nonexistent domain names within the `com` and `net` domains resolved to IP addresses, which listened for and responded to several services, including HTTP and SMTP.

Where an HTTPS request was made, the proxy made no attempt to tamper with the TLS connection or return artificial content. We observed no TLS warnings related to invalid or self-signed certificates—except in the few cases where the certificates were actually self-signed, as observed by direct HTTPS requests. Thus, there was no evidence of the proxy attempting to impersonate the target server when HTTPS was in use. Even in the case where an HTTP response would be created by the proxy for a given domain name (i.e., nonexistent domain, connection timeout, or connection refused), the HTTPS equivalent request (i.e., a `CONNECT` request) would still result in an error, as opposed to the proxy returning artificial content. Thus, as long as HTTPS was attempted by the client, no attempt was made by the proxy to create responses.

We note that introducing additional parameters into our experimentation might have produced a more comprehensive analysis. For example, we might have issued the requests from different vantage points or used different User-Agent strings. However, even with our simple analysis, we were able to answer the most important questions associated with our research.

VII. Vulnerable Clients

From the query counts in the DNSDB entries, we can infer something about the number of queries for `wpad.domain.name` that were made during different phases. However, because those counts include no client information (i.e., IP addresses of recursive resolvers), we have no sense for the diversity of the queries. We now use data from additional sources to quantify the pervasiveness of clients potentially vulnerable to man-in-the-middle attack due to vulnerable network configuration.

The Day-in-the-Life (DITL) data set is an effort sponsored by the DNS Operations, Analysis, and Research Center (OARC) to capture 48 hours worth of queries at the DNS root servers on a yearly basis [23]. Using DITL, we analyzed DNS queries observed at the root from 2010 through 2022. We extracted the clients and query counts for all queries for `wpad.domain.name` each year for the following root servers: A, C, J, and K. We selected these letters from the 13

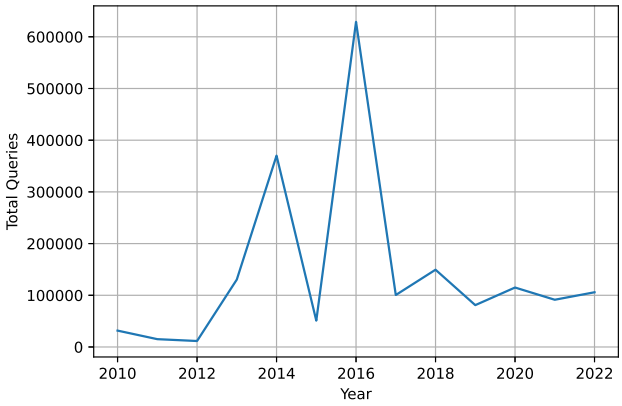


Fig. 1: The total number of DNS queries for `wpad.domain.name` observed during the DITL collection.

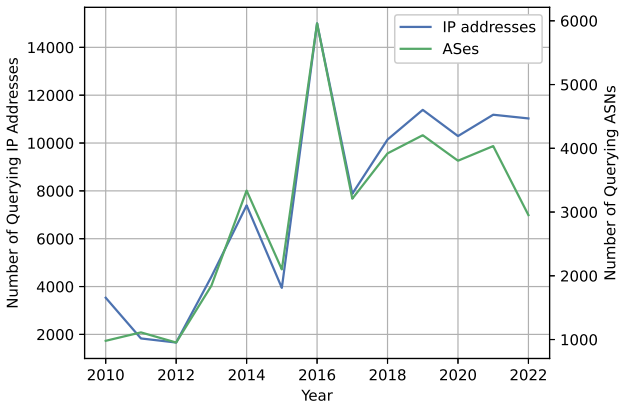


Fig. 2: The total number of unique IP addresses and ASes from which DNS queries for `wpad.domain.name` were observed during the DITL collection.

total letters because they were the only four root letters that contributed data in every DITL year we analyzed.

Queries observed at the root servers are likely to represent some fraction of the `wpad.domain.name` queries asked of recursive resolvers during the collection period. Not every such query will result in a query reaching a root server—or any authoritative server, for that matter—due to caching. Nonetheless, periodic cache misses in conjunction with `wpad.domain.name` queries cause a query to be directed to one of the root servers.

A plot of the total number of queries for `wpad.domain.name` observed at the root servers for each year is shown in Figure 1, and a plot of the IP addresses and ASes from which at least one query for `wpad.domain.name` was is shown in Figure 2. One of the major takeaways from Figure 1 is that the number of queries for `wpad.domain.name` has been nonzero since 2010 and has seen overall increase since then. As recently as 2022, over 100K DNS queries for `wpad.domain.name` were observed during the DITL collection period. That number

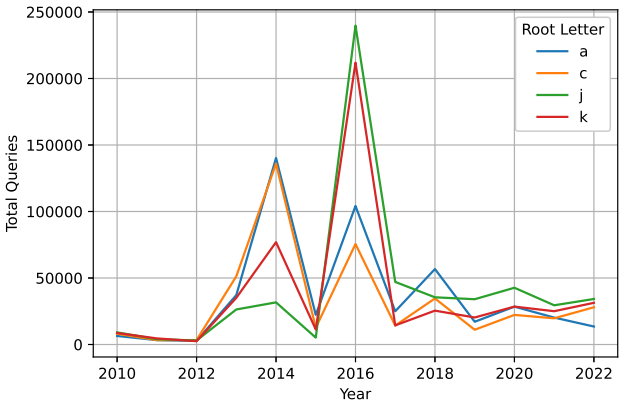


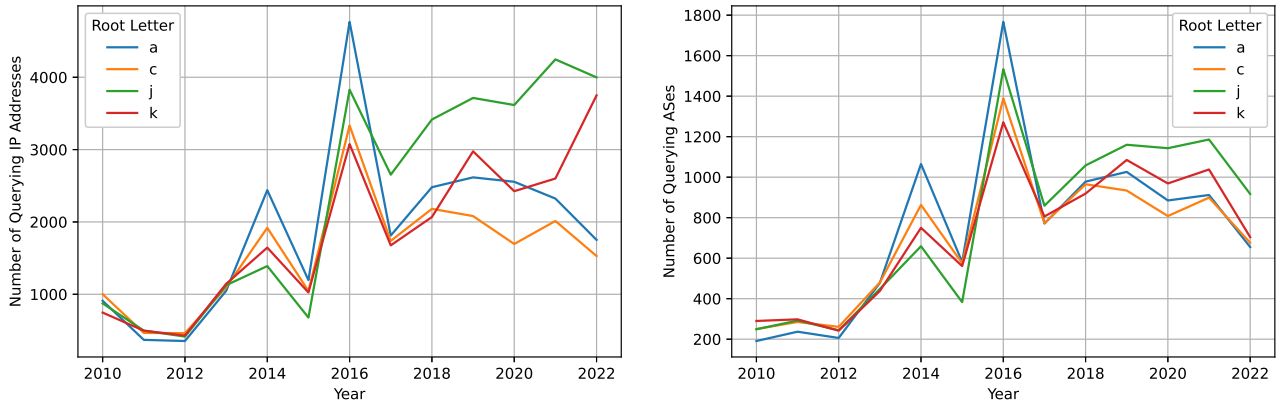
Fig. 3: The total number of DNS queries for `wpad.domain.name` observed during the DITL collection, by letter.

has fluctuated between 80K and 150K since 2017, with the variance decreasing each year. This indicates that there is still a recognizable presence of vulnerable routers being used today.

We also observe in Figure 2 that the numbers of IP addresses and ASes from which `wpad.domain.name` queries originated are higher in 2022 than they were in 2010, with over 11K IP addresses from nearly 3K ASes producing at least one query for `wpad.domain.name`. The number of querying IP addresses has remained relatively steady since 2018, with between 10K and 11K querying. The number of ASes from which queries have originated fluctuated between 2.3K and 4.2K during the years 2017 through 2021. These numbers indicate that queries for `wpad.domain.name` come from a relatively small but not insignificant set of IP addresses and ASes during each DITL collection period. Thus, WPAD clients are not completely isolated.

In 2022 the number of querying ASes dipped below 3K for the first time since 2015. While a decrease in affected end systems is one explanation for this, another consideration is QNAME (query name) minimization [24]. With QNAME minimization, only the right-most label of a given query name is visible at the root servers. Thus, a QNAME-minimizating resolver would send only `name` to the root servers instead of `wpad.domain.name`. While definitively determining the cause of the decrease in querying ASes between 2021 and 2022 is not possible with the data at hand, we can make some observations. First, previous work by Hilton et al. shows that the most significant increase in QNAME-minimizing resolvers came between 2018 and 2020, with no increase in 2021 [25]. Yet the `wpad.domain.name` query data shows no sustainable decrease in querying IP addresses and ASes between 2018 and 2020. Second, a more important fact is number of querying IP addresses and ASes from which `wpad.domain.name` *is* observed. With increasing adoption of QNAME minimization, that observation speaks to the significance of those numbers.

Another major observation in both plots is that both the



(a) (b)

Fig. 4: The total number of unique IP addresses (a) and ASes (b) from which DNS queries for `wpad.domain.name` were observed during the DITL collection, by letter.

overall query count and the number of IP addresses and ASes producing queries was significantly higher in the 2014 and 2016 DITL collections. To better understand this increase, we plotted the same data by root letter in Figure 3 and Figure 4. Those figures show that the increase in both total query count and number of querying IP addresses and ASes was relatively higher in all root letters. While we do not have enough data to tell us for the reason, our further analysis does tell us that the increase does not seem to be due to root letter bias.

Because the client IP addresses typically represent recursive DNS servers, we do not know how many clients—potentially vulnerable—are behind the recursive servers whose behaviors we have analyzed in this section, nor do we know if these queries are actually associated with the D-Link router or more generally with the vulnerability described herein. However, the reports in Section VI-B2 supplement our quantitative analysis with some anecdotal experiences confirming HTTP interception.

VIII. Delegation Removal

In an effort to mitigate the problems experienced with `wpad.domain.name`, Verisign—the registry operator for `name`—removed the delegation NS records for `wpad.domain.name` from the `name` zone in December 2021. Since that time, WHOIS shows the status of `wpad.domain.name` as "clientHold", which is "an uncommon status that is usually enacted during legal disputes, non-payment, or when your domain is subject to deletion." While this status cannot keep vulnerable clients from issuing queries for `wpad.domain.name`, it can keep them from being exploited because `wpad.domain.name` is not delegated and will not resolve to an IP address to which they might otherwise connect. DNSDB shows that NS records for `wpad.domain.name` were last observed on December, 9, 2021.

IX. Conclusion

In this paper we have described the circumstances whereby misconfigured firmware can create the opportunity for abuse of the WPAD protocol and have detailed the specifics of one instance that abuse: `wpad.domain.name`. We followed the delegation of `wpad.domain.name` through both its innocuous phase and its more intrusive phase, documenting DNS resolution, HTTP response for WPAD requests, and HTTP requests made through a designated HTTP proxy. We also used DNS queries at the root servers to look at trends associated with potentially affected clients over time. These trends include both an increased overall query count and an increased overall query diversity since 2010.

The circumstances discussed in this paper reinforce the principle that the potential for opportunistic exploit might go unnoticed unless and until triggered by an external event. In the case of `wpad.domain.name`, DNS queries from vulnerable clients were observed years before they were tampered with, opportunistically exploiting their vulnerability. The triggers in this case were the registration of `wpad.domain.name`, and the responses from the `wpad.domain.name` HTTP server directing Web clients to a third-party HTTP proxy for all subsequent HTTP requests. Had either one of these not happened, clients would be vulnerable to but not be negatively impacted by the router issue.

We hope that the specifics of this exploit, as well as the more general lessons learned, can contribute to a safer Internet.

Acknowledgments

We gratefully acknowledge ICANN for their support of the work that produced this material. We also thank Farsight Security and DomainTools for providing us access to DNSDB, DNS-OARC for providing us access to the DITL collections, and CAIDA for providing us access to the Ark infrastructure to carry out our measurements. Finally, we thank our shepherd and the anonymous reviewers for their comments and guidance.

References

[1] P. Gauthier, J. Cohen, M. Dunsmuir, and C. Perkins, "Web proxy auto-discovery protocol," July 1999. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-wrec-wpad-01

[2] Netscape, "Navigator proxy auto-config file format," March 1996. [Online]. Available: https://web.archive.org/web/20060424005037/wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html

[3] Microsoft, "Ipv6 extensions to navigator auto-config file format," January 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/winhttp/ipv6-extensions-to-navigator-auto-config-file-format

[4] D. Wessels and M. Fomenkov, "Wow, that's a lot of packets." in *Passive and Active Network Measurement Workshop (PAM)*, 2003.

[5] Reuters, "How to hack china for just $1,800." [Online]. Available: https://www.reuters.com/article/urnidgns852573c40069388000257671006e391b/how-to-hack-china-for-just-1800-idUS3771901620091118

[6] Q. A. Chen, E. Osterweil, M. Thomas, and Z. M. Mao, "Mitm attack by name collision: Cause analysis and vulnerability assessment in the new gtld era," in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 675–690.

[7] ICANN, "New generic top-level domains." [Online]. Available: https://newgtlds.icann.org/en

[8] Cybersecurity and I. S. Agency, "WPAD Name Collision Vulnerability," 2016. [Online]. Available: https://www.cisa.gov/news-events/alerts/2016/05/23/wpad-name-collision-vulnerability

[9] Q. A. Chen, M. Thomas, E. Osterweil, Y. Cao, J. You, and Z. M. Mao, "Client-side name collision vulnerability in the new gtld era: A systematic study," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 941–956.

[10] J. Kristoff, M. Ghasemisharif, C. Kanich, and J. Polakis, "Plight at the end of the tunnel," in *Passive and Active Measurement*. Cham: Springer International Publishing, 2021, pp. 390–405.

[11] E. Boulila and M. Dacier, "Wpad: Waiting patiently for an announced disaster," *ACM Comput. Surv.*, vol. 55, no. 10, feb 2023.

[12] M. Kulberg, "In-depth Technical Analysis of WPAD," pp. 14 – 15, 2018. [Online]. Available: https://www.cybersecobservatory.com/wp-content/uploads/2018/04/spring-2018-state-of-the-internet-security-report.pdf

[13] D. Persch, "In-the-Wild WPAD Attack — How Threat Actors Abused Flawed Protocol For Years." [Online]. Available: https://www.sentinelone.com/blog/in-the-wild-wpad-attack-how-threat-actors-abused-flawed-protocol-for-years/

[14] D-Link, "Wireless n 300 router: User manual," July 2017. [Online]. Available: http://legacyfiles.us.dlink.com/DIR-615/REVT/DIR-615_T3_Manual_v1.10(DI).pdf

[15] F. Security, "Passive dns historical internet database: Farsight dnsdb," 2022. [Online]. Available: https://www.farsightsecurity.com/solutions/dnsdb/

[16] M. Bailey, "How to set up dns records on gandi.net to use a custom domain on github pages," 2020. [Online]. Available: https://gist.github.com/matt-bailey/bbbc181d5234c618e4dfe0642ad80297

[17] CAIDA, "Archipelago." [Online]. Available: http://www.caida.org/projects/ark/

[18] I. Archive, "Wayback machine." [Online]. Available: https://archive.org/

[19] Unknown, "Contents of http://wpad.domain.name/wpad.dat on wayback machine." [Online]. Available: https://web.archive.org/web/20210701000000*/http://wpad.domain.name/wpad.dat

[20] Amazon, "Alexa top sites." [Online]. Available: https://aws.amazon.com/alexa-top-sites/

[21] curl team, "curl." [Online]. Available: https://curl.se/

[22] I. VeriSign, "Verisign's site finder implementation," August 2003. [Online]. Available: https://web.archive.org/web/20041109202247/http://www.verisign.com/static/002702.pdf

[23] Domain Name System Operation, Analysis, and Research Center, "Ditl traces and analysis," 2022. [Online]. Available: https://www.dns-oarc.net/oarc/data/ditl

[24] S. Bortzmeyer, R. Dolmans, and P. Hoffman, "RFC 9156: DNS Query Name Minimisation to Improve Privacy," November 2021.

[25] A. Hilton, C. Deccio, and J. Davis, "Fourteen years in the life: A root server's perspective on dns resolver security," in *32nd USENIX Security Symposium (USENIX Seucurity '23)*, 2023.