WPAD: Waiting Patiently for an Announced Disaster

ELYSSA BOULILA, EURECOM, France
MARC DACIER, KAUST, Saudi Arabia

The Web Proxy Auto-Discovery protocol (WPAD[1]) is widely used despite being flawed. Its purpose is to enable a client machine to autonomously identify an appropriate proxy, if any, to connect to. This can be useful in corporate networks, for example. Its vulnerabilities range from enabling an attacker to execute code remotely on client machines, to carry out SSL MITM attacks, to subvert Windows NTLM authentication or even to steal Google authentication tokens. Several publications, talks and blog posts have tried to raise awareness about some of these security issues. 23 distinct CVEs have been published. Nevertheless, WPAD runs by default on Windows machines and most users are unaware of its existence. Our goal is to offer within a single publication a survey of all the known vulnerabilities surrounding WPAD, a presentation of some novel threats related to this protocol, as well as a description of mitigation and detection techniques to prevent the exploitation of its vulnerabilities. We hope that this publication will be an eye opener for all those concerned with the security of their networks and that the offered mitigation techniques will help them to deal with the numerous threats that WPAD brings to their environments.

CCS Concepts: • **Security and privacy → Web protocol security**; Browser security.

Additional Key Words and Phrases: WPAD security, web security, service discovery, proxy security, PAC file, javascript security, NTLM, HTTPS, MITM, SSL, remote code execution

ACM Reference Format:
Elyssa Boulila and Marc Dacier. 2022. WPAD: Waiting Patiently for an Announced Disaster. 1, 1 (October 2022), 28 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Security issues surrounding the WPAD protocol have been discussed for more than 20 years. While old ones are still threatening us, new ones are regularly discovered. Researchers have discussed these problems in various venues over the years, one problem at a time. This paper aims at offering the big picture, bringing all the problems together and more. Our hope is that those reading this survey will realise the seriousness of the dangers created by the WPAD protocol and that the prescribed mitigation techniques will be of help to them as well.

The WPAD (Web Proxy Auto-Discovery Protocol) is a protocol that is used by web clients to locate a proxy, if any, to connect to in order to reach the Internet. For those discovering what this protocol is about, let us insist on the fact that this protocol is most likely used by the machine you are using today. This makes you vulnerable to a large collection of attacks, described in this paper. Why this protocol has not, so far, been used in more high profile attacks than the ones mentioned here after remains unclear to us. However, we should not wait for a disaster to occur before taking the appropriate actions.

[1]All the acronyms can be found in Appendix A

Authors' addresses: Elyssa Boulila, elyssa.boulila@eurecom.fr, EURECOM, Sophia Antipolis, France; Marc Dacier, marc.dacier@kaust.edu.sa, KAUST, Tuwal, Saudi Arabia.

WPAD is documented in an Internet-Draft which expired in 1999 [26]. Nonetheless, today, it is supported by major browsers (Edge, IE, Firefox, Chrome, Safari) and operating systems (Windows, Linux) [36]. It runs by default on Windows10 machines.

The WPAD client searches for a WPAD server using a resource discovery protocol. If found, it contacts it to obtain a configuration file, referred to as a PAC file.

Concerns about the security of WPAD have existed since 1999 [52, 59]. This paper offers a survey on this protocol, not only by summarizing all its known vulnerabilities but also by explaining how previous malicious actors have taken advantage of them in the past. This paper presents more than just a systematic review of WPAD security. It also reports on some new findings and results of experiments and investigations that we have carried out on it. We also discuss possible mitigation and detection techniques.

The main contributions of this paper are:

- A systematic review of past publications related to WPAD, its security and vulnerabilities;
- An investigation into some novel attack scenarios involving WPAD;
- An introduction to the main known attacks that have exploited WPAD;
- An investigation into today's reality of the threat;
- A discussion of the possible mitigation techniques;

To discuss these topics, the paper is structured as follows. Section 2 offers an introduction to WPAD as it is defined in the Internet draft [26] , which we compare to what we observe when using various real world implementations. Section 3 details the proposed methodology to carry out a systematic review of all the known vulnerabilities related to WPAD, discusses them and presents how this protocol has been exploited in the past by malicious actors. We also introduce in that Section novel attack mechanisms that we have discovered and disclosed in a responsible way. Section 4 surveys, through various means, the reality of the threat WPAD represents today. Section 5 surveys the existing mitigation techniques, including the detection ones and discusses their limitations. Section 6 concludes the paper.

2 WPAD OVERVIEW

2.1 The use of WPAD in corporate networks

The Web Proxy Auto-Discovery (WPAD) protocol is a network protocol mostly used in corporate networks. Its goal is to enable machines to connect to a proxy before accessing the Internet. This is a convenient approach if, for instance, a company requires its client machines to use a Web Application Firewall (WAF) before accessing the Internet. WPAD is used for efficiency reasons: by implementing this protocol, machines do not have to be individually configured to connect to a particular proxy. Instead, WPAD provides the appropriate configurations to each machine upon request. The subsequent Sections explain how WPAD works in theory and in practice.

2.2 The theoretical definition

The WPAD protocol definition has never reached the status of an RFC. It has only been proposed as an Internet-Draft [26] which expired in December 1999. This document remains the only agreed upon definition of the protocol and this is why, for the sake of completeness, we present it here after.

Gauthier et al. define the objectives of the WPAD protocol as follows [26]:

- *The [...] web client software needs to be configured with the URL of a proxy auto-configuration file or script. [...] [the WPAD protocol] does not attempt to discuss the contents of these files*
- *The Web Proxy Auto-Discovery (WPAD) problem reduces to providing the web client a mechanism for discovering the URL of the Configuration File [...] (CURL) [...]*

- *the goal of the WPAD process is to discover the correct CURL at which to retrieve the CFILE. The client is *not* trying to directly discover the name of the proxy server. [...]*
- *different clients requesting the CURL may receive completely different CFILEs in response [...] based on a number of criteria such as the "User-Agent" header, "Accept" headers, client IP address/subnet, etc., [...]*
- *[26] [discusses] a range of mechanisms for discovering the [... CURL ...]. The client will attempt them in a predefined order, until one succeeds.*

The Internet draft lists three specific mechanisms to obtain the mentioned CURL, namely: DHCP, SLP and DNS:

DHCP protocol : A DHCP server can offer the CURL as the value associated with the option 252. This usually takes place when a client machine joins the network and contacts the DHCP server for the first time to obtain its IP, etc.,. A client machine can also obtain that information from a server after having issued a DHCPINFORM request. It is worth noting that the option 252 is not listed as one of the valid DHCP options defined in the RFC 2132 [4].

SLP protocol : SLP stands for Service Location Protocol and was defined in 1999 in the RFC 2608 [32]. It enables network applications to discover the location and the configuration of network services by issuing multicast requests asking for, e.g., the WPAD service, and receiving back a unicast reply whose syntax adheres to the "service:" URL scheme name defined in RFC 2609 [31]. An example of a valid answer for the WPAD service could look like service:wpad:http://server:80/dir/filename out of which the CURL is then extracted.

DNS protocol : The client machine extracts its network name by removing its hostname from its own FQDN. It uses that network name to make an SRV request looking for machines offering the WPAD service in that network. If no answer is provided, the same algorithm is used with TXT requests, looking for a "service:wpad" URL scheme name in the responses. If the client still has no success, it makes an A request by prepending the hostname "wpad" to the network name. If none of these requests (SRV, TXT, A) provides a server name, the client repeats this process after having removed the top left component of the network name, until the TLD is reached. This mechanism is called *DNS devolution*. If one of these mechanisms returns a valid hostname for the WPAD server then the CURL is constructed as follows: http://hostname:80/wpad.dat

The configuration file (PAC file) obtained from one of the previous mechanisms is a Javascript file. It should contain a specific function called FindProxyForUrl which, when executed, enables to find an appropriate proxy (if any) corresponding to each fetched URL. The set of allowed Javascript instructions in this file is limited (for example dnsDomainIs, shExpMatch etc.,) [36].

Additionally, WPAD should be supported at the client's level (for example at the browser's level) since the header of each packet changes depending if a proxy is used or not.

2.3 The observed reality

The WPAD protocol has never reached the status of an RFC and its Internet draft expired at the end of 1999. However, it is supported and used by all common operating systems and browsers. It is even enabled by default on Windows machines.

We tested its implementation on different operating systems and browsers. In our tests, we have used Windows 10 Home Edition (version 20H2) and Linux (Ubuntu 20.04). We also have tested how the following browsers would behave to WPAD input: Mozilla Firefox (Version 89.0.2 in Windows, and Version 90.0 in Linux), Google Chrome (Version 91.0.4472.77), Edge (Version 91.0.864.67)[2].

[2]WPAD is also supported on Android, MacOS, and iOS systems

Its implementation in reality differs from the one presented in [26]. We use Algorithm 1 on page 6 to explain the various steps we have observed machines go through when running WPAD. We highlight the notable differences between that observed behavior and the expected theoretical one. The main steps of the observed algorithm are 1:

(1) First, the machine tries to access the value, if any, of the DHCP option 252.
(2) Second, the machine tries to download the PAC file from the CURL, to read it, and to check if it is valid. If all is good, the Windows machine stores this CURL in the registry key: Computer\HKEY_USERS\S-1-5-19\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\WpadDetectedUrl. In this case, the discovery process ends here (status = SUCCESS).
(3) If the option 252 is empty, then the Windows machine will check the registry looking for a previously stored CURL. If found, it will check the validity of the downloaded PAC file. If all is good, the discovery process ends here (status = SUCCESS).
(4) If the DHCP discovery mechanism fails (either because the downloaded PAC file is not valid or because there is no stored WPAD URL in the registry), the machine moves to the next discovery protocol: DNS. It stores the FQDN, the DHCP provided domain names (DHCP option value of "domain name" [4]) and the search domains (DHCP option value of "domain search" [1]) in a new list which we call, for the sake of clearness, "local_domains". If *local_domains* is not empty, the machine prepends "wpad." to each domain and makes DNS A and AAAA lookups for each domain. If a lookup succeeds, the machine tries to download a file called "wpad.dat" from the found wpad server. If the obtained file is valid, the CURL is stored in the registry and the process ends with success (status = SUCCESS).
(5) If the *local_domains* list is empty or if no valid PAC file is found, the client may use some zeroconf protocols [95] (such as NBNS [39], LLMNR [2] and mDNS [91]) in order to locate a local WPAD server by trying to resolve a "wpad.local" FQDN. If such a server is found, the process ends with success if the downloaded PAC file is valid (status = SUCCESS).
(6) If no valid PAC file is found from any of the previously described mechanisms then the discovery ends with failure (status = FAIL).

We now explain the main differences between the observed and the theoretical algorithm. We highlight them in its presentation with small numbers contained within circles:

① Windows machines store the obtained WPAD CURL in a registry value. Since several web clients rely on the system's settings for obtaining proxy settings, these clients use the WPAD registry value. The CURL obtained in a given, possibly untrusted, network could thus be stored and used later on by the machine when connected in another, supposedly secure, network.
② None of the operating systems and browsers that we have tested did support SLP for discovering the WPAD CURL anymore.
③ The network name used to make the WPAD DNS lookups (wpad.network) is not only extracted from the machine's hostname but also from DHCP provided domains (for example from the "domain-name" or the "domain-search" DHCP options).
④ In 2009, Microsoft released a security advisory related to DNS devolution [55]. Some domains may get leaked outside of the organizational boundary and get answered by malicious users. The advisory announced the following changes: *i)* The DNS devolution is not supported if the suffix is extracted from the machine's DNS suffix search list; *ii)* The machine may use the devolution if it uses a primary DNS suffix; *iii)* The devolution is not used if the domain consists of only two labels (domain.TLD)
During our experiments, we have not observed any DNS devolution being carried out, be it by the OS or by the browsers.
⑤ No WPAD related TXT or SRV lookups were observed when launching the Windows 10 machine.

⑥ For some browsers, such as Mozilla Firefox, AAAA lookups were observed for WPAD domains.
⑦ While not being mentioned at all in the draft, client machines are seen sending MDNS requests for "wpad.local" if no local domain name is provided for the machine. Some browsers also rely on Zeroconf protocols [95] such as MDNS [91] LLMNR [2] or NBNS [39] to discover a WPAD server. Microsoft released a security update related to local WPAD resolution [58]. By default, WPAD resolution for auto proxy detection will not use NBNS.
⑧ The whole Javascript code contained in the PAC file is executed when checking its validity (not only the FindProxyForUrl function). Even though the set of allowed accessible Javascript functions is limited, this enables an attacker to put malicious code that will be executed by the machine. Section 3.6.5 details how such attacks are possible.

As opposed to Windows, Linux is configured by default to not use WPAD: the machine ignores the 252 DHCP option value and does not search for a WPAD server. If a browser running on this OS is configured to use WPAD, it may rely on DNS or MDNS to find a WPAD server.

3 TECHNICAL VULNERABILITIES RELATED TO WPAD

To cover all the reported vulnerabilities related to the Web Proxy Auto-Discovery protocol, we have carried out a structured, exhaustive, search using the so-called Preferred Reporting Items for Systematic Reviews and Meta-Analysis (PRISMA) approach [82].

First, the study rationales are presented. Second, the research questions are defined and the search phase is explained. Finally, the realization of the search is discussed.

Each of these steps is described in the subsequent sections.

3.1 Study rationales

As alluded to in the introduction, specific publications detailing one security problem or another related to WPAD have flourished over the years, for more than 20 years. Some problems have been fixed but the most severe ones are still threatening us. Despite being used by a very large number of people, very few are in fact aware of the role of the WPAD protocol and even less of the many security issues it brings with it. We feel strongly that it is important to highlight the threats this protocol represents. We want the practitioners to find here all the information they need to make a conscious decision to keep using this protocol, or not. We want the researchers to find here all the information they need to identify new possible research directions to secure this protocol, or to mitigate the threats it represents, or to identify new vulnerabilities against it. We want the students learning about network security to find here all the information they need to realize the importance of sound and secure network protocol design by looking at WPAD as a perfect counterexample of this.

3.2 Definition of research questions (RQ)

The research questions answered in the following Subsections are:

- **RQ-1** *What is the distribution per year and publication venues of published papers related to Web Proxy Auto-discovery Protocol?*
- **RQ-2** *What are the main security concerns surrounding WPAD?*
- **RQ-3** *How was WPAD exploited in the past by malicious actors?*

3.3 Search phase

The search for relevant publications has been carried out in 4 phases: the identification, screening, eligibility and selection phases.

Algorithm 1 WPAD's generic implementation

```
▹ Get DHCP option 252
url = get_value_of(DHCP _option252)
if url ≠ null then
    ▹ if DHCP provided option 252
    if is_valid(read_PAC _File())⑧ then
        store_url_in_registry_key(url,"WpadDetectedUrl")
        return SUCCESS ▹ valid PAC file
    end if
else
    ▹ Check if there is a WPAD URL stored in the registry
    url = get_value_from_registry("WpadDetectedUrl") ①
    if url ≠ null then
        if is_valid(read_PAC _File())⑧ then
            return SUCCESS ▹ valid PAC file
        end if
    end if
end if
▹ We have not observed SLP requests of WPAD ②
▹ Start the DNS discovery mechanism
local_domains= get_domains( FQDN, search_list, dhcp_domain_name) ③
if local_domains ≠ null then
    for local_domain_name in local_domains do
        ▹ This For loop was part of a bigger While loop in the draft's algorithm. The latter loop was
        responsible for the DNS devolution mechanism. We have not observed it in our experiments ④
        if local_domain_name ≠ null then
            ▹ Only make A records for WPAD ( No SRV or TXT lookups) ⑤
            dns_record = dns_query (QNAME = wpad.local_domain_name. , QTYPE = A)
            dns_record = dns_query (QNAME = wpad.local_domain_name. , QTYPE = AAAA)
            ▹ check A and AAAA records ⑥
            if is_valid(read_PAC _File())⑧ then
                store_url_in_registry_key(url,"WpadDetectedUrl")
                return SUCCESS ▹ valid PAC file
            end if
        end if
    end for
end if
▹ DNS failed
▹ resolve WPAD webserver to IP address on a local network⑦
▹ Try MDNS resolution
mdns_resolv(QNAME =wpad.local.)
▹ Try NBNS resolution
nbns_resolv(QNAME =WPAD.)
▹ Try LLMNR resolution
llmnr_resolv(QNAME =wpad.)
if local_resolution_succeeded() then
    if is_valid(read_PAC _File())⑧ then
        return SUCCESS ▹ valid PAC file
    end if
end if
▹ All the discovery mechanisms failed
return FAIL ▹ could not locate valid PAC file
```

Table 1. Search sources.

Source	type	URL
ScienceDirect	Digital library	https://www.sciencedirect.com/
Scopus	Search engine	https://www.scopus.com/
IEEEXplore	Digital library	https://ieeexplore.ieee.org/
ACM digital library	Digital library	https://dl.acm.org/
Google Scholar	Search engine	https://scholar.google.com/

(1) In the identification phase, we define reliable information sources and we extract the publications related to WPAD based on specific queries.
(2) In the screening phase, we only keep non duplicated English records.
(3) In the eligibility phase, we define some inclusion and exclusion criteria to retain only the papers relevant to our study. This selection phase is constructed in 2 steps:
   - The first step selects records by looking at their title and abstract.
   - The second step makes a full assessment of the text.
(4) The final phase is the selection phase. It keeps only the papers, starting with the oldest ones, that add new contributions to the study.

The next subsections provide more details about each of these phases.

3.3.1 Identification phase: The first step consists in identifying the information sources. Table 1 lists the various digital libraries and search engines used in our study. Publications were extracted from these databases using key search terms and their possible combination. Key search terms include: *wpad*, *security*, *vulnerability*, *web proxy auto-discovery*, *proxy*, *dhcp*, *dns*, *mdns*, *llmnr*, *nbns*, *slp*, *malicious pac*. The queries built to find and collect these results are the following:

- **S-1**: dhcp AND wpad AND (security OR vulnerability)
- **S-2**: dns AND wpad AND (security OR vulnerability)
- **S-3**: wpad AND (mdns OR llmnr OR nbns) AND security
- **S-4**: slp AND wpad AND (security OR vulnerability)
- **S-5**: "malicious pac" AND proxy AND "web proxy auto-discovery"

Additionally, we screened the archives of some major hacking conferences (Black Hat Briefings, DEF CON, HITB Security Conference). We have also looked into the bibliography section of the relevant publications obtained at the end of our search. Moreover, in our own investigation of PAC files, we came across some interesting research blog posts that investigate some WPAD issues. Last but not least, we have also considered the papers that were referencing the published papers we had selected.

3.3.2 Screening phase: Duplicated papers were excluded. Additionally, we only considered studies written in English.

3.3.3 Eligibility phase: Specific eligibility criteria were defined to shortlist the research articles to be included in this systematic review. To refine the results found, we use the following inclusion and exclusion criteria:

- The studies' title, abstract, and full text should be security related.
- Studies should discuss security issues related to WPAD or to the usage of PAC files. Papers that identify how malicious actors have exploited this protocol in the past are also kept.

To explain the necessity of these criteria, we describe here after some examples of papers matching our search queries but later excluded from our study:

- One of the results obtained from S-1 is the paper entitled "WiFiProfiler: Cooperative Diagnosis in Wireless LANS" [15] by Chandra et al. [15]. After assessing its title and its abstract, it appears that this paper is not related to security.
- One of the results obtained from S-2 is the paper entitled "Detecting mass-mailing worm infected hosts by mining DNS traffic data" [40] by Ishibashi et al. [40]. After an assessment of its full text, we decided not to keep it because it only presents WPAD as a domain highly queried by infected machines. But no conclusion or any relevant information was drawn from that about the WPAD protocol itself.

3.3.4 Selection phase: We have read all the papers obtained from the previous phases and have selected, starting from the oldest ones, those that we consider to add significant information to our topic of interest. The selected papers should contain elaborate and clear explanations of the risks regarding the use of WPAD. They should also contribute to the answers to at least one research question.

3.4 The realization of the systematic review

Figure 1 offers a synthetic representation of the results of our search flow.

In the identification phase, 525 records were obtained from the electronic databases and search engines listed in Table 1. We provide the number of results per query and per source in Table 2. In the screening phase, we only keep non duplicate records written in English. This resulted in keeping 209 records.

Subsequently, the eligibility criteria are verified first based on the title and the abstract and second on the full text. 43 articles were eligible for our systematic review. In the reviewing phase, we select papers based on the criteria defined in the above subsection. From the 43 articles obtained from the eligibility phase, we only keep 17 papers.

Moreover, we look into the articles that cite the papers found above. We collect a total number of 313 papers: 1 from [3], 2 from [7], 20 from [17], 5 from [18], 36 from [19], 0 from [24], 11 from [30], 0 from [33], 177 from [37], 2 from [43], 1 from [46], 1 from [47], 0 from [49], 2 from [84], 8 from [87], 2 from [102], and 45 from [104]. After removing the duplicates and keeping only papers written in English, we obtain 287 results. Among these results only 3 papers [19][37] [18] were also obtained from the previous process. The majority of the papers found from looking at the citations of the 17 papers are security-related but do not discuss WPAD in any way. We select the ones that correspond to the eligibility criteria defined in Subsection 3.3.3. This process results in adding 2 more papers.

On top of screening the databases introduced in Table 1, we also screened the archives of the major hacking conferences: Black Hat Briefings (4 results added), DEF CON (2 results added), HITB Security Conference (1 result added). In total 7 results were added from this process to our systematic review.

We also looked manually for some additional related work by screening the references of the publications retrieved from the results of the two processes explained above. This enabled us to add 8 other records.

4 additional articles were found during our investigation of WPAD and 4 other publications are added to our survey from studying the topics discussed in the articles found. 40 papers are kept from this process to be analyzed in this study. We also look into the CVEs reported for WPAD issues. We have found 23 CVEs reports that had the keyword WPAD in them.

In total, we end up selecting 42 papers.

Table 2. Number of results found per query and per source in the identification phase.

	S1	S2	S3	S4	S5	Number of duplicates per source
ScienceDirect	22	28	0	0	0	22 over 50
Scopus	2	7	0	0	1	3 over 10
IEEEXplore	0	1	0	0	0	0 over 1
ACM digital library	4	15	3	0	1	9 over 23
Google Scholar	154	230	36	17	4	194 over 441

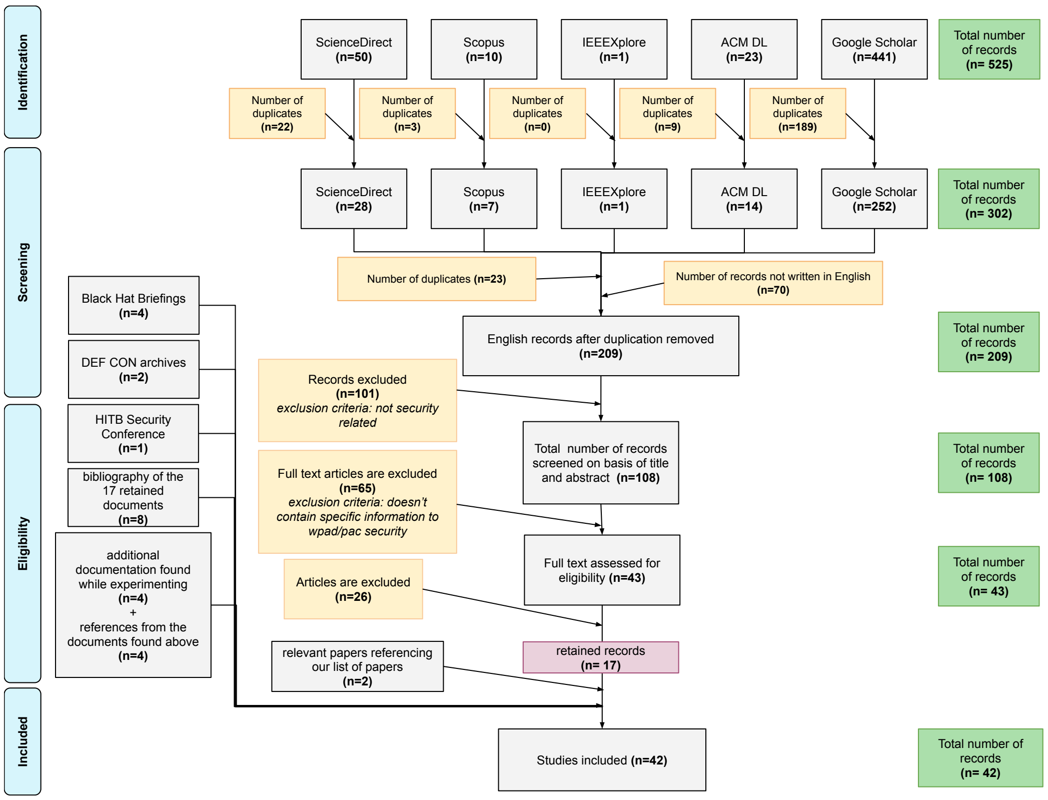
Identification
Screening
Eligibility
Included
ScienceDirect (n=50)
Scopus (n=10)
IEEEXplore (n=1)
ACM DL (n=23)
Google Scholar (n=441)
Total number of records (n= 525)
Number of duplicates (n=22)
Number of duplicates (n=3)
Number of duplicates (n=0)
Number of duplicates (n=9)
Number of duplicates (n=189)
ScienceDirect (n=28)
Scopus (n=7)
IEEEXplore (n=1)
ACM DL (n=14)
Google Scholar (n=252)
Total number of records (n= 302)
Number of duplicates (n=23)
Number of records not written in English (n=70)
English records after duplication removed (n=209)
Total number of records (n= 209)
Black Hat Briefings (n=4)
DEF CON archives (n=2)
HITB Security Conference (n=1)
bibliography of the 17 retained documents (n=8)
additional documentation found while experimenting (n=4) + references from the documents found above (n=4)
Records excluded (n=101) exclusion criteria: not security related
Total number of records screened on basis of title and abstract (n=108)
Total number of records (n= 108)
Full text articles are excluded (n=65) exclusion criteria: doesn't contain specific information to wpad/pac security
Full text assessed for eligibility (n=43)
Total number of records (n= 43)
Articles are excluded (n=26)
relevant papers referencing our list of papers (n=2)
retained records (n= 17)
Studies included (n=42)
Total number of records (n= 42)

Fig. 1. A PRISMA flow of this systematic review.

- 17 publications obtained from the databases introduced in Table 1: [3, 7, 17–19, 24, 30, 33, 37, 43, 46, 47, 49, 84, 87, 102, 104]
- 2 references which cite the documents found above [86, 99]
- 4 presentations from the Black Hat Briefings: [29, 35, 44, 101]
- 2 talks from the DEF CON archives: [5, 16]
- 1 talk from the HITB Security Conference: [100]
- 8 more papers found in the bibliography of the previously found publications [6, 10, 22, 25, 28, 93, 97, 103]
- 4 documents found while trying to understand the results of our experiments [8, 14, 83, 105]
- 4 more references found within the lastly found documents [21, 41, 58, 98]

- 23 CVEs reports: [59–81]

The subsequent sections present the results obtained from the systematic review described above and analyze the 65 obtained documents (42 publications and 23 CVEs). We will now explain the security issues related to WPAD by answering the research questions introduced in Subsection 3.2.

3.5 Answer to research question RQ-1:

What is the distribution per year and publication venues of published papers related to Web Proxy Auto-discovery Protocol?

Figure 2 depicts the distribution of the reported CVEs and the selected papers by year of publication. There are 2 noticeable peaks for papers in this Figure, in 2013 and 2016. The first one, in 2013, follows the discovery of the use of WPAD by several malicious actors [6, 30, 47, 104]. We will further discuss this point in Subsection 3.7 (page 18). The second peak in 2016 corresponds to the white hats' growing interest in the vulnerabilities related to WPAD. During this year, the subject was presented in several major hacking conferences.

Figure 3 depicts the distribution of selected papers by publisher. For Black Hat, the works we kept are not only from the Black Hat Briefings but also from the Trainings, etc.

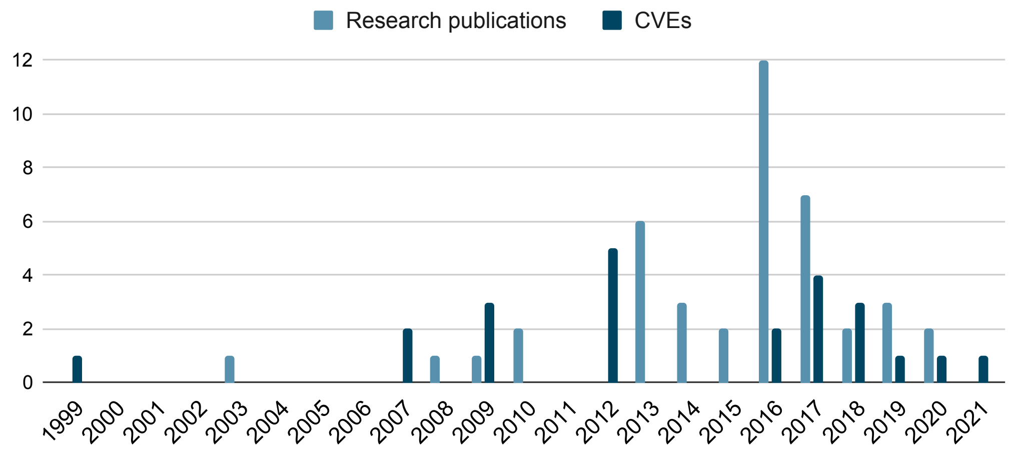

Fig. 2. Distribution of selected papers by year.

3.6 Answer to research question RQ-2:

What are the main security concerns surrounding WPAD?

In the following section, a thorough presentation of the issues surrounding WPAD is provided by means of several categories:

- Seminal work
- Impersonating a WPAD server
- NTLM relay
- SSL man-in-the-middle attacks
- Executing malicious Javascript code
- Argument-injecting attack
- Other recent concerns

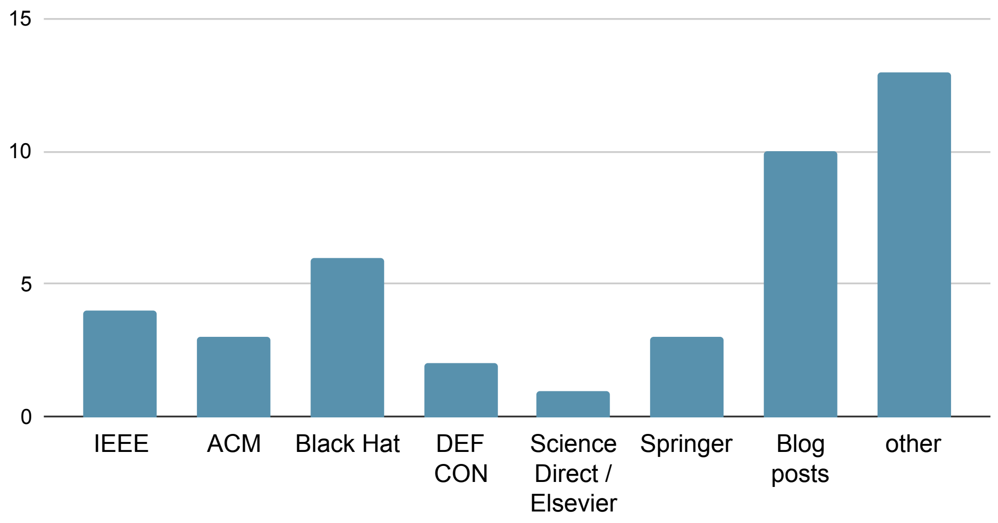

Fig. 3. Distribution of selected papers by publisher.

- DHCP attack revisited
- Reboot-resistant attack due to Windows implementation choices
- WPAD keeps running despite being explicitly disabled on Windows.

The last three subsections correspond to novel attacks that we have found while working on this survey. Whereas this publication does not aim at being an offensive paper, we felt it was relevant to include them here, for the sake of completeness.

It is important to note that we followed a responsible disclosure process and reached out in June 2021 and August 2021 to the Microsoft Security Response Center (MSRC) to report the newly found vulnerabilities. 3 cases were open:

(1) MSRC Case 65533: the DHCP attack scenario explained in Subsection 3.6.8 is addressed in this vulnerability report. The reply we received is "[...] We determined your finding as valid but does not meet our bar for a security update release. In considering this vulnerability we noted that WPAD is not a secure protocol, and also that this vulnerability is known: WPAD Name Collision Vulnerability [...]".
(2) MSRC Case 65588: the Windows persistence problem explained in Subsection 3.6.9 is addressed in this vulnerability report. The reply we received is: "[...] We determined your finding does not meet our bar for immediate servicing. [...] However, we've marked your finding for future review as an opportunity to improve our products. [...]"
(3) MSRC Case 53683: The issue with not being able to completely disable WPAD on Windows machines, as explained in Subsection 3.6.10, is addressed in this vulnerability report. The reply we received is: "[...] Upon investigation, we have determined that this submission does not meet the definition of a security vulnerability for servicing. This report does not appear to identify a weakness in a Microsoft product or service that would enable an attacker to compromise the integrity, availability, or confidentiality of a Microsoft offering. [...]"

We thank and acknowledge Microsoft for their feedback even if in the Case 65533, we do not really understand their reference to WPAD Name Collision Vulnerability (explained in Section 3.6.2,

[17][18]) since our report describes a distinct problem. In all three cases, it is somehow worrying to note that, what appears to be a vulnerability to us, seems to be considered as a non existing problem for Microsoft. We leave it up to the reader to decide which position to embrace.

3.6.1 Seminal work. The first WPAD related vulnerability was reported in CVE-1999-0858 [59], more than 20 years ago. It discloses the DNS devolution problem in Internet Explorer 5. This issue was also published during the same year in a Microsoft Security Bulletin [52].

Surprisingly, the first paper that addresses the issues of WPAD was only published in 2003, i.e., 4 years after the expiration of the WPAD draft and 4 years after the first WPAD related CVE. This paper consists of a cautionary note [87] that alerts about WPAD based attacks. It briefly explains different ways to attack WPAD thanks to the DHCP, SLP, and DNS protocols. Additionally, this paper addresses the dangers of having a malicious proxy intercepting the victim's machine's SSL/TLS traffic through a MITM attack.

3.6.2 Impersonating a WPAD server: In a local network, some machines are configured to use WPAD by default. An attacker can impersonate the WPAD server by answering the legitimate WPAD requests or by forcing the machine to query a malicious WPAD server.

We discuss in this section the different techniques discovered to impersonate the WPAD server.

CVE-2007-1692 [60] reports that an attacker is able to answer DNS requests or NBNS requests for WPAD normally made by the client machine. 2 additional vulnerabilities were reported in 2009: CVE-2009-0093 [62] and CVE-2009-0094 [63]. In the first vulnerability, it was asserted that Windows DNS Servers permit a malicious actor to register the "wpad" hostname via a Dynamic Update request (a DNS Dynamic Update enables a client computer to register and to dynamically update their DNS records in a DNS server). In the second, it was shown that the WINS server doesn't restrict the registration of "wpad" as a NetBIOS name, which would enable remote attackers to register the WPAD name and redirect the network traffic to their systems. Microsoft released a security update [54] that resolves these 2 issues. A security advisory [53] was also released for this and a US-CERT alert was published [12].

Forcing the machine to contact a WPAD server controlled by the attacker can be made using different techniques: adding a malicious DNS entry, forcing the machine to resolve a domain locally, or by spoofing DHCP.

In 2010, *DNS hijacking* was discussed in a Black Hat training session [84]. According to [84], it is "*possible to inject names into the local DNS zone by passing these names as the hostname of DHCP client requests (the -h option for the ISC dhcpcd client)*". A machine can thus inject a name of its choosing within the local DNS zone. If an attacker chooses the WPAD name for itself and successfully injects it, the DNS server will provide the attacker's IP to all machines searching for a WPAD server. "*Once the attacker has become a relay for outbound connections, the door is opened for a wide range of attacks, such as client-side flaw exploitation, cookie theft (and by extension, credential theft), cross-zone data access, and the injection of UNC URLS into the returned HTML.*"

In 2010, [3] mentions briefly that injecting a domain suffix into the search list results in the machine making WPAD requests for a domain controlled by a malicious actor. In this case, the machine would make DNS requests to reach the attacker's server and download a malicious PAC file.

Hijacking NBNS requests was discussed in 2010 in [47] which shows how this technique can be used to propagate malware in a local network. The victim machine uses a malicious proxy configuration. Its web traffic is redirected to a proxy controlled by the attacker. Whenever the infected machine makes HTTP requests to download a file, the proxy replaces the legitimate file and returns a malicious one that has the same metadata. This specific technique was used by the Flame malware [24] to replicate itself. Further information on this is presented in Subsection 3.7.

The Evil Foca tool [5] presented in 2013 and also mentioned in 2014 [33], allows to automate impersonating a WPAD server locally: it answers LLMNR requests for WPAD and sets a rogue proxy server.

Impersonating the WPAD server can also be done remotely due to the DNS devolution problem. As mentioned in Section 2.2, the devolution mechanism consists in the machine removing the top left component of a network name, making DNS requests for the obtained domain and repeating this process until the server is found or until the TLD is reached. This can lead the client to use servers outside of the organizational boundary while attempting to access the host name and thus leaking internal DNS requests for a WPAD server to external DNS servers. Even though this problem was disclosed in CVE-1999-0858 [59] and later in CVE-2007-5355 [61], it remains an active threat as shown by our investigations described in Section 4.1.3.

In 2008, [97] reported DNS leaks of the "wpad.nu" and "wpad.fm" domains. The first thorough analysis of such leakage was made 8 years later in 2016 by [17]. This study discusses the dangers of registering special DNS domain names and impersonating WPAD servers. The researchers show that a large volume of WPAD DNS queries get sent to public DNS nameservers. These queries are, in fact, internal WPAD DNS queries leaked to public DNS namespaces. To impersonate the WPAD server, an attacker only has to register one of these domain names. This is easily doable since the expansion of new gTLDs in 2013 [38] via the introduction of new top-level domains in ASCII characters and in different scripts. The risk of name collisions (external WPAD domains coinciding with internal WPAD domains) is high and this exposes machines to MITM attacks. [17] explains the main cause of WPAD leakage: devices with common configurations can mistakenly generate internal queries when used outside of corporate networks. It also defines highly vulnerable domains (HVD) as domains that *persistently* expose a *large number* of victims. The authors find that all leaked queries are for HVD with new gTLDs. Only 10% of these domains are registered.

Following the release of [17], US-CERT issued in 2016 an official warning about the WPAD name collision vulnerability [14].

In 2016, an additional attack exploiting NBNS was discovered: BadTunnel [102] is a sophisticated attack that allows an attacker to hijack the victim's network traffic by disguising as WPAD. It is based on the following key steps:

(1) The victim's browser is deceived into visiting the WPAD domain. This URL can be in an embedded resource in a webpage for example.
(2) The victim machine sends NBNS requests for WPAD after DNS fails to find the resource.
(3) The attacker provides a malicious NBNS response.

Further technical details are presented in the paper. This attack was also presented in the Black Hat conference [101].

In 2017, [7] describes an issue related to WPAD and DHCP: the attacker can impersonate the DHCP server and can answer the DHCPINFORM message by spoofing the DHCPACK message. Next, it can provide a malicious CURL. For the attack to succeed, the victim machine has to receive this ACK message before the ACK message of the legitimate DHCP server. CVE-2017-6410 [75] points to the fact that some systems still use WPAD by default to locate a proxy configuration file.

3.6.3 NTLM relay: In 2010, [84] presented an NTLM [42] relay attack that combines hijacking WPAD and SMB. "NTLM is a suite of authentication and session security protocols used in various Microsoft network protocol implementations" [27]. NTLM over SMB transport is a common use of NTLM authentication and encryption. In this paper, the authors present an attack that enables them to execute remote code on a victim machine: the machine is deceived into using a malicious proxy which directs the web client to an SMB share. Depending on their operating system's version, some machines automatically negotiate NTLM authentication with the remote SMB server revealing their credentials. But even in the case that such negotiation is not possible, a more sophisticated attack exists: the attacker provides a malicious SMB server which relays the authentication process to an IIS or an Exchange Server. In this case, the machine makes an SMB connection via the attacker's server and the malicious actor obtains an authenticated session on the client's system. This leads to executing remote code on the victim's machine.

Responder [93] is a LLMNR, NBNS and MDNS poisoner. Using this tool, it is possible to steal the victim's credentials: when the victim makes a WPAD name resolution, the attacker impersonates the WPAD server. The victim machine makes an HTTP GET request for the PAC file and the attacker creates a fake authentication page. The client is then asked to enter their domain credentials.

Google Project zero [103] introduced in 2014 an NTLM reflection attack which enables a local user to elevate privileges to the local system. Additionally, the Hot Potato attack [10], when performed by even a low privilege user, affects all users of the machine. Both of these attacks are based on a similar technique: namely NTLM relay and NBNS spoofing.

In 2016, 2 CVEs (CVE-2016-3213 [70] and CVE-2016-3236 [71]) report that, thanks to WPAD, an attacker can carry an elevation of privilege attack. Microsoft published two bulletins on these issues [57] [58]. In [58], it is asserted that:

(1) The location of the file is no longer requested via NBNS;
(2) Authentication does not occur automatically anymore even if it is requested by the webserver.

Despite the patch made by Microsoft, it is still possible to find some workstations that, when prompted for authentication, automatically attempt to authenticate with NTLM authentication [41]. Additionally, the protections mentioned above can be easily bypassed, as demonstrated by a new version of this attack a year later [83]. The attacker provides a malicious wpad server via DNS. To bypass the second protection, the attacker does not ask anymore for an authentication at the WPAD server level but it does so at the proxy level. In fact, when trying to connect to the Internet via the proxy, the machine gets "HTTP 407 Proxy Authentication required" and is provided with a 16-byte random number called the "challenge". Subsequently, Windows will send back the challenge encrypted with the hash of the user's password. The password can be extracted and then used by the attacker to have access to different services.

In 2018, exploiting WPAD was presented in the HITB Security Conference [100] as a typical NTLM relay attack.

A similar technique was presented in 2019 [43]. In 2020, [46] explains this issue and makes references to [10] and [103].

Even though the first paper that discusses this problem [84] was published in 2010, new attack scenarios based on WPAD and NTLM relays were still reported 10 years later [21].

3.6.4 SSL man-in-the-middle attacks. Pretty-Bad-Proxy (PBP) attacks [19] bring additional security concerns regarding WPAD: *"PBP is a malicious proxy targeting browsers' rendering modules above the HTTP/HTTPS layer."*

Setting a malicious proxy that intercepts the client's web traffic is a great opening to make a MITM attack. This becomes even more dangerous for the victim machine if the attacker can have access to the content of the HTTPS traffic. [37] discusses the following scenario: the machine is deceived into using a malicious proxy. This one inserts itself into the communication channel between the victim client and the server by establishing 2 separate SSL connections: one with the client and one with the server. The browser usually presents SSL certificate warnings in this case (which are often ignored by the user). This can also be used in banker infections: the malicious proxy redirects the user to a cloned banking website for example [24].

3.6.5 Executing malicious Javascript code: CVE-2009-3372 [64] states that some malicious regular expressions in the PAC file may result in the browser crashing and in the attacker executing arbitrary code on the victim machine[3]. Moreover, according to CVE-2012-4776 [68], some Microsoft .NET Frameworks have a lack of validation when acquiring the PAC file. This bypasses the security mechanism preventing untrusted code from performing privileged actions (Code Access Security CAS). In fact, the Javascript code in the PAC file is not validated, which enables remote attackers to execute arbitrary code. US-CERT alerted about this in 2012 [13] and Microsoft published a Security Bulletin about it [56]. CVE-2012-2915 [65] and CVE-2012-4504 [66] report stack-based buffer overflow vulnerabilities in PAC files. Also [67] addresses a heap-based buffer overflow and CVE-2012-5580 [69] reports a format string vulnerability.

A new approach to WPAD exploits was mentioned in StackExchange [22] in 2015. It was also detailed in 2016 during the Black Hat conference [44]. The aim of this attack is to get the full URLS (HTTP and HTTPS) a machine requests. This is useful because in some cases these URLS may contain some credentials or authentication tokens. The attack scenario is based on 2 information:

- Two of the functions that PAC files support are *dnsResolve* and *isResolvable*. These functions allow to make DNS lookups for host domain names.
- The PAC file is executed each time a URL is visited by the web client.

Knowing that, the attacker provides a malicious PAC file that:

(1) chops the URL into small chunks that can fit into a DNS query
(2) appends the attacker's domain name to each chunk
(3) invokes *dnsResolve*

The attacker collects these lookups and has access to the URLS fetched by the victim machine.

This technique was discussed in several occasions: in 2016 at the IEEE SSP Conference [28] as well as in the DEF CON conference [16]. In [16], very interesting demos of this exploit showed how leaking URLS allowed to have access to the victim's Google Id, Github Id, LinkedIn Id and several other sensitive information. This technique also enabled attackers to get the victim's OAuth logins [34], Google session, and Google Drive files. This attack bypasses VPN if the wpad server hosting the PAC file is reachable via a public IP. Indeed, the machine connects to the VPN server through a secure tunnel but, since the WPAD server is reachable by the VPN server, it will connect to the attacker's server and the client machine will eventually download the malicious PAC file. This is reported in CVE-2017-5384 [74] which explains that some malicious Javascript code can leak full HTTPS URLS to the attacker's server. The bugs were fixed in Mozilla Firefox [11] [85].

A malware can also inject the malicious PAC file's URL locally on the victim's machine. For Windows, this URL is stored in the Windows registry [44]. This vulnerability will further be discussed in Subsection 3.6.9.

Google Project Zero released in 2017 [25] a blog post about exploiting Windows 10 in a local network by chaining several vulnerabilities together. This attack was dubbed "aPAColypse Now". Despite the limited set of functions allowed in the PAC file, it chains two specific JScript bugs (an infoleak and a heap overflow) during the interception of the malicious Javascript file, and bypasses Windows security mitigation techniques using several other exploits. Following this, a privilege escalation technique is applied to gain the SYSTEM account privileges.

Additional vulnerabilities were found in the implementation of PAC files: CVE-2018-18358 [77] and CVE-2018-18506 [78] report that a malicious PAC file is able to direct the URLS that have "localhost" as a domain name via a proxy controlled by the attacker. This behaviour should not happen since the browser is supposed to connect to localhost using the internal loopback interface [20].

[3]Despite extensive search, we did not find more technical details about this CVE but we mention it anyway for the sake of completeness

The [25] attack was also presented in the Black Hat conference in 2019 [35].

3.6.6 Argument-injecting attack: Sensible-utils [50] is a Debian package which provides a collection of small utilities used to "sensibly select and spawn an appropriate browser, editor or pager". Among these utilities, sensible-browser [51] enables to launch the browser specified by the BROWSER environment variable, with an input string corresponding to the URL to be fetched.

CVE-2017-17512 [72] reports that this string is not validated before launching the browser, i.e., if the string url contains an Input Field Separator character, this leads to the injection of extra arguments when calling the browser.

For example, the following string "http://www.example.com/ –proxy-pac-file=http://dangerous.example.com/proxy.pac" enables to fetch *http://www.example.com/* while using the proxy settings specified by the PAC file found in *http://dangerous.example.com/proxy.pac*. This enables remote attackers to forward the traffic to their server after conducting an argument-injection attack via the –proxy-pac-file argument.

CVE 2017-17523 [73] and CVE-2018-10992 [76] report the same argument-injection vulnerability in lilypond-invoke-editor, a small helper program in LilyPond 2.19.80 [48], a music engraving program. Note that CVE-2018-10992 exists because of an incomplete fix for CVE-2017-17523.

3.6.7 Other recent concerns: The recent CVEs reported are:

- CVE-2019-8454 [79] reports a vulnerability with Check Point Endpoint Security clients that allows injecting a malicious PAC file into the victim machines.
- CVE-2020-26154 [80] specifies that a large PAC file that is delivered without a Content-length header can cause a buffer overflow in libproxy.
- CVE-2021-0393 [81] reports that it is possible to take advantage of an out-of-bounds write bug in Android to execute a malicious code provided by the PAC file.

3.6.8 DHCP attack revisited: As explained before, one way to give a client a bogus WPAD server name is to use the option 252 in DHCP. Since most networks do have one (or several) DHCP server(s) up and running, the attacker must either pose as another valid DHCP server or spoof the identity of a valid one and send a valid reply to the client before the genuine server. In both cases, the success of the attack is not certain as it relies on the ability of the attacker to respond faster than the other DHCP servers [25].

There is a much simpler way to *push* the 252 option into a client though by leveraging the fact that a DHCP server sends two distinct messages to the client: a DHCPOFFER message and a DHCPACK one. Option 252 is typically provided in the first message but our experiments show that, if provided in the second one, the value will happily be accepted by the client. Being able to send a valid DHCPOFFER before the server might be tricky for an attacker (race condition) but it is trivial to achieve for the DHCPACK one since the attacker, as opposed to the server, does not need to wait for the client to send its message. It makes it extremely simple for an attacker to succeed in providing a 252 option to every DHCP client with almost 100 % chances of success. Worse, if the DHCP server uses the 252 option in its DHCPOFFER message and the attacker also in a following spoofed DHCPACK message, the client will keep the latter value, i.e., the malicious one. This is not a new vulnerability, *per se*, it is the way the protocol works.

Another very efficient way to misuse DHCP to carry out an attack against WPAD has to do with the DHCP option 119 (the DNS "domain search" option [1]) and the DHCP option 15 (the DNS "domain name" option [4] of the client). As explained before, the WPAD client will prepend the string "wpad" to these various network names in order to search for a valid WPAD server. Option 119, in particular, is very appealing as it provides a list of names. The attacker only has to add its own, external, network name to ensure that the client will reach out to him (if no other local WPAD server exists, of course). As in the previous case, this attack can be run thanks to the second message to be sent by the server, making the success almost guaranteed for the attacker. An attacker can observe the DHCP traffic in a network in order to retrieve the search list normally provided by the legitimate DHCP server. It then appends to this list a malicious domain and supplies the new list to victim machines; only the search list provided in the DHCPACK message is taken into account.

3.6.9 Reboot-resistant attack due to Windows implementation choices. Our experiments also showed that the Windows machine stores the URL of the PAC file in its registry if the following conditions are satisfied:

- The file has either a ".dat" extension or a MIME type equal to "application/x-ns-proxy-autoconfig"
- The file contains a proper FindProxyForUrl function
- The machine is able to reach the Internet using the proxy configuration specified in the PAC file

It is worth noting that an apparently innocuous file entitled, for instance, "*index.html*", can be accepted as a PAC file as long as its MIME type is "application/x-ns-proxy-autoconfig". A contrario, any filename that ends with ".dat", eg. "photo.dat" will also be accepted, even if his mime type does not seem related at all to a proxy file, such as, eg., "image/jpeg".

If the file is valid, Windows will store the URL of the PAC file in the following registry key:
Computer\HKEY_USERS\S–1–5–19\SOFTWARE\Microsoft\
Windows\CurrentVersion\Internet Settings\Wpad\WpadDetectedUrl

We found out experimentally that this registry value is not deleted after a reboot. As long as the machine is able to connect to the proxy and as long as no other WPAD server is offered to the client, this URL will be used after each reboot to connect to the proxy. In other words, if you use a machine within an insecure network where any of the previous attacks can succeed, the registry could be poisoned by a malicious CURL and rebooting the machine before connecting it to another, supposedly secure, network will not clean it from that malicious input. Despite not being subject to any attack anymore, the machine will keep using the proxy (assuming it is reachable, e.g., as a public server on the Internet), *de facto* exfiltrating information from the inside of the secure network to the outside world. In other words, any Windows machine connected, even only once, to an insecure network could then trivially be a MITM attack victim for a very long time, no matter which other network it is later connected to!

3.6.10 WPAD keeps running despite being explicitly disabled on Windows. In the Windows proxy settings, the user can "theoretically" disable the WPAD protocol by turning off the "Automatically detect settings" option. Surprisingly, we discovered that this is not enough to completely put out of action WPAD, i.e., not use WPAD to find a proxy in order to connect to the Internet.

As a proof of concept, we run an experiment with a Windows client machine which has the "Automatically detect settings" option turned off. When the machine starts, the DHCP server provides, at the IP allocation phase, a 252 option which contains a valid CURL.

Not only does the machine accept the 252 DHCP option but it also stores the provided CURL in the Windows registry. It also replaces any previous WpadDetectedUrl registry value. The Windows machine should have ignored the provided 252 option. But in reality, it accepts it, contacts the WPAD server, downloads the PAC file and executes the Javascript code. Combining this bug with the Javascript issues explained in Subsection 3.6.5 and the persistence problem described in Subsection 3.6.9 makes the machine very vulnerable, despite its user having explicitly (apparently to him) disabled the usage of WPAD.

3.6.11 Conclusion: WPAD has been discussed on several occasions and its vulnerabilities have been presented in several research papers. Every protocol WPAD relies on to discover a server has been proven to be vulnerable: DHCP option values may be spoofed, DNS queries may be leaked, local attackers may impersonate a WPAD server, a malicious Javascript can be embedded in the PAC file and the HTTPS traffic may be hijacked. Additionally, the mechanisms on which WPAD relies do not use any form of authentication. This makes impersonating the WPAD server very easy for attackers.

From the previous subsections, one can easily derive an attack scenario that has a very high chance of success that would be very stealthy and highly persistent.

Imagine a client machine in a public WIFI hotspot (bar, airport, restaurant, etc.). To get an IP, the client will look for a DHCP server. An attacker will see the new comer arriving and can trivially succeed in pushing a valid CURL into its machine. The PAC file downloaded by the client can have a name and a type that looks genuine. If that client machine goes into another environment (e.g., at home or within its company intranet), as long as there is no valid WPAD server running in those networks, all communications (http, https, smtp, etc.) can be redirected through a hostile third party. Worse, the attacker may write a PAC file that only redirects some very specific traffic to its proxy, leaving the rest of the network activities of the client unchanged.

Worse, it appears that some of the issues described in the previous Subsections are not specific to WPAD but impact many other protocols as well. The authors of [17] published a follow up paper [18] in which they study further the collision problem caused by DNS leaks. It was found that this issue exists for several other services (for e.g., isatap [96], ldap [94] etc.,). "Out of the 48 identified exposed services, we find that nearly all (45) of them expose vulnerabilities in popular clients". WPAD appears to be only the visible tip of the iceberg.

3.7 Answer to research question RQ-3: How was WPAD exploited in the past by malicious actors?

Over the years, some malicious actors took advantage of the vulnerabilities found in the Web Proxy Auto-Discovery protocol to replicate malware over the network, to redirect the user to phishing pages or to look into the HTTP/HTTPS traffic of internal networks to possibly steal credentials or access tokens. WPAD was used in 2 major exploits in the past (Flame Malware [104] and Trojan bankers [6][8]). First, Web Proxy Auto-discovery was used by the Flame malware [104][24] to replicate itself on an infected network. It was discovered in 2012 and was a massive, highly sophisticated cyberespionage malware that targeted computers from the Middle East. It exploited WPAD to impersonate Microsoft Windows Update hosts. Whenever a machine configured with automatic proxy detection broadcasts WPAD NBNS packets, the attacker answers those queries with their own IP address. When the computer tries to access one of the Windows Update hosts, it will be redirected to the attacker's proxy. A malicious file is then delivered by the proxy as a Microsoft update file. Most of the payloads had valid signatures (now revoked by Microsoft). The victim machine receives and processes these files as valid updates, thus executing the malicious code.

Additionally, several trojan bankers have also been known to force infected machines to access malicious PAC files since 2009. Blackmoon [8] was discovered in 2014. It was a banking trojan designed to phish user credentials from various South Korean banking institutions. Whenever a victim machine tries to access a banking website, the URL is going to be fetched by the attacker's proxy and the user is going to be redirected to a phishing page. In 2013, most of the Brazilian Trojan bankers were known to have a feature which can add a malicious PAC to the browser's config [6][49]. Cybercriminals were also known to use obfuscation and encryption to make the detection of malicious PAC files harder.

4 ONGOING ATTACKS RELATED TO THE WPAD PROTOCOL

4.1 Real world investigations

4.1.1 Foreword. While carrying out this study, we were quite surprised by the relatively small number of known attacks leveraging WPAD. One reason could be that these attacks are well suited for targeted and stealthy attacks which are difficult to detect and rarely make the headlines. Nevertheless, we thought it was important to find some evidence that these threats were still real and actively exploited. Three distinct approaches have been taken: *i)* looking for suspicious PAC files in the wild thanks to Farsight passive DNS database, *ii)* looking at the Polish WPAD sinkholes, *iii)* searching zeek logs thanks to Vern Paxson.

4.1.2 Finding suspicious PAC file thanks to DNSDB. Thanks to Paul Vixie and the members of Farsight company, we have been able to search for recently leaked WPAD names in DNSDB [92]. We searched the database for wpad.* domains and crawled each match for a "wpad.dat" file since this is the default name to search for when discovering WPAD servers through DNS. We have retrieved many such files and also came across a number of sinkholes (see next Section). One case caught our attention: 25 WPAD domains (such as wpad.rs., wpad.msonline.co.za., wpad.ke., wpad.bi, wpad.fo, etc.) were linked to the same IP address (46.8.8.100). All these servers were providing the same "wpad.dat" file which had the following content:

- If the destination domain is not: *.windowsupdate.com , *.microsoft, com , *.baidu.com, *.kaspersky.com, *.axaltacs.net, *.live.com or *.dhl.com …
- … **and** if the target IP is not reserved for loopback (127.0.0.0/255.0.0.0) or for Private Networks (RFC 1918): 10.0.0.0/255.0.0.0, 172.16.0.0/255.255.224.0, 192.168.0.0/255.255.0.0 …
- … **then** use address 185.38.111.1 on port 8080 as a proxy.

That IP address 185.38.111.1 is also linked to the very peculiar FQDN *wpad.domain.name* which was found to be part of a WPAD attack campaign that lasted from November 24 until December 14, 2017, using the very same PAC file, described in detail in [45]. As of July 26 2021, that IP is also flagged by a number of DNS black lists. The proxy was up and running until the end of June but was not responsive anymore in July 2021.

We came across many other peculiar WPAD servers but could not correlate their names or IP with a known attack campaign like in the previous case. We note that, even in this case, we cannot, for sure, ascertain that this proxy acts maliciously. The very large number of public WPAD servers providing valid PAC files suggest that some of them are very likely doing so with malicious intent. It would be worth carrying out a deeper investigation of this suspicious situation but such analysis lies outside the scope of this survey.

4.1.3 Results found from sinkholing WPAD domains . In 2016, at Black Hat, Maxime Goncharov looked into WPAD leaked names [29]. He found a large number of hits for registered WPAD domains and warned about the likelihood of ongoing targeted attacks. He highlighted a registrant owning several peculiar WPAD domain names (wpad.pl, wpad.sk, wpad.be etc.,) and also retrieved a suspicious PAC file from these servers. This investigation was continued by the members of the Polish team RedTeam.pl and they reported their findings in a blog post in 2019 [105]. They found a number of names registered to carry out WPAD attacks. CERTs of the targeted TLDs were contacted and they took action. Some remaining domains were taken over by Cloudflare and the Polish CERT (NASK [9]) established a sinkhole for all WPAD domain names ending with ".pl". Members of that team were kind enough to share with us some recent statistics from their sinkholes that illustrate how real is the risk related to WPAD names leakage:

- NASK monitors 153 WPAD domains.

- The sinkholed domains include names such as wpad.pl but also WPAD subdomains for functional and regional domains in .pl such as mail.pl, edu.pl, warszawa.pl, etc.,
- Between May 27th, 2021 and July 12th, 2021:
  1.13 millions: is the daily average number of hits for the monitored domains.
  5800: is the daily average number of distinct IP addresses querying the sinkholes.
  "wpad.com.pl" and "wpad.pl" are the top 2 domains per number of hits and also per number of IP addresses.

These numbers clearly highlight the opportunities for attackers to register WPAD domains and carry out attacks very easily. It also shows that DNS devolution is still carried out, up to the top level domain, by a large number of clients in the wild. This is another evidence of vulnerable implementations waiting to be attacked. Our previous analysis highlighted the existence of servers, potentially malicious, waiting for clients to query them. This latest analysis confirms the existence of numerous misguided clients that, most likely, erroneously query these public servers. Studying the, probably nefarious, consequences of these thousands of unexpected queries would be interesting but lies outside the scope of this survey.

4.1.4 Zeek logs. Last but not least, we have tried to find traces of ongoing attacks in Zeek logs [90] (formerly known as Bro [89]). On top of looking for specific signatures, Zeek keeps a number of log files that can be used, among other things, for post mortem analysis. The "http.log" file, in particular, keeps a record of every file downloaded with the HTTP protocol as well as its MIME type. A client victim of a WPAD attack, carried out in a non stealthy way, will leave a trace in that file by means of a log entry for a downloaded file of MIME type "application/x-ns-proxy-autoconfig". Vern Paxson was kind enough to run a query on our behalf on the logs of 6 sites which span industry, a national lab, and a large university, totalling 33 months of activity, with close to a million distinct IP monitored daily. Much to our surprise, there was not a single such entry, yet there were a number of "wpad.dat" files downloaded with a MIME type "text/html". Without having access to the downloaded files, we cannot tell whether they were malicious or not. It is strange though to only see a very small fraction of machines downloading these files. One could expect that, as part of an organisation policy, either all machines, or none, will download such files. Investigating this further lies beyond the scope of this paper but this analysis highlights the difficulty in discriminating between legit and malicious use of these WPAD files.

5 MITIGATION TECHNIQUES

5.1 Prevention

Following the paper [17] which introduces the name collision problem explained in Section 3.6.2, a white paper [98] has been published in 2016. It proposes enterprise remediation strategies for the WPAD name collision vulnerability:

- On end-systems, [98] recommends to disable the WPAD protocol whenever web proxies are not needed. On the other hand, in an enterprise that makes use of these proxies, it is best to hardcode the corresponding IP addresses on each end-system. Another solution would be to configure end-systems to run local DNS name servers in order to avoid leaking queries to the external DNS servers.
- In the enterprise network, it is recommended for internal systems to use FQDNs instead of internal top-level domains (iTLDs). This avoids leaking wpad.*.iTLD lookups to external servers. Additionally, it is best to configure internal DNS resolvers to drop and report on any outbound queries for wpad.*. Any outbound requests for a "wpad.dat" file should be blocked by a Application Level Gateway (ALGs), a Next-Generation Firewall (NG-FW) or a web proxy.

- ISP outreach: It is recommended to drop all outbound wpad.* queries.

As described in our paper, the attack surface for WPAD is quite large and fixing all possible problems is an almost impossible task. A long term mitigation solution would be to design, implement and deploy a secure service discovery solution. In the meantime, we believe three distinct options are possible:

(1) Disable WPAD on all clients, at the operating system and browser level, by means of a system compliance software whenever web proxies are not needed.
(2) Make it mandatory for clients to use WPAD within the network you need to protect and block any outgoing traffic that does not go through the genuine, internal proxy. While this approach helps in securing a given network, it renders the client machines vulnerable to external attacks whenever they move into another network (e.g., airport, bar, etc.,)
(3) Install a network based intrusion detection system to identify clients that could have been victims of such an attack. This is not trivial as the attacks could be very stealthy, especially if the CURL contains an IP address rather than a name (no DNS query) and leverages HTTPS rather than HTTP (no visibility). Moreover, if the PAC file redirects a targeted set of domains, the network behavior of the infected machine will not be out of the ordinary. The next subsection covers these techniques, together with their limitations.

5.2 Detection

In 2016, a patent [99] was filed for detecting MITM attacks. Among other things, this document presents a method for identifying an impersonation of a WPAD server (made by spoofing name resolution protocols such as NBNS, LLMNR and MDNS) in order to provide a malicious PAC file. The implemented tool proposes to make a request for a PAC file and, if a response is received, compares it to a prior legitimate version. If the two versions match, then no MITM attack is identified; if they do not, then an attack is confirmed. On the other hand, if authentication is required to have access to the PAC file, the tool sends again another request for the configuration file, but this time with fake credentials. Finally, if the authentication succeeds, then an attack is confirmed (this means that an attacker is collecting credentials by forcing victims to authenticate when searching for a WPAD server). If the authentication fails then no MITM attack is confirmed.

The authors of [17] also filed in 2017 a patent [86] in which they present solutions to detect highly vulnerable domains using passive DNS measures.

We propose three additional, complementary, approaches for the detection and mitigation of the WPAD related attacks:

(1) *Detection of vulnerable machines*
(2) *Pre-compromise detection*
(3) *Post-compromise detection*

Each approach is described in its corresponding subsection here below.

5.2.1 Detection of vulnerable machines.

Some security network administrators consciously decide to run WPAD within their network. This is a convenient approach if, for instance, they have a web application firewall (WAF) to access the Internet. However, as discussed later in Subsection 5.2.2, it makes them vulnerable to an insider attack. Worse, if some of these machines are mobile, they could end up being connected into an insecure network. The example of a laptop connected to a public hotspot comes to mind[4]. If its default configuration leads it to search for a WPAD server, it is at the mercy of any malicious user connected to that foreign network.

[4]Note that the usage of a VPN in such insecure networks will not protect the machine against WPAD attacks [16]

To mitigate WPAD attacks run by insiders or outsiders, it appears to be a good practice i) to not run WPAD within the network you aim at protecting, ii) to detect all machines that run WPAD by observing the service discovery messages they will send, iii) to modify their default configurations.

Unfortunately, as explained before (subsection 3.6.10 on page 17), it does not seem possible to prevent Windows machines from using WPAD, at least not totally. Furthermore, there are situations where using WPAD could improve security; in a network with stationary machines protected by a WAF, for instance. In those cases, it is important to be able to detect attempts at running WPAD attacks, within the network, before a successful compromise. We discuss this in the next Subsection.

5.2.2 Pre-compromise detection. An insider has several options to run a WPAD attack against a vulnerable machine.

(1) It can misuse DHCP by sending a spoofed packet containing the 252 option with the CURL he wants to inject into a target.
(2) It can misuse DHCP by sending a spoofed packet containing the options 15 (the DNS "domain name" option [4]) or 119 (the DNS "domain search" option [1]) to incite the client to connect to a WPAD server outside its network.
(3) It can misuse DNS by sending a spoofed reply to a query for a wpad.* domain name, containing the IP address of a machine it controls.
(4) It can run one of the many MITM against one of the zero conf protocols used to discover a valid WPAD server [23].
(5) It can run a link layer attack, e.g., ARP poisoning, to impersonate a genuine WPAD server.
(6) It can run a session hijacking attack to push a malicious PAC file when a client connects to a valid WPAD server (unless if HTTPS is used)

All these attacks will leave some observable fingerprints in the network traces. One could think of using white lists with known domain names and IPS of the genuine WPAD servers to spot the malicious ones. One could also detect spoofed packets by the existence of duplicate packets with distinct contents. That might seem feasible, relatively simple and with a very low false negative rate.

Unfortunately, such detection is prone to so-called *squealing attacks* [88] in which the attacker crafts packets that will generate false alarms, with the hope that the defender will eventually shut down the detection mechanism. It is indeed trivial to send fake duplicate ARP, DHCP or DNS packets, for instance, without any risk of being identified.

Leaving these squealing attacks aside, even if we detect no vulnerable machine or insider attack, we cannot ignore the fact that a machine could have been previously compromised outside the network. It is thus important to also look for signs of these already compromised machines. This is discussed in the next Subsection.

5.2.3 Post-compromise detection. A compromised machine is one in which an attacker has successfully injected a malicious CURL. We need to detect when the malicious CURL is used. For the attack to be successful, the victim must i) resolve the FQDN of the WPAD server found in the CURL, ii) download the PAC file from it and iii) forward its traffic to the proxy defined in the PAC file.

Each step offers opportunities for detection as well as difficulties, namely:

(1) **DNS:** The defender detects the DNS resolution of the WPAD name server.
  Pros: any non white listed DNS query for a FQDN starting with *wpad* is flagged. The risk of false positives should be pretty low.
  Cons: the WPAD server name does not need to start with *wpad* (this is just a convention) and the CURL could contain an IP instead of a name: high risk of false negatives.
(2) **PAC file transfer:** The defender detects the transfer of a PAC file.

Pros: Any HTTP file transfer for a file with the *.dat* suffix or the MIME type "application/x-ns-proxy-autoconfig" coming from a non whitelisted WPAD server is flagged.

Cons: The very common suffix *.dat* could lead to high false positives. Furthermore, attackers could insert objects with these characteristics into ads or genuine web pages to cause a *squealing attack* [88].

(3) **PAC file content:** The defender checks the content of suspected PAC files.

Pros: Any suspected PAC file that contains a *FindProxyForUrl* function is confirmed to be a PAC file.

Cons: The PAC file could be obfuscated using, e.g., the *eval* function [6], leading to false negatives. It is also trivial to insert that string into any file used for a *squealing* attack.

(4) **Proxy behavior:** The defender flags machines with anomalous HTTP traffic.

Pros: Compromised machines will only have HTTP or HTTPS traffic to a single IP address, that of the proxy, as opposed to all other machines.

Cons: The PAC file can redirect only some very specific traffic to the malicious proxy, leaving the rest unchanged, leading to possible high false negative rate.

Furthermore, since the CURL can be specifying the use of HTTPS instead of HTTP to fetch the PAC file, this will defeat the techniques 2 and 3 described here above.

Last but not least, the idea of clustering the network behaviours of clients with the hope that the compromised ones will end up in the same group can also be defeated since different clients requesting the same CURL could receive a different PAC file, based on a number of criteria such as the "User-Agent" header, the client IP address/name. This means that compromised machines with the same CURL would not necessarily exhibit the same anomalous behaviour, if any.

To summarize, one can think of a number of ways to detect these attacks and, in many cases, this can be done successfully as long as the attack is not designed to be stealthy. Unfortunately, as shown in the next Section, in the worst case scenario all is lost.

5.2.4 An example of a barely detectable attack. Let us imagine the following scenario: the victim is a mobile user who connects its Windows 10 machine to a public hotspot in which the following CURL is pushed into its registry: "*https://malicious_public_ip/index.html*" where "*malicious_public_ip*" is the public IP address of a WPAD server controlled by the attacker and "*index.html*" is a PAC file (downloaded with the MIME type "application/x-ns-proxy-autoconfig") which only redirects a small set of targeted domains to a malicious proxy. Once connected back in its corporate network, the machine will not make any DNS lookups for an explicit WPAD, only an HTTPS connection to an IP address. Neither the name nor the content of the PAC file will be visible.

The only suspicious elements could be i) the establishment of an HTTPS connection to an IP which has not been resolved through a DNS query, ii) the absence of some outgoing HTTP and HTTPS traffic that the machine used to do before being compromised. The latter element can only be detected if a complete model of all outgoing connections of each machine is maintained, which is not realistic in the general case. The former is an actionable piece of information but highly likely to lead either to some false positives because of cached or hard coded IPs in some applications, or to some false negative in large networks due to the sheer volume of DNS and HTTP/HTTPS requests. In other words, in such a scenario, very simple to implement, the presence of a compromised machine exfiltrating data will, very likely, remain invisible.

6 CONCLUSION

This paper has offered an exhaustive and systematic review of all the known vulnerabilities related to the usage of the WPAD protocol. The study has included a presentation of the definition of the protocol as well as of its observed implementations. A number of research questions have been asked and answers provided that outline the severity of the problems posed by the exposure of machines to this protocol. We have also presented new ways to misuse the protocol and have carried out some investigation that revealed the reality of the presented threats in today's Internet. Additionally, we have offered some thoughts on possible prevention techniques. Last but not least, we have proposed some detection techniques of WPAD related attacks and discuss their limitations The real conclusion of this work is that WPAD is a protocol that is widely used, vulnerable in many different ways, hard to fix and whose attacks against it can be very stealthy. Everyone should seriously worry about it and take appropriate measures against it. Unfortunately, it only is the tip of the iceberg as many other protocols rely on the same underlying insecure service discovery methods, as explained in the course of this study and detailed in [18].

ACKNOWLEDGMENTS

We want to express our deepest gratitude to the members of the Farsight and the Corelight companies, Redteam.pl and NASK for having helped us in our investigations reported in Section 4.1. In particular, we want to thank, by alphabetical order, Przemek Jaroszewski, Piotr Kijewski, Pawel Pawlinski, Vern Paxson, Paul Vixie and Adam Ziaja.

REFERENCES

[1] B. Aboba and S. Cheshire. 2002. *Dynamic Host Configuration Protocol (DHCP) Domain Search Option.* RFC 3397. RFC Editor.

[2] B. Aboba, D. Thaler, and L. Esibov. 2007. *Link-local Multicast Name Resolution (LLMNR).* RFC 4795. RFC Editor.

[3] Richard John Matthew Agar. 2010. *The domain name system (DNS): Security challenges and improvements.* Technical Report. Tech. rep., Royal Holloway, University of London.

[4] Steve Alexander and Ralph Droms. 1997. *DHCP Options and BOOTP Vendor Extensions.* RFC 2132. RFC Editor.

[5] C Alonso. 2013. Fear the Evil FOCA attacking internet connections with IPv6.

[6] Fabio Assolini and Andrey Makhnutin. 2013. PAC – the Problem Auto Config. https://securelist.com/pac-the-problem-auto-config/57891/

[7] K. R. Atul and K. P. Jevitha. 2017. Prevention of PAC File Based Attack Using DHCP Snooping. In *Security in Computing and Communications*, Sabu M. Thampi, Gregorio Martínez Pérez, Carlos Becker Westphall, Jiankun Hu, Chun I. Fan, and Félix Gómez Mármol (Eds.). Springer Singapore, Singapore, 195–204.

[8] Floser Bacurio, Rommel Joven, and Roland Dela Paz. 2016. Over 100,000 South Korean Users Affected by BlackMoon Campaign. https://www.fortinet.com/blog/threat-research/over-100-000-south-korean-users-affected-by-blackmoon-campaign

[9] Daniel Josef Bem. 1992. NASK—Research and academic computer network in Poland. *Computer networks and ISDN systems* 25, 4-5 (1992), 431–437.

[10] Stephen Breen. 2016. Hot Potato – Windows Privilege Escalation. https://foxglovesecurity.com/2016/01/16/hot-potato/

[11] Bugzilla. 2016. https://bugzilla.mozilla.org/show_bug.cgi?id=1255474

[12] US CERT. 2009. Alert (TA09-069A). https://us-cert.cisa.gov/ncas/alerts/TA09-069A

[13] US CERT. 2012. Alert (TA12-318A). https://us-cert.cisa.gov/ncas/alerts/TA12-318A

[14] US Cert. 2016. WPAD Name Collision Vulnerability. https://us-cert.cisa.gov/ncas/alerts/TA16-144A

[15] Ranveer Chandra, Venkata N. Padmanabhan, and Ming Zhang. 2006. WiFiProfiler: Cooperative Diagnosis in Wireless LANs. In *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services* (Uppsala, Sweden) *(MobiSys '06).* Association for Computing Machinery, New York, NY, USA, 205–219. https://doi.org/10.1145/1134680.1134702

[16] Alex Chapman and Paul Stone. 2016. Toxic Proxies: Bypassing HTTPS & VPNs to pwn your online identity. DEF CON. https://doi.org/10.5446/36312 $Lastaccessed : 09Jul$2021.

[17] Qi Alfred Chen, Eric Osterweil, Matthew Thomas, and Z. Morley Mao. 2016. MitM Attack by Name Collision: Cause Analysis and Vulnerability Assessment in the New gTLD Era. In *2016 IEEE Symposium on Security and Privacy (SP).* IEEE, 675–690. https://doi.org/10.1109/SP.2016.46

[18] Qi Alfred Chen, Matthew Thomas, Eric Osterweil, Yulong Cao, Jie You, and Z Morley Mao. 2017. 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.* ACM, 941–956.

[19] Shuo Chen, Ziqing Mao, Yi-Min Wang, and Ming Zhang. 2009. Pretty-bad-proxy: An overlooked adversary in browsers' https deployments. In *2009 30th IEEE Symposium on Security and Privacy.* IEEE, IEEE, 347–359.

[20] Bugs Chromium. 2018. 899126 - chromium - An open-source project to help move the web forward. - Monorail. https://bugs.chromium.org/p/chromium/issues/detail?id=899126

[21] Shawn Evans. 2020. Responder: Beyond WPAD • NopSec. https://www.nopsec.com/responder-beyond-wpad/

[22] Leonid Evdokimov. 2015. Can Web Proxy Autodiscovery leak HTTPS URLs? https://security.stackexchange.com/questions/87499/can-web-proxy-autodiscovery-leak-https-urls

[23] Dhia Farrah and Marc Dacier. 2021. Zero Conf Protocols and their numerous Man in the Middle (MITM) Attacks. In *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, IEEE, 410–421.

[24] Max Fillinger and Marc Stevens. 2015. Reverse-engineering of the cryptanalytic attack used in the flame super-malware. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 586–611.

[25] Ivan Fratric, Thomas Dullien, James Forshaw, and Steven Vittitoe. 2017. https://googleprojectzero.blogspot.com/2017/12/apacolypse-now-exploiting-windows-10-in_18.html

[26] Paul Gauthier, Josh Cohen, Martin Dunsmuir, and Charles E. Perkins. 1999. *Web Proxy Auto-Discovery Protocol.* Internet-Draft draft-ietf-wrec-wpad-01. IETF Secretariat. https://www.ietf.org/archive/id/draft-ietf-wrec-wpad-01.txt https://www.ietf.org/archive/id/draft-ietf-wrec-wpad-01.txt.

[27] Eric Glass. [n. d.]. The NTLM Authentication Protocol and Security Support Provider. https://curl.se/rfc/ntlm.html

[28] Nicolas Golubovic. 2016. Attacking browser extensions. *Ruhr-Universitat Bochum* 3 (2016).

[29] Maxime Goncharov. 2016. badWPAD - The Lasting Menace of a Bad Protocol. https://www.trendmicro.com/vinfo/pl/security/news/vulnerabilities-and-exploits/badwpad-menace-of-a-bad-protocol

[30] André Ricardo A Grégio, Dario Simões Fernandes, Vitor Monte Afonso, Paulo Lício de Geus, Victor Furuse Martins, and Mario Jino. 2013. An empirical analysis of malicious internet banking software behavior. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 1830–1835.

[31] E. Guttman, C. Perkins, and J. Kempf. 1999. *Service Templates and Service: Schemes.* RFC 2609. RFC Editor.

[32] Erik Guttman, Charles Perkins, John Veizades, and Michael Day. 1999. *Service Location Protocol, Version 2.* RFC 2608. RFC Editor. http://www.rfc-editor.org/rfc/rfc2608.txt http://www.rfc-editor.org/rfc/rfc2608.txt.

[33] Richard Habeeb. 2014. IPv6 SLAAC MITM Attack Process and Mitigation. (2014).

[34] Dick Hardt et al. 2012. The OAuth 2.0 authorization framework.

[35] Ben Hawkes. 2019. Project Zero: Five Years of 'Make 0Day Hard'. https://www.blackhat.com/us-19/briefings/schedule/#project-zero-five-years-of-make-day-hard-15900

[36] Peter Hayes. [n. d.]. Browser Support. https://findproxyforurl.com/browser-support/. Accessed: 2021-03-30.

[37] Lin Shung Huang, Alex Rice, Erling Ellingsen, and Collin Jackson. 2014. Analyzing forged SSL certificates in the wild. In *2014 IEEE Symposium on Security and Privacy*. IEEE, IEEE, 83–97.

[38] ICANN. 2013. New Generic Top-Level Domains. https://newgtlds.icann.org/en/about/program

[39] NetBIOS Working Group in the Defense Advanced Research Projects Agency. 1987. *Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods.* STD 19. RFC Editor.

[40] Keisuke Ishibashi, Tsuyoshi Toyono, Katsuyasu Toyama, Masahiro Ishino, Haruhiko Ohshima, and Ichiro Mizukoshi. 2005. Detecting mass-mailing worm infected hosts by mining DNS traffic data. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data.* 159–164.

[41] Fox IT. 2017. Relaying credentials everywhere with ntlmrelayx. https://blog.fox-it.com/2017/05/09/relaying-credentials-everywhere-with-ntlmrelayx/

[42] K. Jaganathan, L. Zhu, and J. Brezak. 2006. *SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows.* RFC 4559. RFC Editor.

[43] Mohamed Amine Kerrich, Adnane Addaim, and Loubna Damej. 2019. Proposed Solution for HID Fileless Ransomware Using Machine Learning. In *International Conference on Advanced Communication Systems and Information Security*. Springer, 180–192.

[44] Amit Klein and Itzik Kotler. 2016. Crippling HTTPS with unholy PAC. https://www.blackhat.com/us-16/briefings/schedule/#crippling-https-with-unholy-pac-3778

[45] Mikael Kulberg. 2018. . https://www.cybersecobservatory.com/wp-content/uploads/2018/04/spring-2018-state-of-the-internet-security-report.pdf

[46] Craig Laprade, Benjamin Bowman, and H Howie Huang. 2020. PicoDomain: a compact high-fidelity cybersecurity dataset. *arXiv preprint arXiv:2008.09192* (2020).

[47] Dan Li, Chaoge Liu, Xu Cui, and Xiang Cui. 2013. POSTER: Sniffing and Propagating Malwares through WPAD Deception in LANs *(CCS '13)*. Association for Computing Machinery, New York, NY, USA, 1437–1440. https://doi.org/10.1145/2508859.2512520

[48] LilyPond. [n. d.]. LilyPond. https://lilypond.org. Accessed: 2022-07-21.

[49] Elsevier Ltd. 2010. PAC attack redirects browsers to malicious sites using proxy hack. *Network Security* 2010, 4 (2010), 2–20. https://doi.org/10.1016/S1353-4858(10)70041-6

[50] Debian Manpages. [n. d.]. Package: sensible-utils (0.0.17 and others). https://packages.debian.org/en/sid/sensible-utils. Accessed: 2022-07-21.

[51] Debian Manpages. [n. d.]. sensible-browser - sensible-utils - Debian Manpages. https://manpages.debian.org/stretch/sensible-utils/sensible-browser.1.en.html. Accessed: 2022-07-21.

[52] Microsoft. 1999. Microsoft Security Bulletin MS99-054 - Critical. https://docs.microsoft.com/en-us/security-updates/securitybulletins/1999/ms99-054

[53] Microsoft. 2007. Microsoft Security Advisory 945713. https://docs.microsoft.com/en-us/security-updates/securityadvisories/2007/945713

[54] Microsoft. 2009. https://msrc-blog.microsoft.com/tag/ms09-008/

[55] Microsoft. 2009. Microsoft Security Advisory 971888. https://docs.microsoft.com/en-us/security-updates/SecurityAdvisories/2009/971888?redirectedfrom=MSDN

[56] Microsoft. 2012. Microsoft Security Bulletin MS12-074 - Critical. https://docs.microsoft.com/en-us/security-updates/securitybulletins/2012/ms12-074

[57] Microsoft. 2017. Microsoft Security Bulletin MS16-063 - Critical. https://docs.microsoft.com/en-us/security-updates/securitybulletins/2016/ms16-063

[58] Microsoft. 2017. Microsoft Security Bulletin MS16-077 - Important. https://docs.microsoft.com/en-us/security-updates/securitybulletins/2016/ms16-077

[59] MITRE. 1999. CVE-1999-0858. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0858

[60] MITRE. 2007. CVE-2007-1692. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-1692

[61] MITRE. 2007. CVE-2007-5355. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5355

[62] MITRE. 2009. CVE-2009-0093. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0093

[63] MITRE. 2009. CVE-2009-0094. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0094

[64] MITRE. 2009. CVE-2009-3372. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3372

[65] MITRE. 2012. CVE-2012-2915. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2915

[66] MITRE. 2012. CVE-2012-4504. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4504

[67] MITRE. 2012. CVE-2012-4505. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4505

[68] MITRE. 2012. CVE-2012-4776. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4776

[69] MITRE. 2012. CVE-2012-5580. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-5580

[70] MITRE. 2016. CVE-2016-3213. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-3213

[71] MITRE. 2016. CVE-2016-3236. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-3236

[72] MITRE. 2017. CVE-2017-17512. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17512

[73] MITRE. 2017. CVE-2017-17523. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17523

[74] MITRE. 2017. CVE-2017-5384. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5384

[75] MITRE. 2017. CVE-2017-6410. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6410

[76] MITRE. 2018. CVE-2018-10992. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-10992

[77] MITRE. 2018. CVE-2018-18358. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-18358

[78] MITRE. 2018. CVE-2018-18506. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-18506

[79] MITRE. 2019. CVE-2019-8454. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-8454

[80] MITRE. 2020. CVE-2020-26154. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-26154

[81] MITRE. 2021. CVE-2021-0393. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-0393

[82] David Moher, Alessandro Liberati, Jennifer Tetzlaff, Douglas G Altman, Prisma Group, et al. 2009. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. *PLoS medicine* 6, 7 (2009), e1000097.

[83] Dirk-Jan Mollema. 2017. MITM6: compromising IPv4 networks via IPv6 - Fox-IT. https://www.fox-it.com/en/news/blog/mitm6-compromising-ipv4-networks-via-ipv6/

[84] HD Moore. 2010. Tactical Exploitation. *Course Slides], Black Hat USA* (2010).

[85] Mozilla. 2017. Security vulnerabilities fixed in Firefox 51. https://www.mozilla.org/en-US/security/advisories/mfsa2017-01/

[86] Eric M Osterweil, Danny R McPherson, Matthew A Thomas, and Qi Alfred Chen. 2020. Detecting and remediating highly vulnerable domain names using passive DNS measurements. US Patent 10,652,271.

[87] Andreas Pashalidis. 2003. A cautionary note on automatic proxy configuration. In *IASTED International Conference on Communication, Network, and Information Security, CNIS 2003, New York, USA, December 10-12, 2003, Proceedings*. Citeseer, 153–158.

[88] Samuel Patton, William Yurcik, and David Doss. 2001. An Achilles' heel in signature-based IDS: Squealing false positives in SNORT. In *Proceedings of RAID*, Vol. 2001. Citeseer.

[89] Vern Paxson. 1999. Bro: A system for detecting network intruders in real-time. *Computer networks* 31, 23-24 (1999), 2435–2463.

[90] Zeek project. [n. d.]. The Zeek network intrusion detection monitor. https://zeek.org (Last accessed: 09 Jul 2021.

[91] Hosnieh Rafiee. 2015. *Multicast DNS (mDNS) Threat Model and Security Consideration.* Internet-Draft draft-rafiee-dnssd-mdns-threatmodel-03. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-rafiee-dnssd-mdns-threatmodel-03.txt http://www.ietf.org/internet-drafts/draft-rafiee-dnssd-mdns-threatmodel-03.txt.
[92] Farsight Security. 2021. Farsight DNSDB API Documentation. https://docs.dnsdb.info/
[93] Stern Security. 2013. Local Network Attacks: LLMNR and NBT-NS Poisoning. https://www.sternsecurity.com/blog/local-network-attacks-llmnr-and-nbt-ns-poisoning/
[94] J. Sermersheim. 2006. *Lightweight Directory Access Protocol (LDAP): The Protocol.* RFC 4511. RFC Editor. http://www.rfc-editor.org/rfc/rfc4511.txt http://www.rfc-editor.org/rfc/rfc4511.txt.
[95] Farhan Siddiqui, Sherali Zeadally, Thabet Kacem, and Scott Fowler. 2012. Zero configuration networking: Implementation, performance, and security. *Computers & electrical engineering* 38, 5 (2012), 1129–1145.
[96] F. Templin, T. Gleeson, M. Talwar, and D. Thaler. 2005. *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP).* RFC 4214. RFC Editor.
[97] Niels Teusink. 2008. Hacking random clients using WPAD. http://blog.teusink.net/2008/11/about-two-weeks-ago-i-registered-wpad.html
[98] Verisign. 2016. White Paper:Enterprise Remediation for WPAD Name Collision Vulnerability. (May 2016). https://www.verisign.com/assets/Enterprise_Remediation_for_WPAD_Name_Collision_Vulnerability.pdf
[99] Venu Vissamsetty, Muthukumar Lakshmanan, Sreenivasa Sudheendra Penupolu, and Ankur Rungta. 2019. Detecting man-in-the-middle attacks. US Patent 10,250,636.
[100] Jianing Wang and Junyu Zhou. 2018. NTLM Relay Is Dead, Long Live NTLM Relay. https://conference.hitb.org/hitbsecconf2018dxb/sessions/ntlm-relay-is-dead-long-live-ntlm-relay/
[101] Yang Yu. 2016. BadTunnel: How Do I Get Big Brother Power? https://www.blackhat.com/us-16/briefings/schedule/#badtunnel-how-do-i-get-big-brother-power-3915
[102] Yang Yu. 2016. BadTunnel: NetBIOS Name Service spoofing over the Internet. *Tencents Xuanwu Lab* (2016).
[103] Google Project Zero. 2014. Issue 222: Windows: Local WebDAV NTLM Reflection Elevation of Privilege. https://bugs.chromium.org/p/project-zero/issues/detail?id=222&redir=1
[104] Sami Zhioua. 2013. The middle east under malware attack dissecting cyber weapons. In *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*. IEEE, IEEE, 11–16.
[105] Adam Ziaja. 2019. https://blog.redteam.pl/search?q=wpadblocking.com

A APPENDIX A

Table 3 lists all the acronyms mentioned in the paper and their definitions.

Table 3. List of acronyms and their definitions.

Acronym	Definition
ALG	Application Level Gateway
CAS	Code Access Security
CURL	Configuration URL
CVE	Common Vulnerabilities and Exposures
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FQDN	Fully qualified domain name
HITB	Hack In The Box
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HVD	Highly vulnerable domains
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical and Electronics Engineers
ISP	Internet Service Provider
LAN	Local area network
LLMNR	Link-Local Multicast Name Resolution
MDNS	Multicast DNS
MITM	Man In The Middle
NASK	Naukowa i Akademicka Sieć Komputerowa
NBNS	NetBIOS Name Service
NG-FW	Next-Generation Firewall
NTLM	NT (New Technology) LAN Manager
OS	Operating System
PAC	Proxy Auto-Configuration
PBP	Pretty-Bad-Proxy
RFC	Request for Comments
SLP	Service Location Protocol
SMB	Server Message Block
SRV	Service Record
SSL	Secure Sockets Layer
TLD	Top-level domain
TLS	Transport Layer Security
TXT	Text Record
URL	Uniform Resource Locator
VPN	Virtual private network
WAF	Web application firewall
WINS	Windows Internet Name Service
WPAD	Web Proxy Auto-Discovery