Mapping and Analysis of Common Vulnerabilities in Popular Web Servers*

Matyas Barocsai[1], Johan Can[2], Martin Karresand[1][0009−0009−4203−451X], and Simin Nadjm-Tehrani[2][0000−0002−1485−0802]

[1] Dept. of Cyber Defence and C2 Technology, Swedish Defence Research Agency (FOI), Sweden
matyas.istvan.barocsai@foi.se
martin.karresand@foi.se

[2] Dept. of Computer and Information Science, Linköping university, Sweden
johca907@student.liu.se
simin.nadjm-tehrani@liu.se

Keywords: Cybersecurity, Demonstration, Vulnerability, Web Server, NVD, CVE

Abstract The digitalization of the modern society has made many organizations susceptible to cybercrime through exploitations of software vulnerabilities. The popular web servers Apache HTTP and Nginx make up around 65% of the market for web server software and power the majority of all websites on the internet. Vulnerabilities that occur in these two software programs therefore pose a significant risk to the millions of users.

This paper maps the most common vulnerability types in these web servers by retrieving, filtering, and analyzing information related to around 195,000 reported vulnerabilities. The results not only show that 5 vulnerability types according to the NIST classification, namely CWE-20, CWE-200, CWE-22, CWE-79, and CWE-787, account for almost 25% of all reported vulnerabilities in Apache HTTP and Nginx, but also that these vulnerability types are commonly found in other web software as well. The outcomes of this study are useful for constructing proof-of-concept insecurity demonstrations and for applying in awareness exercises and cybersecurity education.

* The fourth author was supported by the Resilient Information and Control Systems (RICS) project financed by the Swedish Civil Contingencies Agency (MSB).

1 Introduction

The prevalence of attacks on enterprise systems and their historical development has highlighted interfaces that are hard to fix once and for all. Web interfaces have been shown to be notoriously hard to secure, and remain a fertile ground for attacks through exploiting both application-related features and weaknesses in mechanisms introduced to fix security (identity and authentication management, access control, logging and monitoring, etc) [1]. Stopping adversarial access to systems through vulnerabilities requires hardening strategies at both the client [2] and the server ends. This paper focuses on understanding which web server weaknesses persist over time.

Application developer awareness of security controls and attack surfaces is promoted as a long term approach to improve security while the attack and defence landscapes change over time [3]. The backdrop of our work is the provision of learning environments where demonstration of attacks in a realistic environment can enhance the security awareness of developers and users. In this paper we attempt to understand the landscape of web service security in order to identify which types of demonstrations can be used to show the absence of security. We begin by trying to identify the range of web server vulnerabilities seen in massively deployed frameworks. We also expose the historical development of relevant vulnerabilities and provide an overview of the trends.

Vulnerabilities in software can be reported to organizations that run vulnerability databases, which aim to evaluate and catalog vulnerabilities. There are several such databases including Common Vulnerabilities and Exposures (CVE)[3], National Vulnerability Database (NVD)[4], and VulnDB[5]. CVE is a program with the goal of identifying, cataloging and disclosing known vulnerabilities to the public, and is operated by the MITRE Corporation. NVD is a publicly available vulnerability database maintained by National Institute of Standards and Technology (NIST) under the U.S. Department of Commerce. VulnDB is maintained by a commercial enterprise.

Many studies have been conducted to analyze these databases and detect trends and patterns in reported vulnerabilities, but most focus on other software such as operating systems and programming languages, and less on web servers. However, with the increasing number of web servers and internet users, the discovery and exploitation of vulnerabilities in web servers have greater consequences than before.

While education and demonstrations of vulnerabilities can help to increase knowledge of information security, the demonstration of attacks needs performing the learning exercise in an environment that does not create harm. Cyber ranges such as CRATE[6] can be used for training and testing without risking damage to real systems, but developing demonstrations and exercises for vulnerabilities can be a slow process due to the different configurations and versions of software. Thus, it is important to know

[3] `https://www.cve.org/`
[4] `https://nvd.nist.gov/`
[5] `https://vulndb.cyberriskanalytics.com/`
[6] `https://www.foi.se/en/foi/research/information-security/crate---swedens-national-cyber-training-facility.html`

where the focus of a particular exercise should be to provide high impact outcomes. The work in this paper helps to identify web server vulnerabilities that can be considered most relevant when performing such exercises currently.

The paper has the following contributions:

- A systematic investigation of known web server vulnerabilities with the aim of understanding the trends and current state.
- Devising a set of criteria to identify and extract the most relevant vulnerabilities to focus current awareness campaigns on those.

The work includes developing tools that automatically extract web server related vulnerabilities from the NVD database between 1988 and 2022 (amounting to 1 GB of data), and then analyzing the extracted records with respect to interesting properties: the criticality of the vulnerability, the dominance of common weaknesses given their applicability within highly popular frameworks, and the most frequent Common Weakness Enumerations (CWEs). We focus on two web server frameworks, Apache and Nginx, that together comprise the majority of all deployed web software today [4]. All in all, 195,777 CVE records are included in the collected data (up to and including October 2022), from which 174,448 records have a labelled criticality degree. The records are further analyzed and mapped to the categories of interest in terms of dominant weaknesses for each web engine type.

The paper is organized as follows. Section 2 covers the needed background and describes the related works. Section 3 explains the methodology for collecting and processing the vulnerability records and the results from each step of the analysis. Section 4 presents the outcomes of the analysis from Section 3. Section 5 discusses the outcomes presented in Section 4, and main takeaways of the study. Section 6 concludes the paper and presents future works.

2 Background and related works

The vulnerabilities in the CVE database, known as CVE records, are discovered and reported to the CVE Program by organizations, researchers or private individuals. Reported vulnerabilities are evaluated and published by partnered organizations in the CVE Program. Additionally, each vulnerability processed by the CVE Program receives an identifier, the CVE ID, which can be used to uniquely identify the vulnerability. As of April 2023 a little over 199,000 vulnerabilities have been reported to the CVE Program.

Many CVE core vulnerabilities can be similar in terms of their characteristics, causes, or their consequences when exploited. Therefore, there is value in categorizing these vulnerabilities into different types. CWE is a list of identified vulnerability types found in software or hardware. Unlike a CVE record, which refers to a specific vulnerability in a software, a CWE type represents a weakness or characteristic, potentially shared by multiple vulnerabilities, and can be used to categorize CVE vulnerabilities into larger groups. Example CWE types, encountered during our study and thereby relevant for the paper are presented in Table 1.

Table 1. CWE types relevant for the study (sorted by CWE ID)

ID	Name
CWE-16	*Configuration*
CWE-20	*Improper Input Validation*
CWE-22	*Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')*
CWE-78	*Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')*
CWE-79	*Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')*
CWE-89	*Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')*
CWE-94	*Improper Control of Generation of Code ('Code Injection')*
CWE-119	*Improper Restriction of Operations within the Bounds of a Memory Buffer*
CWE-120	*Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')*
CWE-125	*Out-of-bounds Read*
CWE-190	*Integer Overflow or Wraparound*
CWE-200	*Exposure of Sensitive Information to an Unauthorized Actor*
CWE-264	*Permissions, Privileges, and Access Controls*
CWE-287	*Improper Authentication*
CWE-295	*Improper Certificate Validation*
CWE-310	*Cryptographic Issues*
CWE-319	*Cleartext Transmission of Sensitive Information*
CWE-399	*Resource Management Errors*
CWE-400	*Uncontrolled Resource Consumption*
CWE-416	*Use After Free*
CWE-444	*Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling')*
CWE-476	*NULL Pointer Dereference*
CWE-787	*Out-of-bounds Write*
NVD-CWE-Other	Other
NVD-CWE-noinfo	Insufficient Information

Common Vulnerability Scoring System (CVSS) is a system for scoring the severity of a vulnerability. The severity is reflected as a numerical value between 0 and 10 and is the result of an evaluation of the vulnerabilities' characteristics. During evaluation multiple factors are rated, each a member of one of three categories: *Base*, *Environmental* or *Temporal*. The CVSS scoring is often given as both CVSS v2 and v3 values. In CVSS v3 the criticality is extended to also include *None* and *Critical* and the value ranges adjusted accordingly. Table 2 shows the CVSS v2 and v3 scores as defined by their specification documents [5, 6].

Table 2. Comparison of the levels of severity between CVSS v2 and CVSS v3

Severity	CVSS v2	CVSS v3
None		0.0
Low	0.0 - 3.9	0.1 - 3.9
Medium	4.0 - 6.9	4.0 - 6.9
High	7.0 - 10.0	7.0 - 8.9
Critical		9.0 - 10.0

The reported vulnerabilities in the NVD are synchronized with the CVE Program, meaning that newly discovered and published CVE records will be added to the NVD as well. The purpose of the NVD project is to analyze published CVE records and contribute additional detailed information to each vulnerability. The NVD extends the CVE Program by providing evaluations of the CVSS score, the CWE type as well as giving a list of known configurations of software in which the vulnerability has been found.

2.1 Vulnerabilities, attacks and security in web servers

Understanding the intricacies of web servers, such as common attacks, security and development process can greatly aid in understanding why certain vulnerabilities and CWEs occur to a greater extent.

Morton et al. [7] investigated the risks of the new asynchronous web server architecture, particularly in Nginx. Asynchronous web servers offer performance and scalability benefits by enabling memory sharing between clients, something that could potentially pose risks. The authors aimed to investigate if the risk of memory-based vulnerabilities is higher in these new architectures. They presented a method and framework to identify security-critical data by tracking clients' memory handling and examined how this can be used to analyze the exploitation of historical vulnerabilities in Nginx. The study concluded that the asynchronous architecture of Nginx poses a greater risk for memory-based attacks.

Woo, Alhazmi, and Malaiya [8] conducted a study on the potential number of undiscovered vulnerabilities in the popular web server software Apache and Microsoft Internet Information Services (IIS). They used two quantitative discovery models primarily designed for operating systems to estimate the number of undiscovered vulnerabilities, utilizing vulnerability data from NVD. The study revealed that the discovery process for vulnerabilities in web server software follows a pattern that can be modeled, which can be used to predict potential undiscovered vulnerabilities. The authors concluded that the discovery models developed for operating systems can also be applied to web server software.

Alhazmi and Malaiya [9] studied the number of undiscovered vulnerabilities in web server software and the possibility of predicting the number of vulnerabilities in future years. Like Woo, Alhazmi, and Malaiya [8], the authors examined the web server software Apache and Microsoft IIS and used vulnerability data from NVD. They also investigated the use of two models they created to predict undiscovered vulnerabilities. Based on their results, the authors argue that both models have applications in software development processes but that further research is needed to increase their accuracy.

Piantadosi, Scalabrino, and Oliveto [10] investigated the vulnerability fixing process in open-source software. The authors examined who fixes the vulnerabilities, how long it takes to fix them, and the process for fixing them. They analyzed 337 reported CVE vulnerabilities in two open-source programs, Apache HTTP Server and Apache Tomcat, and linked each vulnerability to the specific changes made in the software to fix it.

Piantadosi, Scalabrino, and Oliveto found that the programmers who fixed the vulnerabilities were typically more experienced than the average developer, and that most vulnerabilities were fixed with just one patch. However, approximately 3% of vulnerabilities were not fixed and reappeared in the future. About 80% of vulnerabilities were fixed before the CVE was announced, which is desirable. The remaining vulnerabilities varied greatly in the time taken to fix them, ranging from within 10 days to several years.

While studies such as [7–10] do not provide a list of the most common vulnerabilities in web servers, their research greatly highlights that web server architecture and development process does have an impact on which vulnerabilities occur and potentially reoccur in web server software. Additionally, they demonstrate how vulnerability databases such as NVD and CVE can be utilized in the research of web servers. Notably, Woo, Alhazmi, and Malaiya [8] and Alhazmi and Malaiya [9] illustrate how information provided by the NVD can be used to map vulnerabilities to certain web server technologies. Our work, therefore builds on those insights and extends them to the full range of web server vulnerabilities.

2.2 Analysis of vulnerability databases

Analyzing records in vulnerability databases such as NVD and CVE can reveal trends and previously unknown relations between vulnerabilities and web servers. Studies which have explored NVD data provide a solid foundation for finding trends, and their methodology can be applied to identify the most common vulnerabilities in web servers.

Gorbenko et al. [11] investigated which operating systems have the highest number of reported vulnerabilities and the life-cycle of these vulnerabilities. The authors combined and mapped known vulnerabilities from the CVE and NVD databases between 2012 and 2016. The results show that both the number of reported vulnerabilities and security risk have increased between 2012 and 2016. Furthermore, the authors found that the average time it takes for developers to fix a vulnerability is the same regardless of how critical the vulnerability is.

Kuhn, Raunak, and Kacker [12] analyzed trends in older and current vulnerabilities by examining reported vulnerabilities in the NVD vulnerability database between 2006 and 2016. Their earliest analysis between 2006 and 2010 showed that the most common vulnerability types were XSS and SQL injection attacks (CWE-79 and CWE-89). About 15% of the reported vulnerabilities during this time were unclassified and did not belong to any vulnerability type. A later analysis between 2010 and 2017 showed that the most common vulnerability types were CWE-16: Configuration and CWE-20: Improper Input Validation. The authors attribute the decrease in SQL injection and XSS vulnerabilities partly to new tools and methods that facilitate their discovery. Furthermore, the later analysis showed that the number of unclassified vulnerabilities has decreased, which the authors believe shows how the work in cybersecurity has developed and become normalized over the years.

Fan et al. [13] created a vulnerability database for large open-source software written in C/C++. They combined information from the CVE vulnerability database with related vulnerability information, and connected vulnerabilities to the specific code that was vulnerable and the code that fixed the vulnerability. The authors aimed to facilitate vulnerability analysis by providing information on the specific vulnerable code. The result was a database with 3754 vulnerabilities from 348 different GitHub projects. The authors analyzed trends in their vulnerability database and found that CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer and CWE-20: Improper Input Validation were the most common vulnerability types among the examined software.

Anwar et al. [14] investigated the quality of the evaluations performed by NVD for each reported CVE. The study examines whether NVD's evaluations are consistent and, among other things, whether correct CVSS scores and CWE types are provided by NVD. Furthermore, the authors investigate whether the publication date listed in NVD adds value and whether it correctly reflects the vulnerability's public awareness. By examining URLs listed as references in NVD for each vulnerability, the authors found that in several cases, the vulnerability's existence had been publicly disclosed several months before the publication date in NVD. The authors argue that the publication date shown in NVD only indicates when a vulnerability was published in the database and not when it was first mentioned in the public domain. Therefore, the publication date may not necessarily be valuable, and the first detection date should possibly be provided instead.

Regarding CWE types, Anwar et al. found that several vulnerabilities categorized as NVD-CWE-Other actually contained a specified CWE type in the vulnerability description. Therefore, the authors could correctly categorize an additional 1732 vulnerabilities identified as NVD-CWE-Other into their correct CWE types. Finally, the authors conclude that they have been able to assess and address several of the inconsistent attributes in NVD that they identified. They believe that it is valuable to address these inconsistencies to increase the quality of NVD, not only because the vulnerability database is widely used in cybersecurity but also because a consistent vulnerability database would provide a better overview of vulnerability trends.

Williams et al. [15] investigated hidden trends in NVD by analyzing its content with machine learning. They aimed to examine how the most common vulnerabilities relate to the products with the most reported vulnerabilities by analyzing recurring words and phrases in vulnerability descriptions. The study identified the top 50 software products and top 50 vulnerability types between 2000 and 2017. The results show unexplored trends in NVD and clear connections between common vulnerability types and products they occur in. The authors suggest that the results, reinforced by various machine learning models, are valuable for developing secure software. Further research will be conducted on the presented analysis model, Supervised Topical Evolution Model (STEM), to predict vulnerabilities in software in the future.

None of the studies by Gorbenko et al., Kuhn, Raunak, and Kacker, Fan et al., Anwar et al., Williams et al. provide an overview and analysis of the vulnerability trends in web servers and which CWEs that contribute to the most vulnerabilities in these technologies. Williams et al. for example uses machine learning to detect phrases related to CWEs. Their results estimate which of these phrases are most commonly associated with vulnerabilities, but they do not identify the specific CWE types that are most prevalent. Furthermore, Williams et al. only examine vulnerabilities reported to NVD between 2000 and 2017, which means that a great amount of vulnerabilities since 2017 remain unmapped.

Despite being unrelated to web technologies, the methodology used in [11–15] can be viewed as an evidence that conducting a quantitative investigation of vulnerability trends is feasible, relevant, and valuable. Furthermore, it highlights how valuable the data in the NVD is. To the best of our knowledge, no other works have devised tools to systematically extract and analyze the web server-specific vulnerabilities from the NVD/CVE databases and CWE categorizations, in order to provide insights on trends and weaknesses in this domain.

3 Systematic analysis of web server vulnerabilities

The analysis and investigation of potential trends in vulnerabilities were pursued by adopting a methodology that consisted of four stages. The four stages were as follows:

Data collection: Collect data on reported vulnerabilities

Data filtering: Convert/filter the data into a manageable format

Analysis: Analyze the data for trends and patterns and identify relationships between common vulnerabilities and popular web servers

Presentation: Present the results of the analysis using a suitable visualization

Each of the four stages presented challenges that had to be overcome. The subsequent subsections provide an account of these challenges and their corresponding solutions.

3.1 Vulnerability data collection

In this study, data was retrieved from NVD, as this database is; openly accessible to individuals, large and contains many vulnerabilities, reliable, and has frequently been used in previous research [8, 9, 11, 12, 14, 15]. Furthermore, NVD can be seen as an extension of the CVE database. The two databases are fully synchronized, but NVD also provides some additional information about vulnerabilities, such as CVSS score, references, CWE type, and possible vulnerable system configurations. The data in the NVD can be accessed through their website or by downloading the vulnerability data to a local machine. The NVD offers various formats for downloading the data, such as XML and JSON. For this study, JSON was selected as it was considered to be the most manageable. A total of 21 separate JSON files were downloaded, containing unfiltered information on all reported vulnerabilities between the years 1988 and 2022, summarised in Table 3.

Table 3. Overview of downloaded JSON files

JSON-file	Year	Lines	Size [MB]
1	1988 - 2002	622,643	21.1
2	2003	170,922	6.0
3	2004	352,021	12.6
4	2005	565,333	19.9
5	2006	862,034	30.5
6	2007	807,329	28.5
7	2008	940,392	33.3
8	2009	890,111	32.4
9	2010	946,944	34.7
10	2011	942,934	34.8
11	2012	1,096,774	40.1
12	2013	1,240,427	45.5
13	2014	1,124,411	40.8
14	2015	1,114,185	40.2
15	2016	1,433,754	51.6
16	2017	2,070,494	73.9
17	2018	2,174,419	77.1
18	2019	2,476,073	87.2
19	2020	3,247,700	112.8
20	2021	3,585,469	124.4
21	2022	1,665,930	58.7
Total	34 years	28,330,299	1 006.1

3.2 Vulnerability data filtering

In Table 3, it is clear that manual filtering of vulnerabilities would not be feasible due to the large amount of data that would need to be manually reviewed. For this reason, there was a need to automate the filtering of the data obtained from the JSON files. To this end, a tool was developed with the purpose of automatically filtering all vulnerabilities in the downloaded JSON files and storing them in a local database. The tool was developed in the Python programming language[7], as this language has good compatibility with JSON files. Furthermore, the SQLite database[8] was chosen because it has good compatibility with Python.

The developed tool was useful and necessary because it could filter out useful fields of each vulnerability instead of having to deal with irrelevant data related to the vulnerabilities. Much of the information provided about CVE vulnerabilities from the JSON files was unnecessary and therefore needed to be filtered out. The majority of all data such as reference links, information about configurations, details about how CVSS values were calculated, the latest modification date, and who assigned the vulnerability CVE ID were not relevant to this work.

The purpose of the filtering was to find vulnerabilities based on the type of vulnerability, when it was reported, how serious it was, and its ID. Therefore, data fields that did not have information about this could be filtered out. The following fields were left after filtering the JSON files through the developed Python tool:

CVE ID: This field is used to identify the vulnerability. The CVE ID assigned to a vulnerability is unique and is therefore well suited as a key for a database entry.

CVSS v3: The newer CVSS v3 is the primary way of rating the severity level of vulnerabilities from 2016 onwards. Using this field, the severity level among the reported vulnerabilities can be analyzed.

CVSS v2: The older CVSS v2 was used to rate the severity level of vulnerabilities before 2016, but it also appears frequently in newer vulnerabilities.

Publication date: This field shows on which date the vulnerability was published in the NVD. This field can be used to examine vulnerability trends over time.

CWE: This field shows the CWEs that are associated with the vulnerability and can be used to investigate how different vulnerability types have developed over time.

Description: This field provides a brief summary and explanation of the vulnerability and in which software it can be found. This field can be used to find vulnerabilities related to specific technologies, in our case web servers.

By using these limited number of database fields, the amount of information stored about each vulnerability was greatly decreased. Consequently, it became significantly easier to find desired data and investigate potential trends in the reported vulnerabilities.

3.3 Criteria for vulnerability selection

With the local database, which contained the filtered contents of the NVD, extraction and exploration of hidden vulnerability trends was made significantly easier. The use

[7] `https://www.python.org/`

[8] `https://www.sqlite.org/index.html`

of an SQLite database made it possible to write SQL queries in order to extract the data which was pursued.

Some vulnerability types would clearly be more prevalent than others. Nevertheless, most software systems still have a wide variety of vulnerability types reported, some which have only been reported once throughout the software's lifespan. To focus only on the most common vulnerability types, those types that were only reported once or twice were disregarded and grouped together. This way only the historically 15 most reported vulnerabilities were analyzed and investigated.

A few points of interest were pursued when extracting data for the analysis of trends in the reported vulnerabilities for the web servers Apache HTTP and Nginx. When examining the data extracted, the following points were studied:

- Has the number of reported vulnerabilities increased or decreased since the introduction of the web server?
- Which vulnerability types have been the most common?
- Is there a uniquely prominent vulnerability type, which has occurred in a far greater extent?
- Are there any specific years which are distinctive for the web servers?
- How does the most common vulnerability types compare between the two web servers?

Fig. 1, shows how trends in the NVD can be explored using the SQL queries, in this case simply the number of reported vulnerabilities each year.

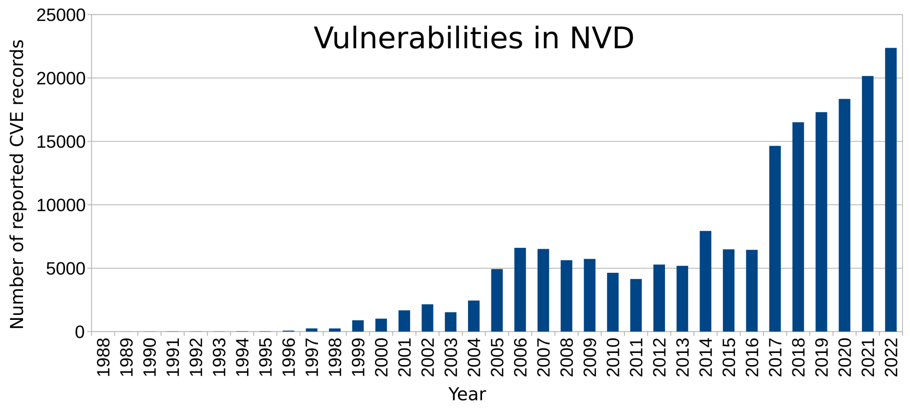

Figure 1. Reported vulnerabilities in the NVD between the years 1988 and 2022

Even in this simple example, Fig. 1, interesting trends and characteristics can be seen. It is evident that the number of reported vulnerabilities increases each year when examining the long-term trend. There are also two significant increases in 2005 and

2017. Between these two major increases, during the years 2000-2005 and 2006-2016, the number of reported vulnerabilities remained relatively stable. Since 2017, there has been a consistent annual increase in the number of reported vulnerabilities. Apart from the two major spikes of increase, the year 2014 stands out, with a visibly higher number of reported vulnerabilities compared to the surrounding years.

4 Analysis outcomes

By collecting, filtering, and examining the reported CVE records in the NVD, the investigation for the most common vulnerability types in Apache HTTP and Nginx could be completed. This section presents the results of this analysis.

4.1 General overview of the vulnerabilities in the NVD

When not considering the program or system in which a reported vulnerability occurs, the most common vulnerability type reported in the NVD can be found. Studying which vulnerability types are the most common in the entire NVD can be beneficial as it gives an insight into which types are common regardless of system. In Fig. 2, the three most commonly reported vulnerabilities each year can be seen, without regard to the system it occurred in.

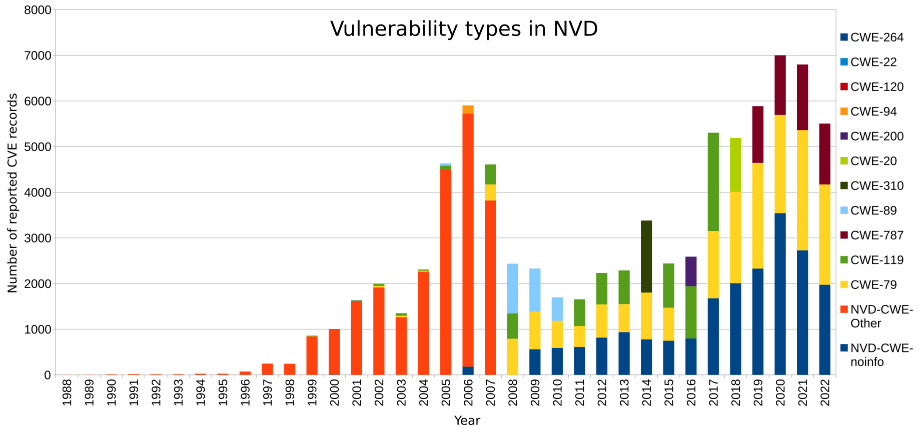

Figure 2. The 3 most common vulnerabilities each year in NVD

In Fig. 2, it can be observed that until 2008, the vulnerability type NVD-CWE-Other dominated, followed by the dominance of the vulnerability type NVD-CWE-noinfo. Both types are used in the publication of unclassified vulnerabilities by the NVD.

Furthermore, it is evident that the vulnerability types CWE-79 and CWE-119 have been commonly occurring for an extended period between 2007 and 2017, while the type CWE-787 has had a clear presence since 2019.

Year 2014 stands out, as the vulnerability type CWE-310 suddenly becomes one of the most prevalent types, only to diminish in the year after. Similarly, the types CWE-200 and CWE-20 emerge among the most reported vulnerability types in 2016 and 2018, respectively, but diminish in later years. Table 4 presents the 5 most reported vulnerability types out of a total 195,477 reported vulnerabilities.

Table 4. The 5 most common vulnerability types in NVD

Vulnerability type	N	% of total
NVD-CWE-Other	23,419	11.9
NVD-CWE-noinfo	20,273	10.3
CWE-79	18,979	9.7
CWE-119	7,479	3.8
CWE-787	5,319	2.7
Sum	75,469	38.4

In Table 4, it is clear that the vulnerability types NVD-CWE-Other and NVD-CWE-noinfo constitute the largest proportion of all reported vulnerabilities in the NVD, accounting for over 20% of all reported vulnerabilities. Furthermore, when considering only properly classified vulnerabilities, it can be observed that the type CWE-79: Improper Neutralization of Input During Web Page Generation, stands out as the most prevalent vulnerability type.

4.2 Historical view of web server vulnerabilities

The result of the investigation analyzing reported vulnerabilities related to the web servers Apache HTTP and Nginx shows that between 1997 and 2022, 300 different vulnerabilities related to Apache HTTP were reported. Furthermore, 141 vulnerabilities related to the web server software Nginx were reported between 2009 and 2022. In Fig. 3, the development of the 15 most common CWE types and their frequency for vulnerabilities related to Apache HTTP can be seen, and in Fig. 4, the corresponding information for Nginx are displayed.

In Fig. 3, it is clear that vulnerability types representing non-classifiable vulnerabilities, CWE-Other and NVD-CWE-noinfo, constitute a significantly large portion of the 15 most common vulnerability types. Furthermore, it can be seen that NVD-CWE-Other was more common between 1997 and 2007, while NVD-CWE-noinfo was more common after 2007.

Two vulnerability types representing classifiable vulnerabilities, CWE-20 and CWE-200, are commonly found between 2011 and 2017. CWE-79 is common between 2007

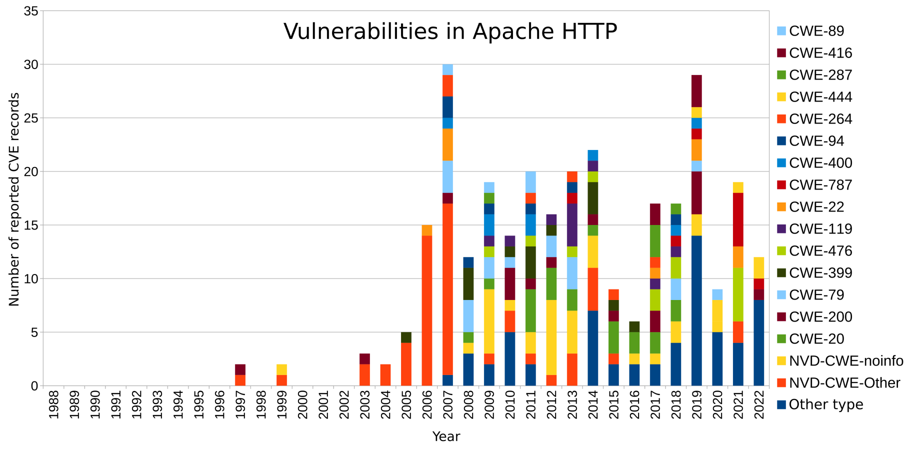

Figure 3. The 15 most common vulnerabilities in Apache HTTP

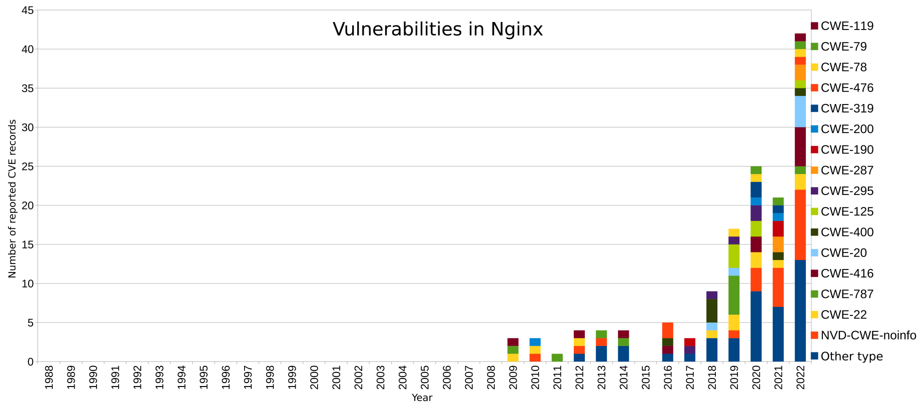

Figure 4. The 15 most common vulnerabilities in Nginx

and 2013 but has since then decreased in frequency. Of the total of 300 vulnerabilities related to Apache HTTP, 152 vulnerabilities are classified in one of the 15 most common CWE types. 87 vulnerabilities have not been classified and therefore belong to either CWE-Other or NVD-CWE-noinfo, and 61 vulnerabilities were classified to CWE types that are not among the 15 most common.

The number of reported vulnerabilities each year since 2006 is approximately the same, around 15-20. Distinctive years are 2007 and 2019 when more vulnerabilities were reported than the average. Furthermore, 2015, 2016, and 2020 are distinctive years when significantly fewer vulnerabilities were reported than the average during these years.

For reported vulnerabilities in Nginx, as shown in Fig. 4, it can be seen that the vulnerability type representing non-classifiable vulnerabilities, NVD-CWE-noinfo, is prevalent, especially in the last three years. Vulnerability types CWE-22 and CWE-787 have a clear occurrence for many years. Of the total of 141 vulnerabilities related to Nginx, 78 of these are classified in one of the 15 most common CWE types, while 42 vulnerabilities have been classified to CWE types that are not among the 15 most common. 21 vulnerabilities have not been classified and belong to NVD-CWE-noinfo. Between 2009 and 2017, approximately the same number of vulnerabilities were reported each year. From 2018 onwards, there is a clear increase in reported vulnerabilities each year. A distinctive year is 2022, where almost twice as many vulnerabilities were reported as the previous year.

4.3 The most common vulnerabilities

The analysis of reported vulnerabilities in Apache HTTP and Nginx web servers reveals that the most common vulnerability types represent over 50% of all reported vulnerabilities, with slight variations in occurrence percentages. The analysis investigated 441 vulnerabilities in web servers and found that some vulnerability types are common in both web server software.

As shown in Table 5, the 15 most common vulnerability types represent just over 51% of all reported vulnerabilities related to the Apache HTTP web server, and the vulnerability type CWE-20 is the most common.

Analogously, in Table 6, it is clear that the 15 most common vulnerability types represent around 55% of all reported vulnerabilities related to the Nginx web server. Furthermore, the vulnerability type CWE-22 is the most common. The table shows that the number of the three most common vulnerability types differs very little. CWE-22 has only occurred in 1-2 more instances than CWE-787 and CWE-416. The gap between CWE-416 and CWE-20 is clearer, with a difference in percentage of more than 2 percentage points. For the other vulnerability types in the table, there are no significant differences in either number or percentage, with most vulnerability types being equally common.

Table 5. The 15 most common vulnerabilities in Apache HTTP, excluding NVD-CWE-Other and NVD-CWE-noinfo

Apache HTTP		
CWE-ID	N	% of total (300)
CWE-20	21	7.0
CWE-200	17	5.7
CWE-79	17	5.7
CWE-399	14	4.7
CWE-476	13	4.3
CWE-119	10	3.3
CWE-22	9	3.0
CWE-787	9	3.0
CWE-400	8	2.7
CWE-94	7	2.3
CWE-444	6	2.0
CWE-264	6	2.0
CWE-287	5	1.7
CWE-89	5	1.7
CWE-416	5	1.7
Sum	152	51

Table 6. The 15 most common vulnerabilities in Nginx, excluding NVD-CWE-Other and NVD-CWE-noinfo

Nginx		
CWE-ID	N	% of total (141)
CWE-22	11	7.8
CWE-787	10	7.1
CWE-416	9	6.4
CWE-20	6	4.3
CWE-125	6	4.3
CWE-400	6	4.3
CWE-295	5	3.5
CWE-287	4	2.8
CWE-200	3	2.1
CWE-190	3	2.1
CWE-119	3	2.1
CWE-319	3	2.1
CWE-476	3	2.1
CWE-78	3	2.1
CWE-79	3	2.1
Sum	78	55

The result of the analysis, which investigated 441 vulnerabilities in web servers, 300 in Apache HTTP and 141 in Nginx, shows that some vulnerability types occur more frequently than others and that some vulnerability types are prevalent in both web server softwares. Table 7 presents the 5 most common vulnerability types in Apache HTTP and Nginx.

Table 7. The 5 most common vulnerability types in both web servers Apache HTTP and Nginx

Top 5				
CWE-ID	N	% of total (441)	Rank (Apache HTTP)	Rank (Nginx)
CWE-20	27	6.1	1	4
CWE-200	20	4.5	2	9
CWE-22	20	4.5	7	1
CWE-79	20	4.5	3	15
CWE-787	19	4.3	8	2
Sum	106	23.9		

In Table 7, vulnerability CWE-20 is the most common vulnerability type among the web servers Apache HTTP and Nginx. This vulnerability type has occurred 7 times more than other vulnerability types. CWE-200, CWE-22, and CWE-79 are equally common, occurring on a par with CWE-787, and each account for 4.5% of the 441 reported vulnerabilities. Together, the top 5 most common vulnerability types make up 24% of all reported vulnerabilities among the web servers.

5 Why do we need to act?

The above analysis brings us insights about the current state of common web server vulnerabilities. In this section we discuss why having this data can help us to focus on problems that are most relevant to fix. But before that, we need to get the attention of software developers and system owners about these issues.

5.1 Impact of severe vulnerabilities

The exploitation of a vulnerability can have a large impact on systems used globally. Noteworthy examples such as HeartBleed (2014) [16], EternalBlue (2017) [17], and ZeroLogon (2020) [18] have all caused severe impacts on numerous systems worldwide. A more recent example is Log4Shell (2021) [19, 20], a vulnerability in the Apache Log4j logging utility, which led to significant disruptions worldwide due to the broad usage of the logging library.

While severe and impactful vulnerabilities most likely will not disappear entirely, developers can prepare by having a good understanding of what vulnerabilities they can expect. Possessing a deep understanding of all the vulnerability types found in the CWE list is an unrealistic demand. Therefore, it becomes highly valuable to identify a narrower subset of vulnerability types that occur in the systems in which they work, such as the most common or the most severe ones. Even for the narrower class of vulnerabilities the recent history tells us that systems are not patched immediately once a CWE has been associated with a component in them. Therefore, we need to keep pressing for attention towards the prevalent ones.

5.2 Demonstrations to raise awareness

Cybersecurity demonstrations serve not only as a means to raise awareness about cybersecurity issues but can also be valuable as an educational tool. Puys, Thevenon, and Mocanu [21] and Kalyanam and Yang [22] show that incorporating cybersecurity demonstrations into training platforms for cybersecurity education can have a significant effect on the participants' understanding of challenges within cybersecurity and secure development practices.

In order for cybersecurity demonstrations to be effective, it is necessary that they demonstrate interesting and relevant cybersecurity issues. When focusing on web servers, demonstrating the exploitation of a well-known vulnerability can provide valuable insight into the impacts and consequences of insecure systems. However, selecting an appropriate well-known vulnerability to base a demonstration on is a non-trivial task.

Simplifying this task could potentially be achieved through mapping the most common vulnerabilities in the most popular web servers. The identification of the most common vulnerability types can prove to be valuable as they signify recurring security concerns and highlight design flaws within the systems. For developers working with these web servers, understanding the impact and consequences of the most common vulnerability types can be advantageous, as well as understanding the reason for the vulnerabilities occurrence. Subsequent to our analysis, demonstrations of the most relevant ones were realised in a virtualized environment. However, the details of these implementations are beyond the scope of this paper and the interested reader is referred to the work by Barocsai and Can [23] for details.

6 Conclusions

Vulnerability databases offer immense value with their contents, but they can also potentially provide far more information, beyond their intended purpose. Through thorough analysis, one can extract valuable information about vulnerability trends and the relationship between vulnerability and software.

In this study, over 195,000 vulnerability records were collected, filtered and analyzed to investigate the prevalence of different vulnerability types in the two widely used web server softwares, Apache HTTP and Nginx. Results show that the top 15 most frequently reported CWE types account for almost half of all reported vulnerabilities. Furthermore, it can be seen that almost 25% of the reported vulnerabilities in these web servers are categorised to be of just 5 CWE types.

Cybersecurity demonstrations play a crucial role in education and raising awareness. However, they require that their contents and demonstrated vulnerabilities are both interesting and relevant. Finding suitable vulnerabilities to demonstrate is therefore a challenging and time-consuming task. This study aims to contribute to this area, by narrowing the down the selection of vulnerabilities, making it easier to find a suitable match.

References

[1] A. van der Stock, B. Glas, N. Smithline, and T. Gigler. *OWASP Top 10:2021*. Last accessed 2023-06-07. Sept. 2021. URL: `https://owasp.org/Top10/`.

[2] S. Roth, S. Calzavara, M. Wilhelm, A. Rabitti, and B. Stock. "The Security Lottery: Measuring Client-Side Web Security Inconsistencies." In: *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 2047–2064. ISBN: 978-1-939133-31-1. URL: `https://www.usenix.org/conference/usenixsecurity22/presentation/roth`.

[3] M. Sahin, T. Ünlü, C. Hébert, L. A. Shepherd, N. Coull, and C. M. Lean. "Measuring Developers' Web Security Awareness from Attack and Defense Perspectives." In: *2022 IEEE Security and Privacy Workshops (SPW)*. 2022, pp. 31–43. DOI: `10.1109/SPW54247.2022.9833858`.

[4] *Netcraft: May 2023 Web Server Survey*. Last accessed 2023-10-27. URL: `https://www.netcraft.com/blog/may-2023-web-server-survey/`.

[5] *Common Vulnerability Scoring System Version 2*. Last accessed 2023-10-27. URL: `https://www.first.org/cvss/v2/guide`.

[6] *Common Vulnerability Scoring System Version 3.1*. Last accessed 2023-10-27. URL: `https://www.first.org/cvss/specification-document`.

[7] M. Morton, J. Werner, P. Kintis, K. Snow, M. Antonakakis, M. Polychronakis, and F. Monrose. "Security Risks in Asynchronous Web Servers: When Performance Optimizations Amplify the Impact of Data-Oriented Attacks." In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2018, pp. 167–182. DOI: `10.1109/EuroSP.2018.00020`.

[8] S.-w. Woo, O. H. Alhazmi, and Y. K. Malaiya. "Assessing Vulnerabilities in Apache and IIS HTTP Servers." In: *2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*. 2006. DOI: `10.1109/DASC.2006.21`.

[9] O. H. Alhazmi and Y. K. Malaiya. "Measuring and Enhancing Prediction Capabilities of Vulnerability Discovery Models for Apache and IIS HTTP Servers." In: *2006 17th International Symposium on Software Reliability Engineering*. 2006, pp. 343–352. DOI: `10.1109/ISSRE.2006.26`.

[10] V. Piantadosi, S. Scalabrino, and R. Oliveto. "Fixing of Security Vulnerabilities in Open Source Projects: A Case Study of Apache HTTP Server and Apache Tomcat." In: *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*. 2019, pp. 68–78. DOI: `10.1109/ICST.2019.00017`.

[11] A. Gorbenko, A. Romanovsky, O. Tarasyuk, and O. Biloborodov. "Experience Report: Study of Vulnerabilities of Enterprise Operating Systems." In: *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*. 2017, pp. 205–215. DOI: `10.1109/ISSRE.2017.20`.

[12] R. Kuhn, M. Raunak, and R. Kacker. "It Doesn't Have to Be Like This: Cybersecurity Vulnerability Trends." In: *IT Professional* 19.6 (2017), pp. 66–70. DOI: `10.1109/MITP.2017.4241462`.

[13] J. Fan, Y. Li, S. Wang, and T. N. Nguyen. "AC/C++ code vulnerability dataset with code changes and CVE summaries." In: *Proceedings of the 17th International Conference on Mining Software Repositories*. 2020, pp. 508–512. DOI: `10.1145/3379597.3387501`.

[14] A. Anwar, A. Abusnaina, S. Chen, F. Li, and D. Mohaisen. "Cleaning the NVD: Comprehensive Quality Assessment, Improvements, and Analyses." In: *IEEE Transactions on Dependable and Secure Computing* 19.6 (2022), pp. 4255–4269. DOI: 10.1109/TDSC.2021.3125270.

[15] M. A. Williams, S. Dey, R. C. Barranco, S. M. Naim, M. S. Hossain, and M. Akbar. "Analyzing Evolving Trends of Vulnerabilities in National Vulnerability Database." In: *2018 IEEE International Conference on Big Data (Big Data)*. 2018, pp. 3011–3020. DOI: 10.1109/BigData.2018.8622299.

[16] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman. "The Matter of Heartbleed." In: IMC '14. Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 475–488. ISBN: 9781450332132. DOI: 10.1145/2663716.2663755.

[17] D. Goodin. "NSA-leaking Shadow Brokers just dumped its most damaging release yet." In: *ArsTechnica* (Apr. 2017). Last accessed 2023-06-09. URL: https://arstechnica.com/information-technology/2017/04/nsa-leaking-shadow-brokers-just-dumped-its-most-damaging-release-yet/.

[18] NIST. *CVE-2020-1472 Detail*. Last accessed 2023-06-09. Aug. 2020. URL: https://nvd.nist.gov/vuln/detail/CVE-2020-1472.

[19] R. Hiesgen, M. Nawrocki, T. C. Schmidt, and M. Wählisch. *The Race to the Vulnerable: Measuring the Log4j Shell Incident*. 2022. arXiv: 2205.02544 [cs.CR].

[20] S. Feng and M. Lubis. "Defense-In-Depth Security Strategy in Log4j Vulnerability Analysis." In: *2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*. 2022, pp. 01–04. DOI: 10.1109/ICADEIS56544.2022.10037384.

[21] M. Puys, P.-H. Thevenon, and S. Mocanu. "Hardware-In-The-Loop Labs for SCADA Cybersecurity Awareness and Training." In: *Proceedings of the 16th International Conference on Availability, Reliability and Security*. ARES 21. Vienna, Austria: Association for Computing Machinery, 2021. ISBN: 9781450390514. DOI: 10.1145/3465481.3469185.

[22] R. Kalyanam and B. Yang. "Try-CybSI: An Extensible Cybersecurity Learning and Demonstration Platform." In: *Proceedings of the 18th Annual Conference on Information Technology Education*. SIGITE '17. Rochester, New York, USA: Association for Computing Machinery, 2017, pp. 41–46. ISBN: 9781450351003. DOI: 10.1145/3125659.3125683.

[23] M. Barocsai and J. Can. "Kartläggning, demonstration och hantering av vanliga sårbarheter i populära webbservrar." MSc thesis. Linköping University, Sweden, Mar. 2023. URL: https://www.diva-portal.org/smash/get/diva2:1743044/FULLTEXT01.pdf.