Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 7, No. 1, April 2021, pp. 18-29
ISSN: 2338-3070, DOI: 10.26555/jiteki.v7i1.18512

Measuring Unsupported Applications in Indonesia Popular Websites

Pascal Alfadian Nugroho [1], Hizkia Steven [2]
[1] Universitas Katolik Parahyangan, Jl. Ciumbuleuit No. 94, Bandung 40252, Indonesia
[2] PT DNArtworks Komunikasi Visual, Ruko Graha Boulevard Blok A No. 9, Tangerang Selatan 15810, Indonesia

ARTICLE INFO

Article history:

Received November 11, 2020
Revised March 05, 2021
Accepted March 28, 2021

Keywords:

Measurement;
Security;
Websites

ABSTRACT

A security vulnerability exists in unsupported systems, and using applications supported by their maintainer help to reduce attacks based on such vulnerabilities. However, website administrators may ignore this exercise due to various reasons. This research measures the top 1,500 websites in Indonesia on how much of them are using supported applications to prevent such attacks, based on the application version number. The measurement is performed automatically using the Wappalyzer tool. From such measurement, we found that most of the applications detected do not contain version information (70%) or invalid version number (11%). We also found that more than half of the websites measured contain at least one unsupported application. In terms of the applications used, we found that many Nginx users worryingly do not keep their server version updated, while Apache and WordPress did a good job in keeping their users using the most recent version. This study highlights the need for website administrators to have their applications up to date to the supported versions, as well as for application developers to promote application updates to their users.

Pascal Alfadian Nugroho
Universitas Katolik Parahyangan, Jl. Ciumbuleuit No. 94, Bandung 40252, Indonesia
Email: pascal@unpar.ac.id

1. INTRODUCTION

In the field of computer security, there are various types of methods to attack a particular vulnerability in a working system. One type of attack is called the "zero-day" attack, i.e., an attack on vulnerability that is exploited before the vulnerability is publicly disclosed [1]. However, after the vulnerability is disclosed, it does not mean that all involved systems are automatically secured. Maintainers of an affected system need to patch their system so that the vulnerability that allows such exploitation is patched. Those vulnerable systems are still prone to an attack called "follow-on attacks," i.e., attacks that happen after the vulnerability is disclosed but before patch deployment is completed [1].

Ideally, most if not all affected systems are patched after the vulnerability is disclosed and a patch is available. However, a user may delay or unwilling to deploy the patch to their machine. This could be due to the fact that not all vulnerabilities are exploited. In fact, reference [2] examined the relationship between severity level of vulnerability and the exploitation from open-source data and found that both are not always related. It is better though to keep all vulnerabilities patched because before such vulnerabilities are patched, there is no guarantee that it is not exploited without the maintainer knowing it. For example, the Heartbleed vulnerability allows an attacker to steal private information without leaving a trace to the system administrator [3].

Most applications provide an easy upgrade feature, where upgrading to a more recent version can be done simply with a click of a button or some automated commands (instead of removing and re-installing the application, for example). In most cases, the more recent version of an application is, the more known vulnerabilities are fixed. However, studies found that even with such ease, many users are unwilling to upgrade to a more recent version of applications. Reference [4] studied such behavior in four client-side applications

(i.e., Chrome, Firefox, Flash Player, and Thunderbird) and found several factors that affect user behavior in deploying such patches:

1. Users' willingness to patch does not seem to depend on the type of improvements in new releases.
2. Silent updates lead to shorter windows of vulnerability for end-hosts.
3. Even with silent updates, the majority of hosts have long windows of vulnerability.
4. Many hosts have long windows of susceptibility to known exploits.

Reference [5] also noted that non-technical users might skip security patches due to bad experiences from previous updates. Reference [6] stated that patch rates could be raised with policy interventions, such as educating the public about its benefit, built-in software report of current patches applied, letting know that others have made patches, transparency about the patch itself, automatic download and application of patches, bandwidth subsidy for patch download, and separating security patch with license checking routine.

On the other hand, the availability of patches for application vulnerabilities largely depends on the application maintainer's willingness to release patches for such vulnerabilities. Due to limited resources, most maintainers only release patches for selectively *supported versions*, usually the most recent ones. This is usually stated in their support policy. In view of this, users of such applications should proactively ensure that their version of the application is kept above the minimum supported version.

This research proposes a method to perform scalable measurement of websites on how much they are using unsupported applications. For example, let us take one website *okezone.com*, which at the time of this research, we found it's built on top of Nginx web server (version unknown), Bootstrap UI framework (version 3.3.7), and jQuery JavaScript library (version 3.1.1), among other applications[1]. At the same time, the maintainer of jQuery claimed that they supported version 3 or above, and the maintainer of Bootstrap claimed that they supported version 4 or above. Therefore, from such an example, we know that the website *okezone.com* contained two applications that were still supported, while for the other one (Nginx), we could not decide whether they are using supported or non-supported version of the application. This research is limited to detecting the version of applications used and not the other types of vulnerability patching. For example, the Debian OpenSSL vulnerability required system administrators to replace their server key pair with a newly generated one in addition to updating the application, i.e., reissuing SSL certificates [7].

2. RELATED WORKS

Reference [8] surveyed applications used by 40 websites of public and private universities, both local and international. Though the samples used were relatively small, and the research did not specifically measure the impact on security, it gave us a rough idea of the applications used for websites accessed by Indonesian visitors, i.e., Google Analytics, JavaScript, PHP, Joomla, Apache web server, and CentOS. Through our research was done eight years later, most applications are still frequently used. The exception is the CentOS operating system, which usage is now lower than Ubuntu (26 websites, compared to 94 websites using Ubuntu). Our research is done on a large scale and more focused on the security side.

Reference [9] checked 1,669 web pages crawled from alexa.com, Wikipedia.com, moz.com/top500, sapo.pt, and pplware.sapo.pt (due to their "high number of outgoing links to many different topics"), where each page has a different domain. The pages were evaluated using two tools: QualWeb (homemade) to measure the accessibility score and Wappalyzer[1] to identify applications used. The research then focused on the correlation between the technologies and the accessibility score instead of security.

Reference [10] checked various Colombian websites against XSS, CSRF, and SQL Injection vulnerability. The team first identified "main Colombian economic sectors" from an article. Based on those sectors, the team looked for big companies that do their business in such sectors in a public presentation slide. In addition, the list was complemented with Alexa's most visited websites in Colombia. Finally, some websites developed by Colombian software companies were handpicked into the list. The total number of websites for such research was 130 websites. The team used Wappalyzer to collect information mostly for the classification of such websites (Name, Economic Sector, Programming Language, CMS, and Web Framework). The real detection of vulnerability was performed using the Acunetix Trial Edition tool. Our research is similar in the sense that we also look for vulnerabilities. However, our research is more focused on the vulnerability that is easier to detect and fix. Hence it should have been fixed and regularly maintained.

Reference [11] extracted 1,000 educational website information using the Wappalyzer tool and used the result as the fingerprint. This fingerprint was then matched to other fingerprints in the database using the "euclidian distance KMN proximity algorithm." Complemented with the Stanford University's public web application vulnerability scanning plug-in library, their research concluded 6% performance of the system.

[1] This is based on identification by Wappalyzer tool, as discussed later in "Research Method" section

However, the paper itself contains too many unclear statements, such as how to conclude that 6% improvement, the list of 1,000 websites, as well as the exact algorithm.

Reference [12] performed a survey to 548 governmental websites in Indonesia to see the technologies used, in particular: web server, web programming, CSS, content management system, web framework, and web 3.0. Reference [13] reviewed 5.6 million websites within the time span of 18 months from HTTPArchive, to see whether they are using outdated software. Their work is similar to our research and done on a much larger scale (our research was done on 1,500 websites and performed on one day). However, there are several differences between the two. Reference [13] looked for software that does not use the latest minor or patch version as extracted from each application GitHub repository, while our research focuses more on the versions supported stated in the maintainer's website. We also used on-the-fly Wappalyzer detection, while reference [13] relied on Wapplayzer information detected earlier by HTTPArchive.

3. RESEARCH METHOD

Our methodology is composed of four steps, and each will be explained in the following subsections.

3.1. Collecting list of popular websites

First, we collect the list of popular websites for a specified country. We do this by retrieving the list from Alexa Top Sites [14] for a given country. It is also possible to perform the collection programmatically, using an API with the same name from AWS Marketplace [15]. The output of this step is a list of domain names in tabular format (we use Comma Separated Values) in Table 1. First ten results of Alexa Top Sites show an example of the result, i.e., the ten most popular websites found for Indonesia, according to Alexa Top Sites. Though Alexa Top Site provides additional information for each domain, only the domain name is considered in the next steps.

Table 1. First ten results of Alexa Top Sites

rank	global_ rank	domain name	reach_ per million	pageviews_ per million	pageviews_ peruser
1	25	okezone.com	4,125,963.40	215,723.90	1.45
2	1	google.com	822,200.00	343,400.00	11.60
3	45	tribunnews.com	1,704,664.70	165,582.76	2.70
4	2	youtube.com	476,200.00	113,200.00	6.60
5	75	grid.id	877,022.26	94,402.62	2.99
6	101	detik.com	642,803.89	78,497.49	3.39
7	105	kompas.com	692,848.09	59,379.64	2.38
8	129	sindonews.com	556,134.02	55,960.03	2.79
9	147	tokopedia.com	488,774.40	54,046.56	3.07
10	156	liputan6.com	426,724.20	57,911.40	3.77

3.2. Identify applications used

For each website, we try to identify the applications and the version number used. This is also possible with the help of a 3rd party library Wappalyzer [16]. Wappalyzer performs detection by sending one or more HTTP requests to the target URL and look for a fingerprint to identify the application and version number. The "application" here is defined as any program that helps builds the website and trackable by the Wappalyzer command-line tool[2], be it the operating system, web server, or the frontend library. The output of this step is detailed information of applications used for each website, one file per website, as shown in Fig. 1. For example, *okezone.com* was detected using at least two applications: comScore (version unknown) and Bootstrap (version 3.3.7).

Since Wappalyzer expects proper URI as the input and we only have domain names from the previous step, we prepended *http://* into the domain names before feeding them to Wappalyzer. Should the website support HTTPS, and it properly redirects HTTP requests to HTTPS, Wappalyzer will be smart enough to follow the redirect. For example, if accessing *http://okezone.com* redirects to *https://okezone.com*, then Wappalyzer will detect the applications in *https://okezone.com* instead. Note that while [17] argued that the HTTPS version of some websites is not equivalent to their HTTP counterpart, we chose not to consider such fact in this research since we were looking more at the applications that built the website as opposed to the content of it.

[2] The Wappalyzer website uses the term "technology", while the response of their command-line tool uses the term "application". In this paper, we chose the latter term for consistency.

In many instances, Wappalyzer may not be able to detect the version number or even the complete list of applications used in a website. This can be due to limited information that can be gathered from the website or the webmaster deliberately hide such information for security reason.


```
  "urls": {
    "http://okezone.com/": {
      "status": 301
    },
    ...
    "https://www.okezone.com/": {
      "status": 200
    }
  },
  "applications": [
    {
      "name": "comScore",
      "confidence": 100,
      "version": "",
      "icon": "comScore.png",
      "website": "http://comscore.com",
      "cpe": null,
      "categories": { "10": "Analytics" }
    },
    ...
    {
      "name": "Bootstrap",
      "confidence": 100,
      "version": "3.3.7",
      "icon": "Bootstrap.png",
      "website": "https://getbootstrap.com",
      "cpe": "cpe:/a:getbootstrap:bootstrap",
      "categories": { "66": "UI frameworks" }
    }
  ],
  "meta": {
    "language": "id-ID"
  }
}
```


Fig. 1. Example of detailed application information from Wappalyzer

3.3. Group by application names and retrieve the currently supported version

Later we will determine whether an application in a given website is being supported or not by its maintainer. To do this, we need to manually look in the official documentation of each application, what is the currently supported version (at the time of this research). Looking for a supported version for all applications would be very time-consuming. Hence we need to filter out the applications.

We do this by programmatically investigating each file to identify further the applications used. From each result in the previous step, we count the usage of each application, as well as whether the version number can be retrieved or not, and have the recap of it as shown in Table 2. Column *num_unversioned* records the number of websites (out of the initial list) where Wappalyzer failed to retrieve the version number for that given application. Similarly, column *num_versioned* records the number of websites where Wappalyzer succeeded in retrieving the version number.

We manually look at the supported versions[3] of an application only for applications with *num_versioned* > 0. For the rest of the list, it will not be beneficial to have their supported version, as there will be no version to compare against. We define "supported" if such version satisfies one of the following conditions:

1. **They were explicitly stated on their website**. For example, PHP has a dedicated page that lists down which versions are supported and when will be their end-of-life [18].
2. **Document for such version is still available**. For example, ZURB Foundation's site documentation page contains documentation for the latest version 6 as well as 5 [19]. Therefore, the minimum version is 5. The rationale behind this is that if there is no public documentation for a certain version, the web developer (i.e., user of such application) will not have official reference for troubleshooting.
3. **Implied**. For example, based on its product lifecycle page [20], Microsoft states that IIS version 7.5 is a built-in component of Windows 7 and Windows Server 2008 R2. Microsoft also states that both Windows

[3] We named this column *min_supported_version* before we knew that some applications actually have a more complex support policy rather than just minimum version. For example, Varnish supported versions were either = 6.0 or ≥ 6.3 [28]. In other words, version 6.1 and 6.2 are no longer supported.

Vista and Windows Server 2008 are no longer supported. Meanwhile, the next version, 8.0, is a built-in component of Windows 8 and Windows Server 2012, and both are still supported. Therefore, it is implied that the minimum supported version for IIS is 8.0.

3.4. Compare version between current usage and supported versions

Finally, we revisit each website and each of their application they are using Wappalyzer and compare the version information against the supported versions for such applications. The comparison algorithm is described in Fig. 2, which classify each application in each website into one of the following:

1. **Not-versioned** means that the application detected by Wappalyzer does not contain version information which we could not compare against. For example, the application comScore for *okezone.com* in Fig. 1 contains an empty "version" attribute in the Wappalyzer result.
2. **Non-conclusive** can mean one of these two:
   a. we were able to retrieve the version number used in the application, but we could not determine whether such version is still supported or not by the maintainer. For example, on the website *eramuslim.com*, Wappalyzer reported that it is using jQuery version 1581914217, whereas the supported version for jQuery is >= 3. We decided that both numbers are incomparable since the former looked more like a timestamp rather than a carefully crafted version number.
   b. The supported versions for a given application are unknown. For instance, jQuery UI simply does not have a support policy on their website.
3. **Unsupported** means that we were able to conclude that the application used was using the version number that is not supported by the maintainer. For example, *okezone.com* used version 3.3.7, which was not supported because the supported version for Bootstrap was version >= 4. We also found that one particular application (i.e., YUI) has no more versions supported [21].
4. **Supported** means that we were able to conclude that the application used was using the version number still supported by the maintainer. For example, *okezone.com* used Bootstrap jQuery 3.1.1, which was well supported by, i.e., version >= 3.

Furthermore, the algorithm also does several extra handlings due to the data we found:

1. Version number containing "alpha" or "beta" is simply classified as "non-supported." For example, Bootstrap once had version 4.0.0-alpha and 4.0.0-beta, and they are still in use by some websites.
2. Other consecutive non-numeric characters at the end of the version number are removed. For example, OpenSSL 1.0.2k is regarded as version 1.0.2 since the letter is used as a bug and security fixes release [22].
3. Version number as a codename is manually renamed to the proper numbering system. For example, Debian's "squeeze" is equivalent to version 6.0 [23].

Table 2. First ten results of the applications recap

name	num_sites	avg_connfidence	num_unversioned	num_versioned	website	min_supported_version
jQuery	1011	99.70	14	997	https://jquery.com	>= 3
Bootstrap	430	99.30	88	342	https://getbootstrap.com	>= 4
JQuery Migrate	298	99.66	31	267	https://github.com/jquery/jquery-migrate	?
PHP	591	99.83	346	245	http://php.net	>= 7.2
Font Awesome	400	99.50	160	240	https://fontawesome.com/	>= 5
JQuery UI	176	99.43	7	169	http://jqueryui.com	?
WordPress	346	100.00	181	165	https://wordpress.org	>= 5.4.2
Underscore .js	124	24.19	2	122	http://underscorejs.org	?
Lodash	125	59.20	3	122	http://www.lodash.com	?

4. RESULTS AND DISCUSSION

We performed both domain name collection and application detection on July 9, 2020[4], covering 1,500 most popular websites for Indonesia. Due to Work from Home policy, all data collection was performed from the researcher's home using the IndiHome[5] internet service provider.

4.1. Infinite Timeout Problem

For several websites (e.g., *bppt.go.id*, *free-power-point-templates.com*, *garuda-indonesia.com,* and *jatimprov.go.id*), Wappalyzer detection took too much time to finish, despite the 5 seconds timeout configured in Wappalyzer. Since detections were performed synchronously, this made the script stops responding and unable to continue detecting applications for the next websites. In such a situation, we manually terminated the script execution and instead performed detection by manually sending such requests to the Wappalyzer REST API, which uses the same algorithm but performed by the Wappalyzer server. Note that we did not repeat this for detection that gracefully timed out in 5 seconds because:

1. The response from REST API is not from instant detection, but one saved from detection in the past. For example, based on the timestamp information bppt.go.id was detected in May 2020, i.e., two months before detection of our other sites.
2. Emulation of normal conditions is preferred, whereby visitors may encounter temporary or permanent disability to access websites that are accessible by other visitors in different networks. We did retry the timed-out websites several times before resorting to the REST API method.

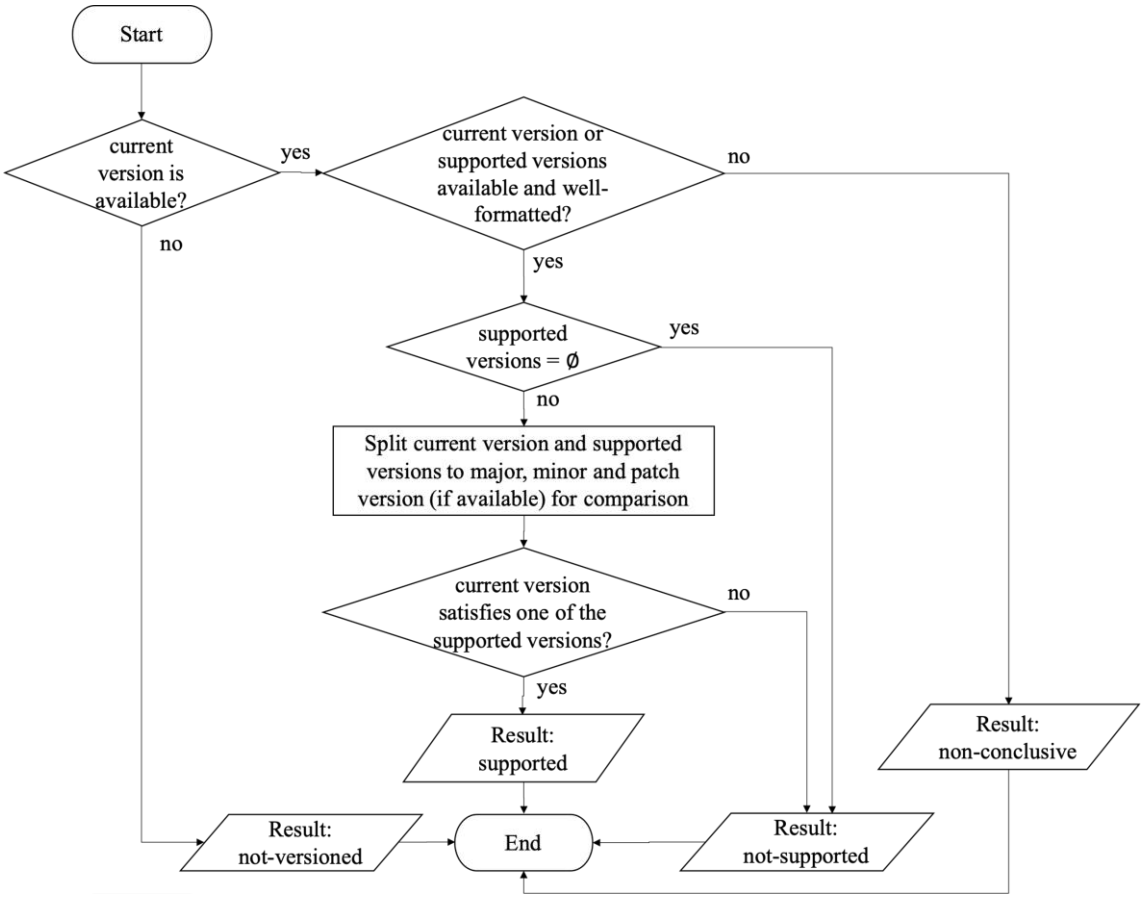


Fig. 2. Algorithm to compare current version versus supported versions

[4] We initially retrieved the sites on 2 March 2020, in the early stage of this research. However, we decided to repeat the retrieval after COVID-19 pandemic situation started, with the hope that it reflects the popular site visited by a more general public since most people stayed at home during such situation.
[5] https://indihome.co.id

4.2. Overall result

From the 1,500 URIs detected by Wappalyzer, we retrieved 1,439 successful identifications, marked with an HTTP status code of 200 (OK). There are several other interesting responses that we retrieved, as described in Table 3. Note that we see status codes 521 and 522, which are not standard HTTP status codes [24]. They are status codes returned by Cloudflare, which is widely known as a Content Delivery Network (CDN) provider. We also found status code 0, which is a predefined status code from Wappalyzer, to inform us that an Unknown Error has occurred.

Do note that the total number of status codes does not equal the number of original URLs listed because Wappalyzer is able to follow redirects. Hence a website may be counted twice or more: one for the final page and one or more for its redirect pages.

We found a total of 12,762 applications from detecting 1,500 websites. As can be observed in Table 4, we could not determine the version number for most of the applications used. As mentioned earlier, there is a probability that the webmaster intentionally hides the version number to prevent attacks (i.e., security by obscurity). Of those where we can detect the version number and compare it against the supported version, more than half were no longer supported by the application maintainer.

Table 3. HTTP statuses of 1,500 websites detected by Wappalyzer

HTTP Status		Count
Code	**Description**	
200	OK	1,439
204	No Content	2
301	Moved Permanently	1,439
302	Found	339
303	See Other	5
307	Temporary Redirect	12
308	Permanent Redirect	11
403	Forbidden	10
404	Not Found	16
500	Internal Server Error	1
503	Service Unavailable	4
521	Web Server is Down	5
522	Connection Timed Out	2
0	Unknown Error	64

Table 4. Overall application count for measurement result

Result	Application count	Percentage
Not-versioned	8,980	70.37
Non-conclusive	1,409	11.04
Unsupported	1,508	11.82
Supported	865	6.78
Total	12,762	100.00

4.3. Supportability per website

Table 5 lists the measurement result for the ten most popular websites. Most of the applications detected are not versioned, but for the detected ones, we were able to find unsupported applications. Furthermore, the unsupported applications used in these top 10 websites are Bootstrap, Font Awesome, jQuery, and PHP.

We also try to put the top 1,500 websites into ten bins of 150 websites each, sorted by rank. We did this to see whether there is a correlation between website rank and its usage of unsupported applications. For each bin, we count the number of websites that use *n* unsupported applications. We focus only on unsupported applications because it allows the possibility for attacks, and it has room for improvements for the webmaster. The result, as shown in Table 6, shows that there is no correlation between website rank and the number of unsupported applications.

Table 5. First ten results of measurement

rank	domain name	not-versioned	non-conclusive	unsupported	supported
1	okezone.com	7	0	1	1
2	google.com	1	0	0	0
3	tribunnews.com	11	2	2	0
4	youtube.com	1	1	0	0
5	grid.id	11	1	2	1
6	detik.com	8	3	0	0
7	kompas.com	10	2	1	0
8	sindonews.com	4	1	1	0
9	tokopedia.com	5	0	0	0
10	liputan6.com	11	1	1	0

Table 6. Number of unsupported applications grouped by website rank

rank	$n = 0$	$n = 1$	$n = 2$	$n = 3$	$n \geq 4$
1-150	56	58	26	9	1
151-300	52	55	29	12	2
301-450	59	43	32	10	6
451-600	56	48	22	21	3
600-750	59	58	22	10	1
751-900	68	44	25	8	5
900-1,051	65	42	30	10	3
1,051-1,200	56	46	34	10	4
1,201-1,350	50	57	31	11	1
1,351-1,500	62	46	29	11	2

4.4. Supportability per application

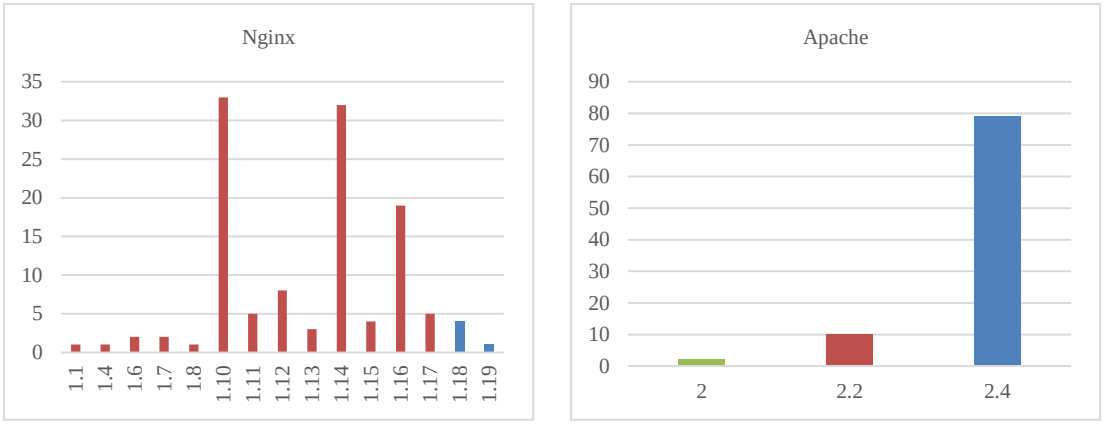
We are also interested to see how good applications are being used by websites in terms of being kept updated to supported versions. We looked for the top 15 applications that were used in the top 1,500 websites and filtered out applications whose version we could not identify in any of the top 1,500 websites. The result is shown in Table 7. Furthermore, we try to examine the version characteristics of some handpicked applications that represent server-side (e.g., Apache and Nginx) and client-side applications (e.g., jQuery and Bootstrap). The result is discussed in the following subsections and the accompanying charts. In each of the chart, we color it as follow: (1) red color represents application versions that are no longer supported by the maintainer, (2) blue color represents application versions still supported, and (3) green represents version number which we can't compare against (non-conclusive).

Table 7. Top applications used

num_sites	name	supported	unsupported	non-conclusive	not-versioned
1,011	jQuery	260	737	0	14
591	PHP	118	127	0	346
478	Nginx	5	116	0	357
430	Bootstrap	114	228	0	88
400	Font Awesome	70	157	13	160
346	WordPress	118	41	6	181
298	jQuery Migrate	0	0	267	31
237	Apache	79	10	2	146

Nginx and Apache

Nginx and Apache are two popular web servers that power various websites. Nginx surprisingly only has their usage supported (version 1.18 or greater) in about 20% of all websites, as shown in Fig. 3 (a). However, Nginx uses the word "legacy" in their website to denote those versions before the "stable" versions. In one definition, legacy is defined as "has been superseded but is difficult to replace because of its wide use" [26]. Moreover, Nginx is often used as a web proxy that handles the HTTP interface to the world, whereas the actual webserver lives behind it. On the other hand, the Apache webserver has a much-simplified versioning mechanism with only two notable versions, 2.2 and 2.4, as shown in Fig. 3 (b). In its bare form, Apache indeed provides only basic features, with additional plugins required for further needs ("mod"). We suggest that the relatively unchanged feature between releases helped them built a more maintainable security factor.



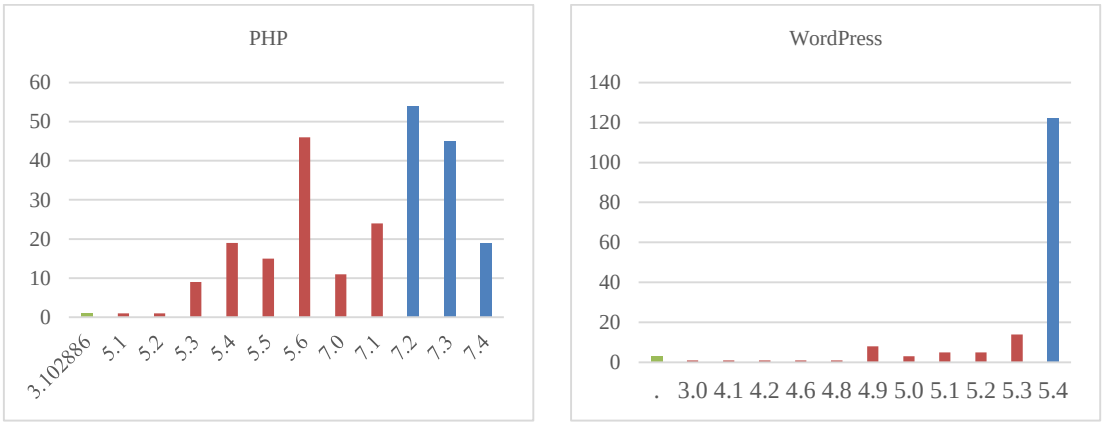
(a) (b)

Fig. 3. Nginx and Apache versions used in websites

PHP & WordPress

PHP, one of the most popular scripting languages that power various websites, also had less than half of its usage in websites supported. From Fig. 4 (a), we believe that the greatest difficulty in using the supported version is to upgrade from version 5.x to another major version 7.x, where most people were stuck at 5.6, the latest of the 5.x series. Do note that PHP does not have version 6 available for the general public.

WordPress is a popular CMS platform that works on top of PHP. Although WordPress has a strict support policy of only the latest version supported [27], the maintainers have successfully pushed its users to have the most recent version, 5.4, i.e., the supported version at the time of this research, as can be seen in Fig. 4 (b). We believe that this is due to the easy one-click upgrade process and high forward compatibility that the developers of WordPress have made to the application.



(a) (b)

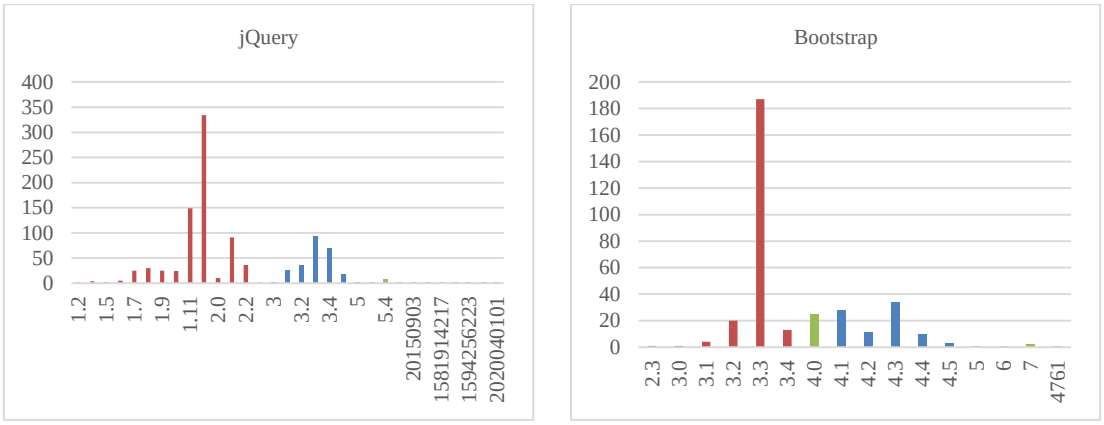
Fig. 4. PHP and WordPress versions used in websites

jQuery, jQuery Migrate and Bootstrap

jQuery is a client-side JavaScript library to help web developers use JavaScript in an easier and readable way. Despite detected as the most used applications in the top sites, it has more than 50% of its usage in the top sites unsupported. We suggest that the low risk associated with using vulnerable jQuery (all versions) reduces the incentive for developers to think more about its security, let alone upgrading to a supported version. As of the end of 2019, jQuery only had seven vulnerabilities recorded in the CVE database, while PHP had 601 vulnerabilities recorded [25]. Note also that jQuery has a helper tool called jQuery Migrate, which helps developers using the latest jQuery library with older jQuery syntax and has been used in 298 out of the 1,500 top sites.

A summary of the jQuery versions used can be seen in Fig. 5 (a). Note that for some versions (e.g., "5"), we manually mark them as "non-conclusive" despite our tool marks them as "supported". This is because the latest stable version of jQuery is 3.5.1. Hence it must be an identification error. It is also interesting to notice that majority of people are still using the 1.x branch. This could be due to the jQuery Migrate tool, which helps developers stay in their initial version instead of upgrading.

Bootstrap is a popular client-side web framework that works on top of jQuery to ease development on the user interface. Fig. 5 (b) shows that most websites are still using Bootstrap version 3.x, with the majority using version 3.3. It seems that the developer skips version 3.4 in favor to version 4.x. Similar to jQuery, we also manually mark some versions as "non-conclusive" despite our tool marked them as "supported", since the version number is out of the range of all Bootstrap versions ever released. Furthermore, version 4.0 is also marked as "non-conclusive", as there are some websites using the alpha or beta version of version 4.0.



(a) (b)

Fig. 5. jQuery and Bootstrap versions used in websites

5. CONCLUSION

We explored the top 1,500 sites according to Alexa rank for Indonesian visitors and identified what types of applications that they are using, along with their version. Those version numbers are compared with the minimum version supported by the maintainer of each application whenever possible. More than half (63%) of the applications used and successfully compared by our script are no longer supported by their maintainer. We also found that different applications have different characteristics on their versioning as well as the usage of supported versions.

Many websites use jQuery of version as far as two major versions behind the currently supported version, though they are helped by the jQuery Migrate library, which enables syntax usage of unsupported version, with the latest library version. Almost half of the PHP users are using unsupported versions, mostly 5.6, the latest minor version of major version 5. Nginx web server only has about 20% of its usage supported, but we believe that it is because Nginx is mostly used as a proxy, hence not much urgency to upgrade to the latest versions. Bootstrap has a very high usage of unsupported version 3.3, but since it is a frontend library, it does not pose too many security risks. WordPress manages to push most of its users to upgrade to the latest supported version, with 74% of websites are using the supported version of WordPress. Lastly, Apache also has most of its usage supported, with only three variations of the version detected.

Such findings highlight the need for website owners to have their applications powering the website kept up to date with the supported versions. As for application maintainers, WordPress and Apache can be considered as a success in urging their users to update their applications to the supported version. Further studies should be performed to identify what is the success factor for having users willingly upgrade to the

supported version, as well as why some users are reluctant to upgrade to the latest version in popular but critical applications like Nginx.

REFERENCES

[1] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," *Proc. ACM Conf. Comput. Commun. Secur.*, no. October 2012, pp. 833–844, 2012. https://doi.org/10.1145/2382196.2382284

[2] B. L. Bullough, A. K. Yanchenko, C. L. Smith, and J. R. Zipkin, "Predicting exploitation of disclosed software vulnerabilities using open-source data," *IWSPA 2017 - Proc. 3rd ACM Int. Work. Secur. Priv. Anal. co-located with CODASPY 2017*, pp. 45–53, 2017. https://doi.org/10.1145/3041008.3041009

[3] L. Zhang *et al.*, "Analysis of SSL certificate reissues and revocations in the wake of heartbleed," *Commun. ACM*, vol. 61, no. 3, pp. 109–116, 2018. https://doi.org/10.1145/3176244

[4] A. Sarabi, Z. Zhu, C. Xiao, M. Liu, and T. Dumitraş, "Patch me if you can: A study on the effects of individual user behavior on the end-host vulnerability state," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10176 LNCS, no. 1, pp. 113–125, 2017. https://doi.org/10.1007/978-3-319-54328-4_9

[5] K. Vaniea, E. Rader, and R. Wash, "Betrayed by updates: How negative experiences affect future security," *Conf. Hum. Factors Comput. Syst. - Proc.*, pp. 2671–2674, 2014. https://doi.org/10.1145/2556288.2557275

[6] D. K. Mulligan and F. B. Schneider, "Doctrine for cybersecurity," *Daedalus*, vol. 140, no. 4, pp. 70–92, 2011. https://doi.org/10.1162/DAED_a_00116

[7] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage, "When private keys are public," p. 15, 2009. https://doi.org/10.1145/1644893.1644896

[8] B. E. Sudarno; Purnama, "Analisis Penggunaan Tools Web Perguruan Tinggi," *Semin. Ris. Unggulan Nas. Inform. dan Komput.*, 2012.

[9] C. Duarte, I. Matos, J. Vicente, A. Salvado, C. M. Duarte, and L. Carriço, "Development technologies impact in Web accessibility," *W4A 2016 - 13th Web All Conf.*, pp. 2–5, 2016. https://doi.org/10.1145/2899475.2899498

[10] E. D. Alvarez, B. D. Correa, and I. F. Arango, "An analysis of XSS, CSRF and SQL injection in colombian software and web site development," *2016 8th Euro Am. Conf. Telemat. Inf. Syst. EATIS 2016*, 2016. https://doi.org/10.1109/EATIS.2016.7520140

[11] H. He, L. Chen, and W. Guo, "Research on Web Application Vulnerability Scanning System based on Fingerprint Feature," vol. 61, no. Mecae, pp. 150–155, 2017. https://doi.org/10.2991/mecae-17.2017.27

[12] N. A. Rakhmawati, S. Harits, D. Hermansyah, and M. A. Furqon, "A Survey of Web Technologies Used in Indonesia Local Governments," *Sisfo*, vol. 07, no. 03, 2018. https://doi.org/10.24089/j.sisfo.2018.05.003

[13] N. Demir, T. Urban, K. Wittek, and N. Pohlmann, "Our (in)Secure Web: Understanding Update Behavior of Websites and Its Impact on Security," 2021. https://doi.org/10.1007/978-3-030-72582-2_5

[14] Alexa Internet Inc, "Alexa - Top sites." https://www.alexa.com/topsites (accessed Mar. 06, 2020).

[15] Amazon Web Services, "AWS Marketplace: Alexa Top Sites." https://aws.amazon.com/marketplace/pp/Amazon-Web-Services-Alexa-Top-Sites/B07QK2XWNV (accessed Mar. 06, 2020).

[16] E. Alias, "Download & Install - Wappalyzer." https://www.wappalyzer.com/download/ (accessed Apr. 06, 2020).

[17] M. T. Paracha, B. Chandrasekaran, D. Choffnes, and D. Levin, "A Deeper Look at Web Content Availability and Consistency over HTTP / S," 2020.

[18] The PHP Group, "PHP: Supported Versions." https://www.php.net/supported-versions.php (accessed Jul. 27, 2020).

[19] ZURB Inc, "Foundation for Sites 6 Docs." https://get.foundation/sites/docs/ (accessed Jul. 27, 2020).

[20] Microsoft, "Search Product and Services Lifecycle Information - Microsoft Lifecycle | Microsoft Docs." https://docs.microsoft.com/en-us/lifecycle/products/ (accessed Jul. 27, 2020).

[21] Yahoo Engineering, "Important Announcement Regarding YUI | Yahoo Engineering." https://yahooeng.tumblr.com/post/96098168666/important-announcement-regarding-yui (accessed Jul. 27, 2020).

[22] OpenSSL Software Foundation, "Release Strategy." https://www.openssl.org/policies/releasestrat.html (accessed Jul. 27, 2020).

[23] Software in The Public Interest, "Debian -- Debian Releases." https://www.debian.org/releases/ (accessed Jul. 27, 2020).

[24] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," *Internet Engineering Task Force*, 2014. https://tools.ietf.org/html/rfc7231 (accessed Feb. 26, 2021). https://doi.org/10.17487/rfc7231

[25] S. Özkan, "CVE security vulnerability database. Security vulnerabilities, exploits, references and more." https://www.cvedetails.com/ (accessed Jul. 24, 2020).

[26] Lexico.com, "Legacy | Definition of Legacy by Oxford Dictionary on Lexico.com also meaning of Legacy." https://www.lexico.com/definition/legacy (accessed Mar. 06, 2020).

[27] Automattic, "Releases | WordPress.org." https://wordpress.org/download/releases/ (accessed Jul. 29, 2020).
[28] P.-H. Kamp, "Releases & Downloads — Varnish HTTP Cache." http://varnish-cache.org/releases/ (accessed Jul. 27, 2020).

BIOGRAPHY OF AUTHORS

Pascal Alfadian Nugroho
He is a lecturer at Universitas Katolik Parahyangan, Bandung, as well as technical director for PT DNArtworks Komunikasi Visual. He holds master's degree in Infocomm Security from National University of Singapore.

Hizkia Steven
She is the project manager at PT DNArtworks Komunikasi Visual. He coordinates his team members on developing websites for various clients. The email is hizkia@dnartworks.co.id