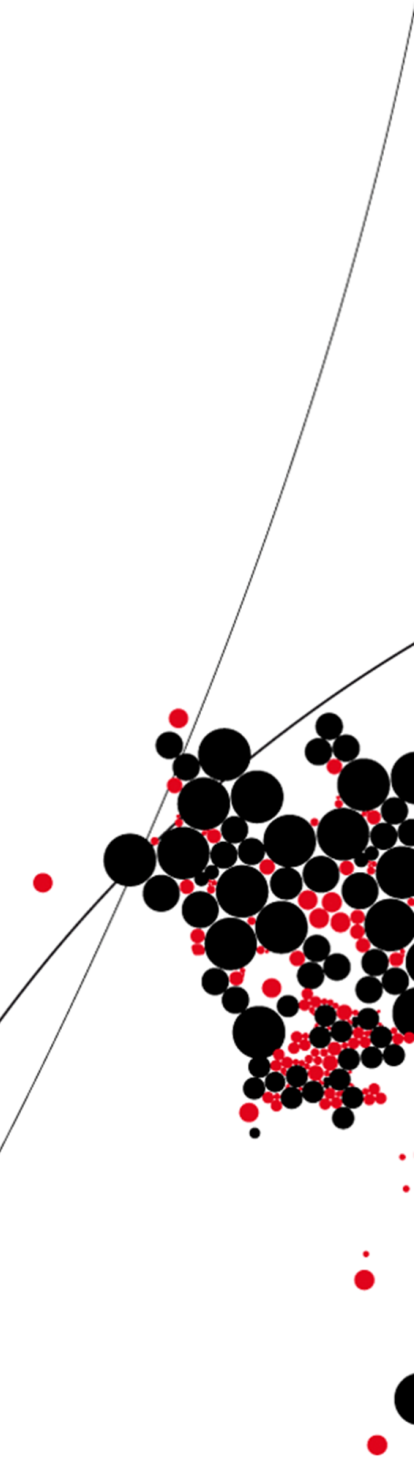
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science



An Internet-wide investigation of publicly accessible databases

Samuel J. Witt

M.Sc. Thesis
23-11-2023

Supervisors:
Dr. R. Holz
Dr. A. Continella

Design and Analysis of
Communication Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

Database technology is a cornerstone of the modern digital society. In this report, we investigate the worldwide landscape of publicly accessible databases and their security postures. We design, implement and carry out 2 Internet scans for open default ports of 8 of the most popular database solutions in existence. Internet scanning has been done numerous times for a multitude of purposes, an overview of required background knowledge is presented, together with a discussion of relevant previous work. Ethical guidelines with respect to Internet scanning as established by the computer science community are followed to minimise the intrusiveness of our scans. Our focus is on the connection and deployment security of publicly accessible databases. To measure the connection security posture, we look at Transport Layer Security (TLS) properties, including versions, cipher suites and certificate validity. Furthermore, we assume the perspective of an outside user in our security assessment of deployed database systems to determine their level of vulnerability. Our scan detected a total of 3.5 million databases among the 8 database solutions, of which 51% offered TLS in 2023 as a way to secure the connection between client and server. 89% of those servers do not use TLS versions prior to TLS 1.2, 78% force RECOMMENDED cipher suites for TLS encryption, 9% of TLS server identity certificates could be verified using Mozzila's CCADB, and 74% of servers offer self-signed certificates. We discovered that 5 database solutions have their version string available to any connecting client, and it is possible to deduce if the version is vulnerable to remote attacks. We show that a significant portion of publicly accessible database servers run versions vulnerable to one or more attacks exploitable from remote using an Internet connection. The key takeaway from our research is that security configuration and software maintenance remain challenges for database server administrators.

Contents

Chapter 1

Introduction

1.1 Motivation

In today's digital society, there is an enormous amount of data created every day by both organisations and individuals. Databases have become the foundation of modern information technology, serving as valuable repositories for research, information sharing, and knowledge exchange across various domains. These databases offer unprecedented opportunities for researchers, businesses, and the general public to access, analyse, and utilise the large amounts of data the world generates in ways that were previously unimaginable.

However, with the immense benefits of these databases come security concerns. As the repository of sensitive and personal information expands, so does the potential for malicious activities and cyber threats. In recent years there have been more and more data breaches, cyber attacks, and unauthorised access incidents, highlighting the need to evaluate the security landscape of such databases rigorously.

This research report seeks to address these concerns by conducting an in-depth investigation into publicly accessible databases worldwide, specifically from a security perspective. We focus our attention on the confidentiality of the communication and the identity of database servers, and their susceptibility to remote cyber attacks. The traffic between client and server might contain sensitive information like usernames and passwords, and therefore it is important to ensure it is sufficiently encrypted to rule out possible third parties reading the communication. Furthermore, the client needs to be able to verify that the server it is connected to is the genuine server they intended to connect to and not a pretender. Scanning is a method to detect servers and establish a connection emulating as a client. In this way the server can be observed and information gathered about the way it presents itself and secures its communication.

We carry out two worldwide scans, one in October 2021 and a second scan in March 2023. The first scan serves as a benchmark for the state of publicly accessi-

ble databases to compare the second scan to in order to ascertain how the security has progressed or declined over 17 months time. Originally the research focused on doing one worldwide scan, however due to personal circumstances of one of the researchers we found ourselves in a position to orchestrate a second scan with minor improvements. This gives us the unique chance to look at how publicly accessible databases and their security measures evolved over an extended time period.

This research report aims to raise awareness about the critical need for robust data protection measures by unveiling the current state of security in publicly accessible databases. The insights gained might serve as a valuable resource for policymakers, database administrators, cyber security professionals, and researchers, aiding with creating effective security strategies and risk management protocols. Ultimately, this study aims to contribute to a safer and more secure digital landscape, improving trust and confidence in the usage of databases.

1.2 Research goals and contributions

The goal of the research is to investigate the connection and deployment security of publicly accessible databases. To measure the connection security posture, we look at Transport Layer Security (TLS) properties, including versions, cipher suites and certificate validity. We assume the perspective of an outside user in our security assessment of deployed database systems to determine their level of vulnerability.

We have 2 main research questions, as follows:

- How secure are connections with publicly accessible databases?
- How secure is database management software (DBMS) running on publicly accessible databases?

With our research, we paint a picture of a worldwide publicly accessible databases landscape consisting of 8 of the worlds most popular database solutions. To achieve this we leverage the Zgrab2 [1] scanning tool for our scans. We analyse the results of the scans and discuss our findings. Our main contributions are:

- A Zgrab2 scanning module for Cassandra databases.
- Data from 2 worldwide Internet scans for 8 popular database solutions.
- An analysis of the gathered data focusing on the connection and software security configurations.
- Insights for database administrators, policymakers, cyber security professionals, and researchers.

1.3 Report organisation

The remainder of this report is organised as follows. In Chapter 2, required background knowledge is given, as well as a discussion of related work and an overview of ethical challenges with internet scanning. Then, in Chapter 3, we outline how the scans are conducted and analysed . Results are presented in Chapter 4, we discuss them in Chapter 5, and conclude in Chapter 6.

Chapter 2

Literature research

2.1 Background

The purpose of this section is to introduce security concepts that are critical to understand the results of this study: Transport Layer Security (TLS), the X.509 standard for certificates, and the Common Vulnerabilities and Exposures (CVE) system.

2.1.1 Transport Layer Security (TLS)

TLS (Transport Layer Security) is a cryptographic protocol used to secure data communication over computer networks, specifically on the internet. It ensures the confidentiality, integrity, and authenticity of data transmitted between a client and a server. TLS achieves this through encryption, a secure digital handshake process, and the use of digital certificates for verifying the identity of servers. A TLS handshake consists of the client and server deciding on which TLS version and cipher suite to use, authenticating the server's identity, and generating session keys for symmetric encryption during the connection. By safeguarding sensitive information and establishing secure connections, TLS plays a vital role in maintaining the privacy and security of online interactions. It has existed for quite some time and undergone several iterations to keep up with an ever evolving cyber security environment, with the most prevalent versions in use today being TLS 1.2 and TLS 1.3. Older TLS versions 1.1 and 1.0 are no longer considered secure at the time of writing [2]. It's predecessor, SSL (Secure Sockets Layer), has critical security concerns and has been deprecated since 2015 [3].

2.1.2 X.509 standard

X.509 is a widely adopted standard for digital certificates in public key infrastructure. These certificates play a crucial role in verifying the authenticity of entities,

such as servers, clients, or individuals, during secure communication. X.509 certificates contain, among others, information about the entity's identity, the Certificate Authority (CA), and the entity's public key. They are digitally signed by the issuer to ensure their integrity, and a chain of trust is established by verifying the issuer's certificate up to a trusted root certificate. X.509 certificates are fundamental to secure communication protocols, such as TLS, as they enable certificate-based authentication, digital signatures, and secure key exchange, enhancing the overall security and trustworthiness of digital interactions.

2.1.3 Common Vulnerabilities and Exposures (CVE)

CVE [4] is a standardised cataloguing system that provides unique identifiers for known security vulnerabilities and weaknesses in software and hardware systems. It works as a reference for identifying and keeping track of vulnerabilities, enabling organizations and individuals to assess potential risks, prioritise security patches, and take proactive measures to protect their systems from cyber threats. The CVE system improves collaboration among security researchers, vendors, and users, facilitating a more effective response to security issues and contributing to a safer digital environment.

The vulnerabilities are categorised and scored according to the Common Vulnerability Scoring System (CVSS). The current version is 3.1, released in June 2019. Every vulnerability has a vector string with information about attack characteristics, enabling efficient categorisation.

2.2 Selecting related research papers

The papers discussed in section 2.3 are a representation of the recent developments in the Internet measurement research area. All the papers discussed were published between 2010 and 2020. They are categorised in the following three subjects: TCP layer scanning, database scanning, and scanning research findings. The purpose of this categorisation is to ensure clear structure of the discussion and to cover all aspects of the proposed research. In each category we review the state-of-the-art research, carefully selected from the wide range of research papers available worldwide. In choosing the papers, we used these categories as a first selection criterion, a paper must fall into at least one of the three categories. Furthermore, to ensure the papers have been reviewed substantially and have contributed meaningful research, the second criterion is that a paper must be accepted at a reputable conference or journal. The reputation of a conference or journal stems from the

quality of the review process, the citations of published papers and the reputation of program committee members.

2.3 Discussion of the research papers

In this section we will discuss previous research that has been done in the topic of scanning the Internet for research purposes.

2.3.1 TCP layer scanning

To the best of our knowledge, two academically peer reviewed papers have been written that propose a new software program with the purpose of Internet-wide scanning [5] [6]. The first, written in 2010 by Derek Leonard and Dmitri Loguinov [5], proposes a new scanner called IRLScanner. Their main goals with the design were to be as polite as possible to scanned networks such that complaints from their administrators are greatly reduced, and to scan significantly faster than was ever possible before, reducing the time to scan the whole Internet from weeks to approximately 24 hours. IRLScanner relies on a custom network driver for the Windows operating system, IRLStack [7].

The second paper is from 2013, and describes ZMap, an open-source network scanner. ZMap scans the whole Internet in under 45 minutes from one machine utilising a gigabit Ethernet connection. It has a modular architecture, allowing users to write their own code to define actions to take once a port has been scanned. Packet sending and receiving runs in parallel to achieve maximum speed. While scanning, a random permutation of the IPv4 address space is used to make sure that networks are not overloaded. The permutation is generated with the guarantee that every IP address is unique, and sequential IPs are spread evenly. Excluding certain IP addresses from the scan can be easily configured. ZMap achieves its high speed of scanning by sending Ethernet-layer packets via a raw socket. Kernel overhead is prevented in this way, and by default TCP connections are automatically closed with a TCP RST packet, as no TCP session is known to the kernel. ZMap was used in many scans to demonstrate its usefulness, and in Table 2.2, guidelines for politeness in scanning the Internet can be found, as determined in this paper. ZMap is available for free and works on the Linux and macOS operating systems.

Two IP protocols are currently widely being used, IPv4 and IPv6. The previously discussed papers consider IPv4 address scanning. Since the IPv6 address space is far too large to allow for a full scan, new techniques are needed to mitigate this problem.

In 2017, Murdock et al. [8] investigated the generation of IPv6 addresses likely to be of interest while performing Internet scans. They explored what should be taken into account when designing such algorithms, and designed their own algorithm: 6Gen. It works with a set of initial ”seed” addresses, from which dense address space regions are discovered, and from there it generates addresses to scan. They evaluated the performance of 6Gen compared with Entropy/IP [9], which was the state of the art at the time of writing their paper. 6Gen performs between 1-8 times better when used with the same data sets in train-and-test runs.

In 2018, using a multitude of IPv6 address sources including Entropy/IP and 6Gen, Gasser et al. [10] explored the concept of IPv6 hit lists. They find that responsiveness of server and client IPv6 addresses is significantly different. Client addresses do not stay active for long, confining measurements to take place within minutes after hit list generation. Server addresses however remain active for weeks. They set up a IPv6 hit list service, a publicly available database with daily updated hit lists and aliased prefixes, enabling further research.

Another relevant paper, written by G. Wan et al. [11], analyses the effect of location on Internet-wide scans. They scanned the Internet from seven geographically and topologically different networks. While scanning from a single origin, on average 1.6–8.4% of HTTP, 1.5–4.6% of HTTPS, and 8.3–18.2% of SSH hosts are not scanned. They show that transient outages, permanent and temporary blocking, geographic biases and packet loss impact scans. They recommend to scan from 2-3 diverse origins to prevent transient loss, and encourage researchers to use multiple source IP addresses, and/or probing addresses multiple times with a delay in between. Scanning from only one location still yields significant results, but it is important to keep potential geographical bias in mind.

In 2015, Durumeric et al. [12] published their paper in which they introduce Censys, a search engine with data collected through Internet-wide scanning. Censys is available online for the research community to conduct research which would otherwise require additional scans. The paper describes 16 protocols regularly scanned and stored. Only standardized ports of those specific protocols are scanned.

Izhikevich et al. [13] published their paper on identifying unexpected Internet services in 2020. Following observations from previous work that a significant portion of Internet hosts might be running on different ports then the default IANA-assigned (Internet Assigned Numbers Authority) ports for the services they run, they find out that many services run on a widely spread out range of ports, and that it is more likely that they are less secure then the same service running on the designated IANA-assigned port. They design a scanner called LZR, which is capable of fingerprinting 99% of identifiable services running on unexpected ports with only 5 handshakes, and 88% with just one single packet. Furthermore, because scanning

all 65535 ports on all IPv4 addresses is not currently feasible, and there is no evidence for any subset of ports to be clearly valuable to scan for a particular purpose, they recommend to scan a small amount of IP addresses on all ports to determine a representative subset of ports of the protocol(s) to be researched. LZR takes an IP/port list or stream of SYN-ACKS from ZMap, and a list of protocols to identify. With this it sends a single packet to each of them, using a raw socket, similar to ZMap.

2.3.2 Scanning of databases

A prior study conducted, similar to the scan proposed in this paper, by Ferrari et al. [14], conducted a large-scale analysis of NoSQL service configurations. They contribute an approach to investigate misconfigurations in NoSQL services while not compromising the integrity of the databases. They created a tool that can automatically scan the IP addresses of cloud service providers to find publicly accessible MongoDB, Elasticsearch, Redis and Cassandra databases. They leveraged their tool to study the configurations and reflect on the security and privacy implications. First they listed the IP addresses of cloud storage providers, then scanned those with Nmap [15] for open ports. Lastly they analysed the found publicly accessible services by sending commands that count all accessible data instances, and commands that try to create a new section in the database in which they write a message stating that the database is insecure and information about the research. The ethical dilemmas this research poses are addressed extensively.

An investigation into the configuration of Amazon S3 cloud storage containers (buckets) , was conducted in 2018 by Continella et al. [16]. Amazon's access policy configuration is quite extensive, and can be misconfigured, possibly leading to a variety of security and privacy issues. They created a tool that can check Amazon S3 buckets automatically for vulnerabilities, verified their tool, analysed the security performance of the bucket ecosystem, and developed a browser extension that protects Internet users from loading potentially malicious resources from publicly writable buckets. S3 buckets are identified with a unique name between 3 and 64 characters, and to create a list of candidate bucket names they created numerous mutations and enumerations from words in the English dictionary. Together with a list of known public bucket names they run a scan, and find 240,461 existing buckets, of which 27,492 are readable, and 6599 writable. They also identified 191 websites that are vulnerable to code injection in writable S3 buckets. Their browser extension uses a hashed list of vulnerable buckets to protect users from exploiting the list. Furthermore, they present a thorough overview of ethical issues and how they dealt with them.

Additional research on cloud storage configuration conducted by Cable et all. [17] focused on multiple bucket storage providers, improving on the work done by Continella et al [16]. They adapt password cracking algorithms to guess bucket names, and create their own cloud scanning tool named Stratosphere that uses those algorithms to generate candidate bucket names, trained with a large amount of bucket names gathered from high quality passive DNS measurement data. Stratosphere relies on training with high quality training data only available to verified researchers to avoid abuse. They evaluate the performance of Continella's tool and an online system called GrayHat [18] with Stratosphere and find that both are biased towards a small portion of the bucket name space as they find relatively short names. They find that the vulnerability of Amazon S3 buckets is a factor 5 times worse than Continella estimated, and also worse than the vulnerability of other bucket providers. They hypothesise that the cause is Amazon's complex permissions model. Furthermore, they show that the problem of finding bucket names is fundamentally limited by unguessable randomness added to bucket names.

2.3.3 Research findings from Internet-wide scanning

Since the creation of ZMap [6] and Masscan [19], numerous studies have used scanning as their methodology for the research carried out.

Springall et al. [20] used ZMap to measure HTTPS behaviour for their research on the use of cryptographic shortcuts in Transport Layer Security (TLS). These shortcuts reduce overhead computation on the server and decrease the latency for the client side, but also significantly reduce the security of forward-secret cryptography. They modified ZMap to work with session IDs and session ticket resumption for their research. They showed that 38% of the Top Million HTTPS websites are affected such that they can be compromised up until 24 hours after cryptography took place, and 10% even up until 30 days. The chosen cryptographic cipher does not influence this vulnerability.

Another security focused paper by Amann et al. [21] used active and passive measurements to investigate the security state of HTTPS, particularly the adoption of recently developed features like HTTP Public Key Pinning (HPKP), Certificate Transparency (CT) and HTTP Strict Transport Security (HSTS). For their active measurements they aggregated a total of 193M domain names from various sources, and resolved those to IP addresses. An IPv6 capable version of ZMap is used to port scan IPv6 addresses, and Goscanner [22] is used to execute TLS handshakes. They find that there might be a correlation between deployment of new features, and configuration difficulty and risks to site availability.

To study the values of TCP's initial congestion window (IW), an important pa-

rameter for performance in the Internet, Rüth et al. [23] adapted ZMap to be able to make TCP connections, and to remember properties of the connection like segment length, to allow calculating the IW. They show that scanning only 1% of the IPv4 Internet at random is enough to get representative IW data for the whole Internet. Their main contribution is to bring data to the debate about what size the IW should be. The trend observed in the evolution of services on the Internet is that the IW is adapted per service and thus becoming less and less a static constant within the Internet traffic.

Durumeric et al. [24] investigate the HTTPS certificate ecosystem in 2013. They conducted 110 Internet-wide scans over 14 months to obtain insights into the practices and adoption of security standards of certificate authorities (CAs). Scanning the entire IPv4 address space is done with ZMap to discover hosts with port 443 open to TCP connections. Then a TLS handshake is performed using OpenSSL, and the obtained certificates validated with custom emulated browser validation. The ethical nature of the research is addressed extensively. They find that CAs lack behind with the implementation of best practices developed by the security community. In November 2011 the CA/Browser Forum, a cooperation between CAs and Browser makers, developed guidelines to deal with many certificate security risks. However, lacking any enforcement, these guidelines have not been widely implemented. Browser support for new security technologies is also a significant part of the problem. Compatibility between CAs and browsers is need for both to function correctly, and neither has really the incentive to adopt a new technology first, since the other party needs to update as well before it achieves any value. Lastly, they found more than 50 root authorities using a key size of 1024-bit for RSA, with the latest expiry date set at 2040. They recommend that more awareness about long term security consequences should be raised, since 1024-bit RSA keys were recommended to use until 2020.

After the discovery of the major security vulnerability in OpenSSL called Heartbleed in 2014, Durumeric et al. [25] conducted Internet-wide scans to find vulnerable hosts, notify them of their security issue, and monitor the patch rate among notified hosts. They use a non-intrusive way to detect the vulnerability in hosts, exploiting the fact that vulnerable versions of OpenSSL reply to Heartbeat requests without any payload or padding, and the length field set to zero. The patched version of OpenSSL and other similar libraries deal with such incoming requests correctly as specified in RFC 6520 [26]: namely to drop the packet. They modified ZMap to send such empty HeartBeat requests to find all vulnerable hosts. The analysis of the resulting data revealed that certificate management requires a good understanding of the security ecosystem and the protocols used, which many administrators do not possess. Furthermore, the mass certificate revocation as a result of Heartbleed

caused CAs to be overloaded, and stresses the need for new methods to make revocation more scalable. Additionally, a need for a well streamlined mass vulnerability disclosure follows the chaotic fashion with which Heartbleed was disclosed, leaving a number of major websites uninformed about Heartbleed for more than 24 hours after it was disclosed. Lastly, while most sites patched within two weeks after disclosure, the patch rate plateaued after that. The global awareness for Heartbleed was not enough to make all hosts affected patch the problem. All vulnerable hosts found during the scanning phase were notified of being vulnerable, and this led to the patch rate increasing with 47%.

Holz et al. [27] used Internet-wide scanning as a data source for their research on the deployment of TLS 1.3 in 2020. The adoption of TLS version 1.2 took over 5 years, and with version 1.3 arriving, they took the opportunity to track the deployment, over a long period starting at the early design phase until a year after standardisation finished. They collected a huge number of domain names from different sources, to create a set consisting of many different subsets to be able to compare the differences in TLS 1.3 deployment between them. They scan periodically, starting in 2017-10 from Aachen in Germany, and shift scanning to Sydney in Australia in 2019-05, after TLS 1.3 was standardised. They use massdns, ZMap and ZGrab, as well as Goscanner [22], as published by Amann et al. [21]. Furthermore, they use passive monitoring data of TLS traffic in North America and a data set from Android application's use of TLS. They find that TLS 1.3 is widely deployed quickly after standardisation, due to an extended phase during which organizations with control over both client and server implementation (like Google and Facebook) could experiment relatively risk-free, and the increased centralisation of the Internet, which makes that if a few large providers activate TLS 1.3, a great portion of the Internet traffic goes via TLS 1.3. They also highlight the importance of a varied data set, limits of passive and active measurements, as well as platform limits can be overcome by combining the data.

2.4 Ethical discussion

Scanning the Internet is subject to ethical concerns. In Table 2.2, recommendations for good citizenship are listed, as described in [6]. These guidelines are very important to keep in mind with Internet measurement research. Previous work on the ethics of network measurements states that the impact on normal Internet users should be minimal [28]. From a scanning perspective, this means that scanning should not be aggressive, and be very clear in describing the purpose and methods of the scans, to not overload network administrators. Craig Partridge and Mark Allman [29] state that all network measurement research should include an ethics

section, and all papers discussed previously indeed discuss the ethical nature of their research. In Table 2.1 an overview of what each paper mentions can be found. Note that Leonard et al. [5] published their research before the ZMap paper existed.

Our research follows all the best practices in Table 2.2, aiming to be as polite as possible, and to only read information that should be publicly accessible. We perform a banner grab on discovered public databases to collect data, which is not published, and only used for research purposes. Any complaints that came in, were handled promptly and adequately.

	Cite [6]	1	2	3	4	5	6	7
Leonard *et al.* [5]	○	●	◐	○	○	●	○	●
Gasser *et al.* [10]	●	○	○	●	◐	◐	○	○
Wan *et al.* [11]	●	○	○	●	◐	●	◐	●
Durumeric *et al.* [12]	●	●	●	●	●	●	●	●
Izhikevich *et al.* [13]	●	○	◐	○	○	●	●	●
Cable *et al.* [17]	●	○	○	○	○	◐	○	○
Springall *et al.* [20]	●	○	○	○	○	◐	○	○
Amann *et al.* [21]	●	○	○	●	●	◐	○	○
Rüth *et al.* [23]	●	○	○	●	◐	◐	◐	●
Durumeric *et al.* [24]	●	●	●	●	●	●	○	●
Durumeric *et al.* [25]	○	○	○	○	○	◐	○	●
Holz *et al.* [27]	●	○	○	●	○	●	●	●

Table 2.1: An overview of ethical information present in papers. The first column depicts if the paper cites the ZMap paper [6] in an ethical context, and the numbered columns refer to the practises listed in Table 2.2. ● Means fully described, ◐ means mentioned and ○ means not explicitly mentioned.

2.5 Our research in perspective with previous work

For conducting our research, ZMap is more suitable then IRLScanner. The primary reason is because ZMap is significantly faster in scanning the Internet, 45 minutes vs 24 hours, if used with a Gigabit Ethernet connection. Another scanner, Masscan, was also developed [19] and achieves the same speed as ZMap. Masscan however, was not developed by the research community and does not have a scientific paper behind it. Furthermore, all research discussed in this paper regarding IPv4 scanning uses ZMap, and many modifications have been developed that can be utilised.

Currently, IPv4 is still the most used Internet Protocol, and represents a good

1. Coordinate closely with local network admins to reduce risks and handle inquiries.
2. Verify that scans will not overwhelm the local network or upstream provider.
3. Signal the benign nature of the scans in web pages and DNS entries of the source addresses.
4. Clearly explain the purpose and scope of the scans in all communications.
5. Provide a simple means of opting out, and honor requests promptly.
6. Conduct scans no larger or more frequent than is necessary for research objectives.
7. Spread scan traffic over time or source addresses when feasible.

Table 2.2: Recommended practices for good Internet citizenship while conducting fast Internet-wide scans [6].

overview of the Internet. While IPv6 is already in use, its scanning is still an ongoing research area, and thus our research will focus on scanning the Internet over IPv4.

Our scan is conducted from one geographical location, for several reasons explained in Section 3.3, following the research findings of G. Wan et al. [11] that this will yield significant results.

We take a similar approach to Internet scanning as all described previous scanning work, where we take ethics very seriously into account, and gather and analyse data with the same tools and mindsets. The expectation is that we will find many publicly accessible database instances, and that security is not always taken into account while setting up the database server.

Chapter 3

Methodology

The process of scanning the world for running database services begins with selecting target databases and determining their default listening Transmission Control Protocol (TCP) ports. Next, all reachable IPv4 addresses are generated and scanned to identify open ports associated with databases. This initial scan produces a list of IPv4 addresses that require further investigation to determine the existence of publicly accessible databases.

3.1 Database services

Many database solutions are in existence, and to investigate them all would exceed the scope of this research. To identify the most popular and widely deployed databases, we refer to the database management system ranking provided by DB-Engines [30]. The scanning software we use, Zgrab2 [1], has implementations for the top 6 databases in the ranking: Oracle, MySQL, Ms-SQL, PostgreSQL, MongoDB and Redis. They are listed in descending order, and have been the top 6 in the same order for years. Zgrab2 is written in Go, a programming language known for its speed, support for concurrency and is designed for the internet. In the scan findings pertaining to MySQL, we identified the inclusion of ”MariaDB” in many version numbers, indicating the presence of a MariaDB server. As a result, we opted to segregate the MySQL and MariaDB instances for individualised examination.

Additionally, since our research aims to include the development of a database scanning module, we chose one more database to investigate. We find it essential to include the wide-column database Cassandra in our investigation. Despite ranking 11th in August 2022, Cassandra holds the highest position among wide-column databases. Notably, its deployment by major companies like Apple and Netflix, coupled with the availability of a Go library to integrate with the Zgrab2 module, strengthens our decision to add Cassandra to our security research. As a result,

we have implemented an additional Zgrab2 module specifically tailored for scanning Cassandra databases.

The following is a short description of the 8 investigated databases:

- **Oracle:**
  Default TCP port: 1521
  Oracle Database is a proprietary and security feature-rich relational database management system. It is widely used in enterprise-level applications and supports large-scale data processing and complex transactions.

- **MySQL:**
  Default TCP port: 3306
  MySQL is an open-source relational database management system (RDBMS) known for its speed, reliability, and ease of use. It is commonly used in web applications and is compatible with various programming languages.

- **Ms-SQL (Microsoft SQL Server):**
  Default TCP port: 1433
  Microsoft SQL Server is a popular relational database management system developed by Microsoft. It offers a wide range of features, including data warehousing, business intelligence, and high-performance data processing.

- **PostgreSQL:**
  Default TCP port: 5432
  PostgreSQL is a powerful open-source object-relational database system known for its flexibility and support for advanced data types. It is commonly used in web applications and data-driven projects.

- **MongoDB:**
  Default TCP port: 27017
  MongoDB is a widely used NoSQL document-oriented database known for its flexibility and scalability. It stores data in JSON-like documents and is often used in modern web applications and big data projects.

- **Redis:**
  Default TCP port: 6379
  Redis is an in-memory data structure store often used as a cache, message broker, or for real-time analytics. It is known for its speed and versatility, making it popular in various applications.

- **Cassandra:**
  Default TCP port: 9042

Apache Cassandra is a wide-column distributed NoSQL database known for its ability to handle large amounts of data across multiple servers, providing high availability and fault tolerance. It is widely used in big data and real-time applications.

- **MariaDB:**
  Default TCP port: 3306
  MariaDB is an open-source relational database management system (RDBMS) that originated as a fork of MySQL. It aims to offer enhanced performance, reliability, and features while remaining compatible with MySQL.

3.2 Ethical considerations

All recommended practices from Table 2.2 are taken into account for the scans. A scanning speed of 30 Mbps is used for the Zmap scan in accordance with the local network admins and upstream provider, to not overwhelm their networks. The benign nature of the project is explained in a simple web page running on the source address, with a clear way of opting out. Scans are not run during weekends in any part of the world. A blocklist is used which contains all IP addresses of people and organisations who opted out of being scanned. The scans are of such nature that only publicly accessible data is read and no write operations are performed.

3.3 Active scan

To ensure an efficient and streamlined active scan, we have chosen to run the scan from a single IP address and one geographical location. This approach allows us the advantage of setting up a simplified and consistent single server infrastructure. By centralising our scanning operations, we can reduce complexity and ensure consistency and reliability of our results. In line with those requirements, we have established a dedicated Linux server, equipped with sufficient broadband network capacity. The server is located in Sydney due to a collaboration with Sydney University for this project. We maintained regular contact with the local network administrators facilitating seamless communication and coordination during the scanning process.

The scans are run with a bash script incorporating all necessary steps. To avoid scanning IPv4 addresses that aren't in use, we employ Pyasn [31]. Pyasn provides a list of all currently reachable IPv4 prefixes. It does this by obtaining the current border gate protocol (bgp) routing information base (rib) from RouteViews [32] and extracting the reachable IPv4 addresses from it. Then, for every port to scan, Zmap [6]

is invoked, doing a TCP syn (synchronise) scan resulting in a list of IPv4 addresses with the specified port open. Using this list, Zgrab2 [1] scans for a running database to connect to, and records findings as JSON encoded data. Zgrab2's implementation is limited at the time of utilisation to TLS versions up to TLS 1.2, hence TLS 1.3 is not investigated in this study. All internet traffic is captured with TCPDump [33].

We develop a Zgrab2 module to scan the Cassandra protocol, using the GoCQL library [34]. The Zgrab2 module creates an instance of GoCQL configured such that it will initiate a connection to the open port designated for Cassandra at the target IPv4 address. The GoCQL code is modified to collect all public configuration data the server has to offer and return it to the Zgrab2 module. In order to have the same consistent TLS fields recorded for TLS connections as all other Zgrab2 database scanning modules, the cryptographic TLS library in GoCQL is replaced with the Zcrypto [35] TLS library.

Table 4.1 provides an overview of the scanned databases, their corresponding outcomes, and the number of TLS encrypted connections. It's important to highlight that the Cassandra scan conducted in 2021 exclusively scanned for databases that offered TLS encrypted connections. However, the scan performed in 2023 includes both encrypted and unencrypted connections. Additionally, the scan conducted in 2023 includes an assessment for the Heartbleed vulnerability in the servers offering TLS. This check was not implemented in the earlier 2021 scan. In all other aspects the scans are performed and implemented identically.

3.4 Data analysis

We store the gathered server configuration data on the server in Sydney, and analyse security aspects. Analysing is done using Python and Bash scripts that refine, count and validate our data.

3.4.1 TLS versions

In the TLS handshake, the TLS version is agreed upon by client and server. Our client offers the server the choice of all existing protocols, and records the chosen version by the server. We aim to map the usage of TLS versions to analyse the security provided by the database servers.

3.4.2 TLS cipher suites

The servers offering TLS do so by advertising their supported cipher suites, which consist of algorithms and key lengths to protect communication. During our scans,

our client offers all cipher suites available in the agreed upon TLS protocol, which includes all supported weak and broken ciphers. The server will choose which suite will be used for encryption. In this way we can measure the level of security the server chooses by default. We collect the agreed upon cipher suites used to encrypt our scan traffic, and compare them to the security current best practices set out in RFC 9325 [36]. To facilitate this, we aggregate them into the categories described in the RFC: 'MUST NOT', 'SHOULD NOT' and 'RECOMMENDED'.

3.4.3 Heartbleed

The Heartbleed vulnerability has been around since April 2014, and Zgrab2 features a non-invasive check if a TLS server is vulnerable. To the best of our knowledge of Zgrab2's implementation the check for Heartbleed is exhaustive. In the first scan in October 2021 this feature was not enabled, however, in the second scan in March 2023 we did enable it, allowing us to ascertain the impact this vulnerability has on database servers in 2023.

3.4.4 X.509 certificates

Our investigation encompasses an examination of the certificates employed by servers to establish their identity. For this purpose, we make use of OpenSSL [37]. To ensure the reliability of these certificates, we rely on the Common CA Database (CCADB) [38], administered by Mozilla, which serves as a repository of trusted root certificates. The CCADB is the leading CA repository initiative [39], utilised for example by Microsoft (Microsoft Trusted Root Certificate Program) and Google (Google Chrome). Note that while the CCADB is the source of root stores for large companies, there is no obligation to strictly use all certificates in the ccadb store, nor restrictions to include additional certificates as deemed necessary. With a bash script, we invoke OpenSSL's "verify" function and configure it to assess the certificates' legitimacy and security attributes.

The core objective of this endeavour is to acquire valuable insights into the extent to which the identities of databases are trustworthy. By assessing the certificates against a reputable source of trusted roots, we aim to obtain the authenticity and security posture of the server certificates. This process aids us in comprehending the level of confidence that can be placed in the identities asserted by these databases, contributing to an extensive assessment of their security mechanisms.

3.4.5 Software versions

Our research encompasses an examination of the software versions installed on the database servers. The scanning procedure involves identifying whether a server exposes its software version to any connecting client, and subsequently records this information.

This data-set allows us to conduct a comparative analysis between the 2021 and 2023 scans for machines responding on the same IP address and port. We cannot say with 100% certainty that it was the same machine and database server in both years, but it is unlikely that a different person or organisation obtaining such an IP address would use it to run the same type of database server on the same port less then 2 years later. By uncovering any version alterations that occurred within this time-frame, we gain valuable insights into the maintenance and update practices of open databases, or the evolution of the usage of database solution software versions. Servers responding on the same IP address and port running the same database software after a 1 year 7 month time period are likely to be the same server or employed within the same data center environment. Version numbers comprising of a major, minor and patch level are extracted from the version string recorded by Zgrab2 to enable this comparison.

In addition, we undertake an in-depth exploration of the software versions themselves. To achieve this, we use CVE-search [40], enabling us to cross-reference version numbers against the CVE database. CVE-search operates on inputs with a product name and a version number consisting of a major, a minor and a patch number, for example ”mariadb:10.9.2”. We extract these version numbers from the version string recorded by Zgrab2, and only search the CVE database with versions from which we can clearly identify major, minor, and patch numbers. This step provides an overview of the vulnerabilities associated with specific software versions and their susceptibility to remote cyber attacks. Our focus will be on vulnerabilities that can be exploited from anywhere on the internet, using regular expressions to filter the vulnerabilities. We take older vulnerabilities categorised with CVSS version 2 into account as well, matching them into the equivalent categories under version 3.1. With the characteristics and impact scores of the vulnerabilities we make the vulnerability of publicly accessible database servers tangible and comparable. This evaluation aids us in appraising the security stance of publicly accessible databases, enhancing our comprehension of their vulnerability to potential threats.

Database	#DB's 2021	#DB's offering TLS	#DB's 2023	#DB's offering TLS
Oracle	214	214 (100%)	9 (-96%)	9 (100%)
MySQL	1,475,271	923,503 (63%)	1,556,114 (+5%)	1,012,351 (65%)
Ms-SQL	357,665	357,665 (100%)	375,601 (+5%)	375,601 (100%)
PostgreSQL	873,734	326,438 (37%)	767,887 (-12%)	206,551 (27%)
MongoDB	77,796	-	94,435 (+21%)	-
Redis	99,050	-	176,553 (+78%)	-
Cassandra	24[1]	24	1,853[1]	9 (0.5%)
MariaDB	461,786	25,382 (5%)	558,314 (+21%)	81,654 (15%)

Table 4.1: Number (#) of databases found in 2021 and 2023. MongoDB and Redis scan implementations do not support TLS handshakes.

[1] Scanning for non-TLS servers was not implemented in the 2021 Cassandra scan.

Chapter 4

Results

The outcomes of both conducted scans are presented in Table 4.1, showcasing the number of databases detected during each scan. Beyond mere database counts, the study's focus is on identifying security-related insights. This section details the noteworthy security findings obtained from the scans, shedding light on potential vulnerabilities, configurations, or trends observed in the database landscape. In total, we found 3,345,540 databases in 2021, and 3,530,766 in 2023, a 6% increase over 17 months. Calculated using only the database statistics which included TLS scanning (not using the MongoDB and Redis database counts), we find that in 2021, 52% of servers offer TLS, which decreased slightly to 51% in 2023. Our Zmap TCP scan results are in Table 4.2, which details for each database port the amount of IP addresses responding to the TCP syn scan.

Database	Port open 2021	#DB's 2021	% 2021	Port open 2023	#DB's 2023	% 2023
Oracle	4,849,667	214	0.00%	5,325,271	9	0.00%
MySQL	7,160,153	1,937,057	27.05%	7,320,985	2,114,428	28.88%
Ms-SQL	3,106,800	357,665	11.51%	3,565,017	375,601	10.54%
PostgreSQL	6,222,291	873,734	14.04%	6,062,813	767,887	12.67%
Mongodb	2,746,427	77,796	2.83%	3,370,439	94,435	2.80%
Redis	3,438,856	99,050	2.88%	3,539,541	176,553	4.99%
Cassandra	5,194,578	24	0.00%	5,232,862	1,853	0.04%

Table 4.2: The number of IP addresses with a machine responding to our Zmap TCP syn scan on the default database ports, and the percentage of those ports which run the expected database service.

4.1 TLS

4.1.1 TLS versions

Figure 4.1 illustrates the preference for TLS versions among various database servers. All versions older then TLS 1.2 are no longer considered secure due to their lack of support for currently recommended cryptographic algorithms and protocols [2].

In 2021, nearly half (49.7%) of Ms-SQL servers opted for a deprecated TLS version to secure their connections. By 2023, this percentage had slightly decreased to 45.2%. This may be a consequence of the default settings which enable TLS 1.0 and TLS 1.1 in SQL Server 2019 and older versions with TLS support [41]. Administrators need to actively disable older TLS versions to bring their servers up to the latest security standards.

With 100% TLS 1.2 in 2021 and 22.2% deprecated versions in 2023, Oracle stands out as well, but please note that we found very few Oracle installations, which means we cannot directly compare these numbers.

In contrast, other database servers displayed relatively far fewer (4.9% and lower) instances of using outdated TLS versions in 2021, and this number further declined by 2023 to at most 2%.

For a comprehensive breakdown of the TLS versions recorded during our scan, please refer to Table A.1 in Appendix A.

4.1.2 TLS Cipher suites

In Figure 4.2 is a bar chart with the cipher suite chosen by the database server to encrypt the connection. In order to present the results in a compact way, we de-

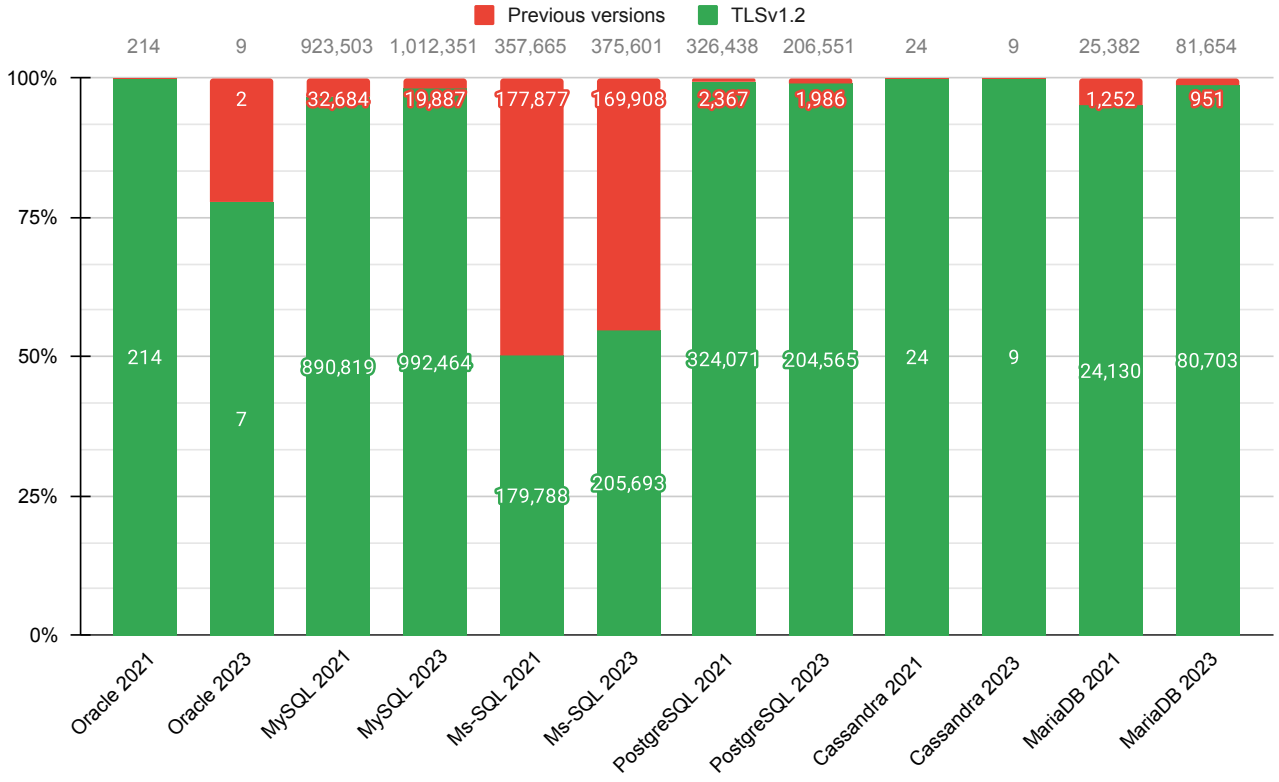


Figure 4.1: TLS versions found in 2021 and 2023.

cided to group the cipher suites in terms of their security, according to the latest best practices described in RFC 9325 [36]. The full specification of the categories is described in RFC 9325 and the meaning of the capitalised words is described in RFC2119 [42]. In appendix A in Table A.2 are the individual cipher suite occurrences.

- **MUST NOT:** Cipher suites in the 'MUST NOT' category offer less than 112 bits of security or contain the RC4 cipher.
- **SHOULD NOT:** Cipher suites in the 'SHOULD NOT' category offer less than 128 bits of security or do not have forward secrecy.
- **RECOMMENDED:** Cipher suites in the 'RECOMMENDED' category have forward secrecy and offer 128 or more bits of security.

We see differing results for the different databases. For Ms-SQL we observe a trend that is in line with its TLS version distribution, namely a 61.3% combined SHOULD NOT and MUST NOT cipher suite usage in 2021, slightly reduced to 53.5% in 2023. This could be correlated with the amount of deprecated TLS versions found, as those do not offer RECOMMENDED cipher suites. PostgreSQL shows a remarkable decrease in SHOULD NOT cipher suite usage from 50.5% in 2021 to 0.5% in 2023. In stark contrast, the MUST NOT cipher suite selection from MariaDB servers, which grew from 20% in 2021 to a very concerning 59.5% in 2023. MySQL shows 86.5% and 84.3% RECOMMENDED cipher suites in 2021 and 2023 respectively. Oracle and Cassandra have very few servers opting for non RECOMMENDED cipher suites.

4.1.3 TLS Heartbleed

In March 2023, a total of 307 PostgreSQL, 1 MariaDB and 6 MySQL servers are found to be vulnerable to the Heartbleed vulnerability [25]. That is almost nothing in all cases. This is in stark contrast with the massive numbers of online servers using a vulnerable version of OpenSSL in 2014 when Heartbleed was discovered. We additionally compare our results to more recent Heartbleed data from November 2020, when a researcher from the SANS Internet Storm Center investigated a number of high-impact vulnerabilities including the Heartbleed vulnerability and found over 200,000 systems affected by Heartbleed worldwide [43]. With a total of 314 vulnerable servers, which is 0.019% of the 1,676,175 database servers offering TLS found in the 2023 scan, the impact of Heartbleed on database servers is significantly small with respect to the 200.000 found by the Storm Center researcher in 2020.

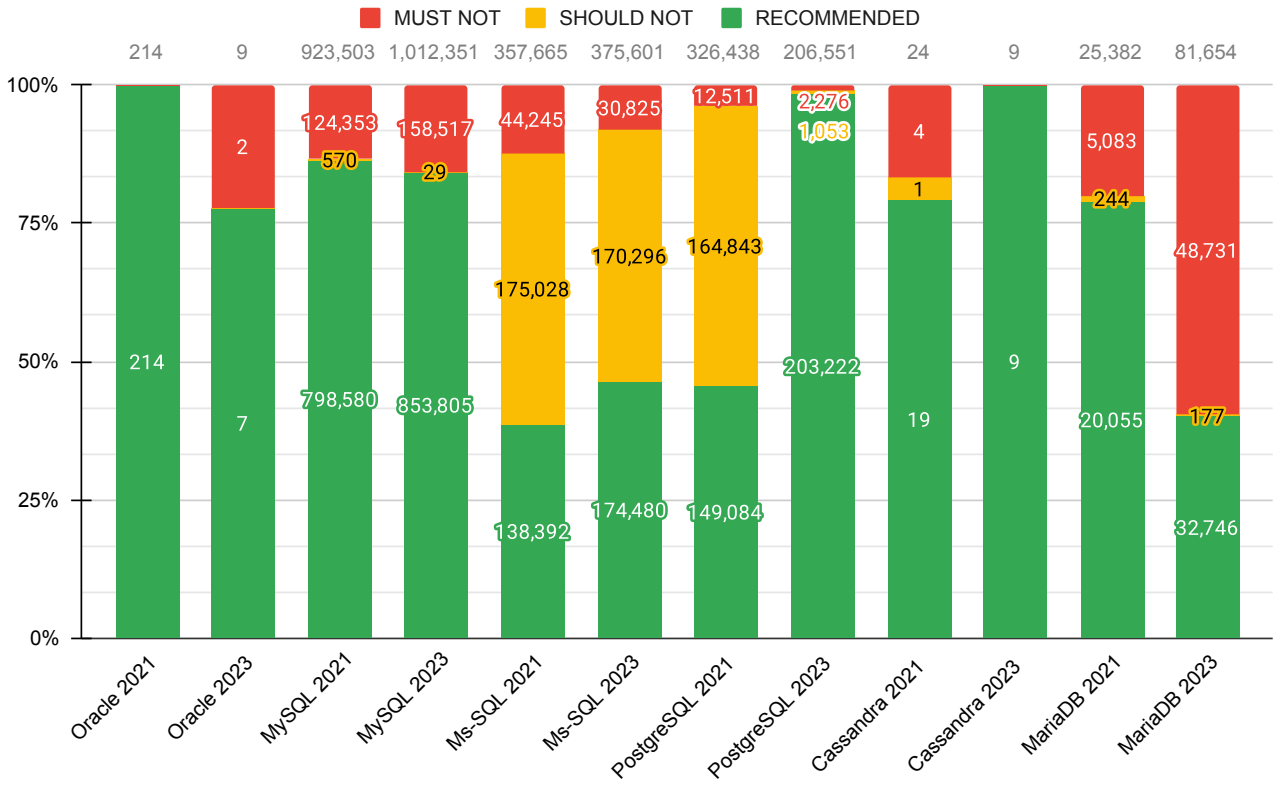


Figure 4.2: TLS cipher suite usage in 2021 and 2023, categorised in accordance with the classification for cipher suites in RFC 9325 [36].

4.1.4 TLS Certificates

Certificates offered by the servers to identify themselves are important for security. In Figure 4.3 are results of running the openSSL verification tool [37] with the CCADB [38] certificate authority information as trusted roots. The full validation results can be found in appendix A in Table A.3.

We use the security definitions of OpenSSL verify [44] for our classification. Please refer to the citation for the full description of how the security levels are determined.

- **Unable to get local issuer certificate:** No trusted root certificate could be found within the CCADB root store to verify the integrity of the certificate.
- **Self signed certificate:** The certificate is signed with a private key belonging to the server owner. There is no guarantee the key has never been compromised.
- **Security levels:** Security level 2 is set to 112 bits of security. 'Certificate key too weak' means that at least one of the keys used does not provide 112 or more bits of security. This can either be one of the server's keys or the CA's keys. Level 3 is set to 128 bits and all ciphers are required to offer forward security. For level 4 a minimum security of 192 bits is necessary.

The results of the certificate validation do not indicate database administrator effort to provide trustworthy database identity certification. 78% of all certificates across databases are either self signed or unable to get local issuer certificate in 2021, which slightly worsened to 82% in 2023. The only database with a highly significant positive development is MariaDB, which went from 27.8% verifiable valid certificates in 2021, to 95.4% in 2023. For MySQL, Ms-SQL and PostgreSQL we see slight increases in valid certificates, from 2.7& in 2021 to 10.1% in 2023 for MySQL, 5.6% in 2021 to 6.5% in 2023 for Ms-SQL and 1.3% in 2021 to 5.8% in 2023 for PostgreSQL. Cassandra servers offered self signed certificates exclusively in 2021, as well as in 2023 with the exception of 1 certificate giving the error of unable to get local issuer certificate.

Oracle is an outlier in 2021 with 96.3% valid certificates, but 0% in 2023. Due to its 96% reduction in publicly accessible databases from 2021 to 2023, it has almost exactly the same amount of self signed and unable to get local issuer certificates in 2023 as in 2021, with an absence of any verifiable valid certificates in 2023. However, none of these IP addresses were seen in both years so we cannot say they are the same servers.

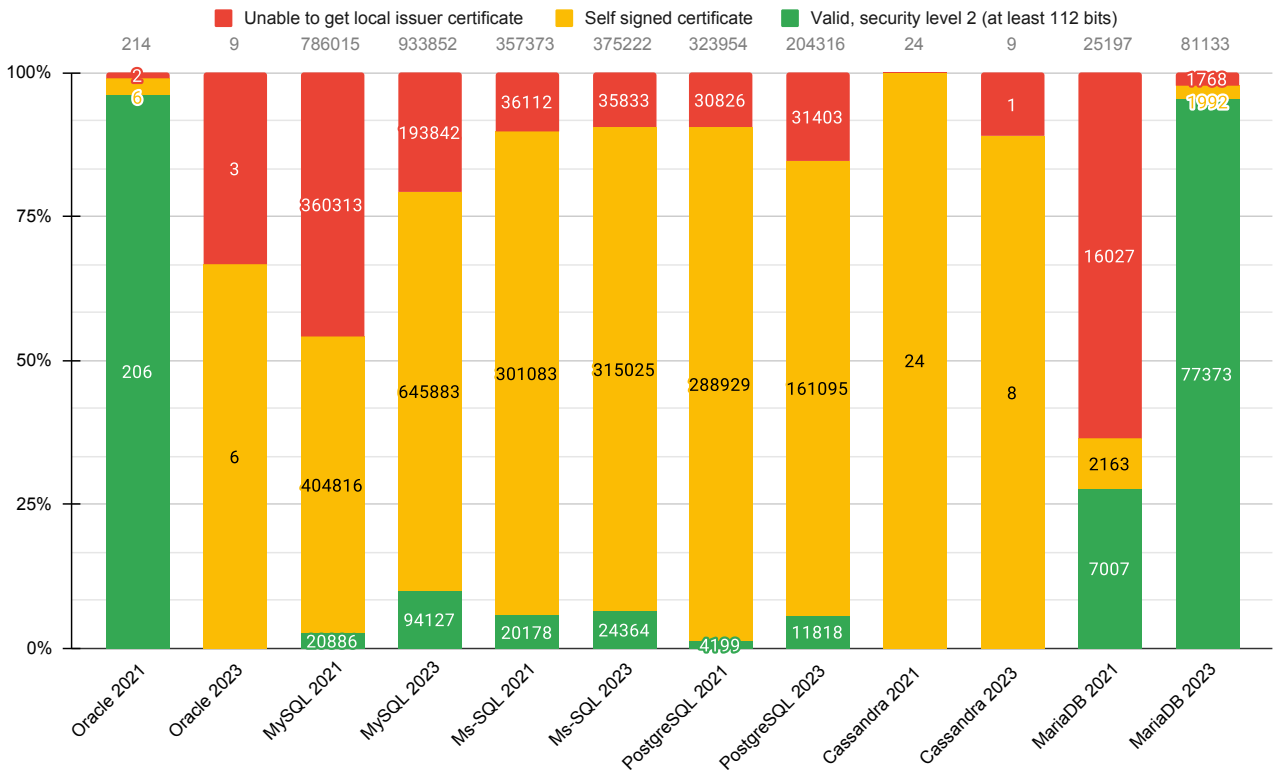


Figure 4.3: Validation of server certificates.

4.2 Database software versions

We acquired software version information for MySQL, Ms-SQL, MongoDB, Redis, Cassandra and MariaDB. Accurate extraction of the major, minor and patch level from the version string is found to be possible for nearly all servers responding with their version string.

Ms-SQL server version strings do not contain information on whether vulnerability fixes have been installed, which makes it not possible for us to determine if a version is affected by vulnerabilities.

Not all Redis servers revealed their version information, possibly due to the fact that Redis administrators can customise commands. The INFO command which our scan used to collect information about the server did not get a response from 82.6% of Redis servers in 2021 and 84.7% of Redis servers in 2023. Additionally, a total of 57 Redis servers run the Redis ”unstable” version, which is 999.999.999 in 2021, and in 2023 this amount has increased to 63. ”Unstable” means that the main development branch at the time of installing is running on the Redis server.

PostgreSQL and Oracle servers did not disclose their version information to us. PostgreSQL requires authentication before the version can be read. In Oracle's case, we found that all databases in both scans reported a version of 0, which does not disclose any useful information for an attacker. However all of them did disclose their TNS (Transport Network Substrate) listener version number (VSNNUM), with which attackers could possibly calculate the actual version number of the server software. This vulnerability in Oracle servers [45] has a medium severity, and the only solution is to restrict access to the database from remote to allowlisted IP addresses only. This might be a reason why we see so few Oracle servers publicly available on the internet.

4.2.1 Remotely exploitable vulnerabilities

The scatter plots in Figure 4.4 detail the software versions we encountered with the amount of servers deploying that version, plotted against the number of remotely exploitable (AV:N) vulnerabilities present in that version. Their respective average impact scores are plotted in Figure 4.5. To make the data comparable between database solutions, we have included scatter plots with the percentage of servers per version on the x-axis in figures 4.6 and 4.7. The total number of servers used to calculate the percentage of servers per version is the total number of servers responding with a version string. Which in the case of Redis means that the aforementioned 82.6% of Redis servers in 2021 and 84.7% of Redis servers in 2023 that did not disclose their version string are not included in the total Redis server

count used to calculate the percentage values. This ensures a comparison across database solutions of all servers identifiable as vulnerable by their version string.

We utilise the CVSS [46] to categorise the found vulnerabilities for a better comprehension of how vulnerable servers are to specific attacks. Below are the terms used in this report explained. Please refer to the citation for the complete CVSS specification.

- **Attack Vector (AV)** is the context within which the vulnerability can be exploited. We only consider attacks from the worldwide internet network for our study, notated as AV:N
- **Attack Complexity (AC)** is how difficult or easy the attack is to perform, for example what other conditions need to be present or controlled. There are two metrics in this category: AC:L means there is no specialised conditions necessary for the attack to succeed, low complexity. AC:H however, means that the success of the attack is not guaranteed, conditions outside of thea attackers control need to be right, high complexity.
- **Privileges Required (PR)** comes down to whether the attacker needs to be authenticated to perform the attack. PR:N means no authorisation is needed, for PR:L the attacker needs to have basic user rights, and for PR:H administrator rights are necessary for the attack. We use the PR:HL notation to indicate vulnerabilities that need at least some form of authentication.
- **Impact scores** are indications of how severe a vulnerability is when it is exploited, relative to other vulnerabilities. A score between 0.1 and 3.9 is considered low severity, 4 to 6.9 is medium, 7 to 8.9 means high, and critical scores are between 9 and 10.

We found in general that versions with higher deployment tend to have fewer vulnerabilities affecting them. This shows from the plots in Figure 4.4, where data points on the upper left of the graphs represent versions with relatively low server deployment and a relatively high number of vulnerabilities. The data points in the lower half of the graph represent versions with relatively low amounts of vulnerabilities. In all graphs we see relatively many versions on the upper left part, and no versions reside in the upper right part. In Figure 4.5 are the corresponding average impact scores, from which we see that versions tend to have similar average impact scores, regardless of their deployment statistics.

More specifically, Cassandra shows the lowest amounts of vulnerabilities in its versions, in 2021 the highest count is 4, and in 2023 one version has 5 vulnerabilities. In 2021 the most used Cassandra version has 0 vulnerabilities, whereas in

2023 this is 1 vulnerability. Note that in 2021 the scan did not identify non-TLS offering Cassandra servers, and in 2023 those servers are included. Most Cassandra versions have a relatively low deployment rate. There is 1 version with a critical average impact score, and the others score medium in 2021, but in 2023 servers are more vulnerable, with more versions scoring critical and high. The most deployed Cassandra version in 2023 scores medium.

For MariaDB and MySQL, the vulnerability counts are significantly higher than for the others. MariaDB has many versions inhibiting 40 to 80 vulnerabilities, and some versions reach as high as 115, in both 2021 and 2023. Again we see that wider deployed versions are affected by less vulnerabilities, at most having below 40 vulnerabilities. MySQL peaks with a few versions affected by nearly 500 vulnerabilities in both 2021 and 2023. We can clearly see that MySQL is the most vulnerable software in terms of sheer number of exploits. In terms of average impact scores, MariaDB shows scores between 6 and 8 mostly, but MySQL impact averages between 5 and 6, with a couple of critically affected versions in both 2021 and 2023, although those are the least wide deployed. MySQL vulnerabilities have an average of medium severity, while for the other database solutions the impact scores are averaging at high severity.

MongoDB and Redis have maximums of 17 vulnerabilities per version in 2021 and in 2023. Redis stands out with 10 vulnerabilities in their most widely deployed version in 2021. In the second scan in 2023 this version is no longer widely deployed, and instead its most widely deployed version now only has 2 vulnerabilities. Average impact scores are all at high, for both Redis and MongoDB in both scans. A few MongoDB versions with low deployment have critical scores, and 1 version in Redis 2023 has a critical score as well.

In Figure 4.6 we can compare the database solutions as the range on the x-axis is now in percentages. Cassandra's x-axis ranges from 0% to 60%, while all others range from 0% to 20%. We see that the distributions are very similar, with a lot of low deployment versions for all databases and in both scans. Few versions are widely deployed with over 5% of the total responding servers, and those have in common that they are affected by significantly less vulnerabilities. However, the average impact scores are not different from versions with lower deployment.

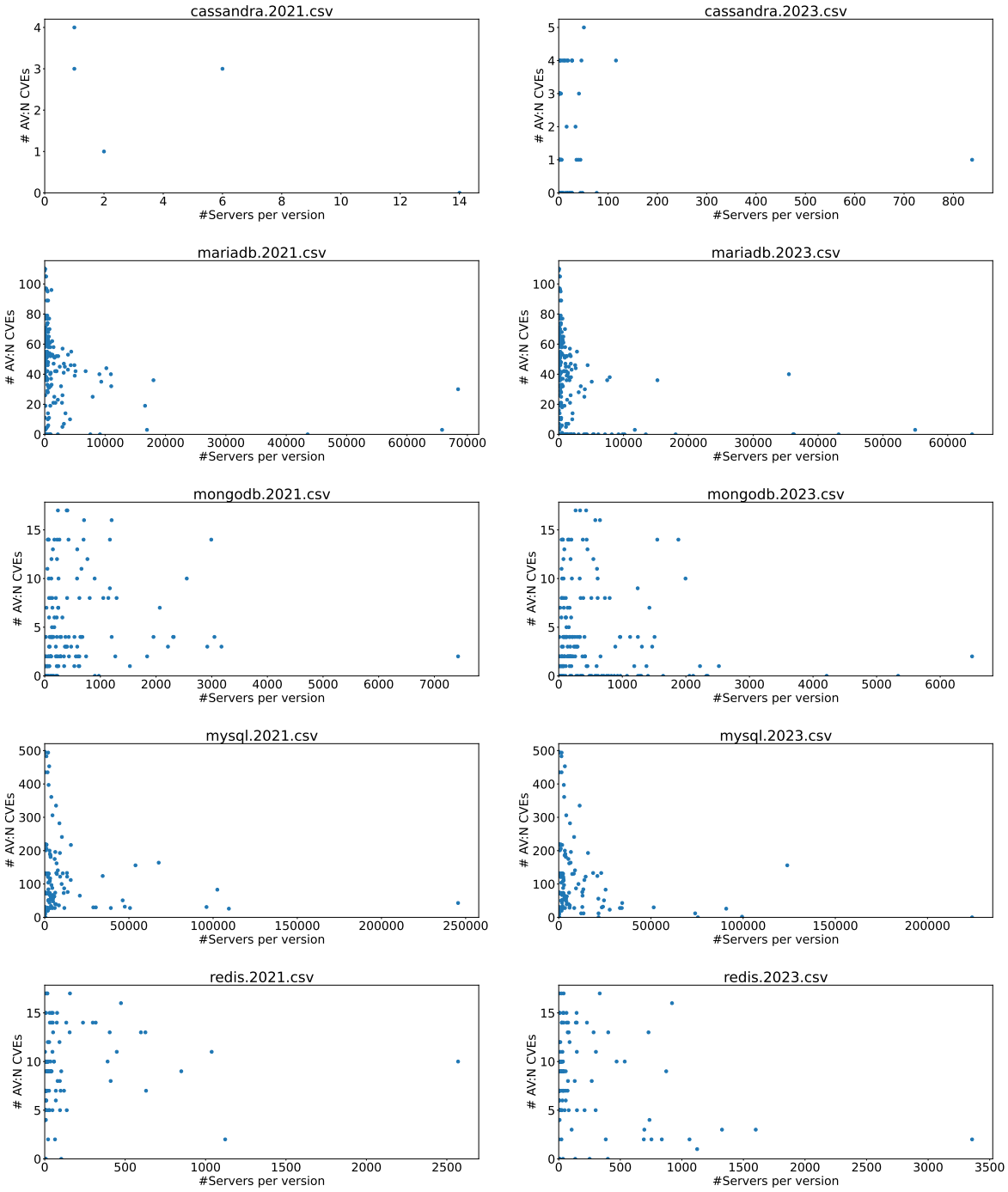


Figure 4.4: AV:N vulnerabilities present in database versions, and the amount of servers found running that version.

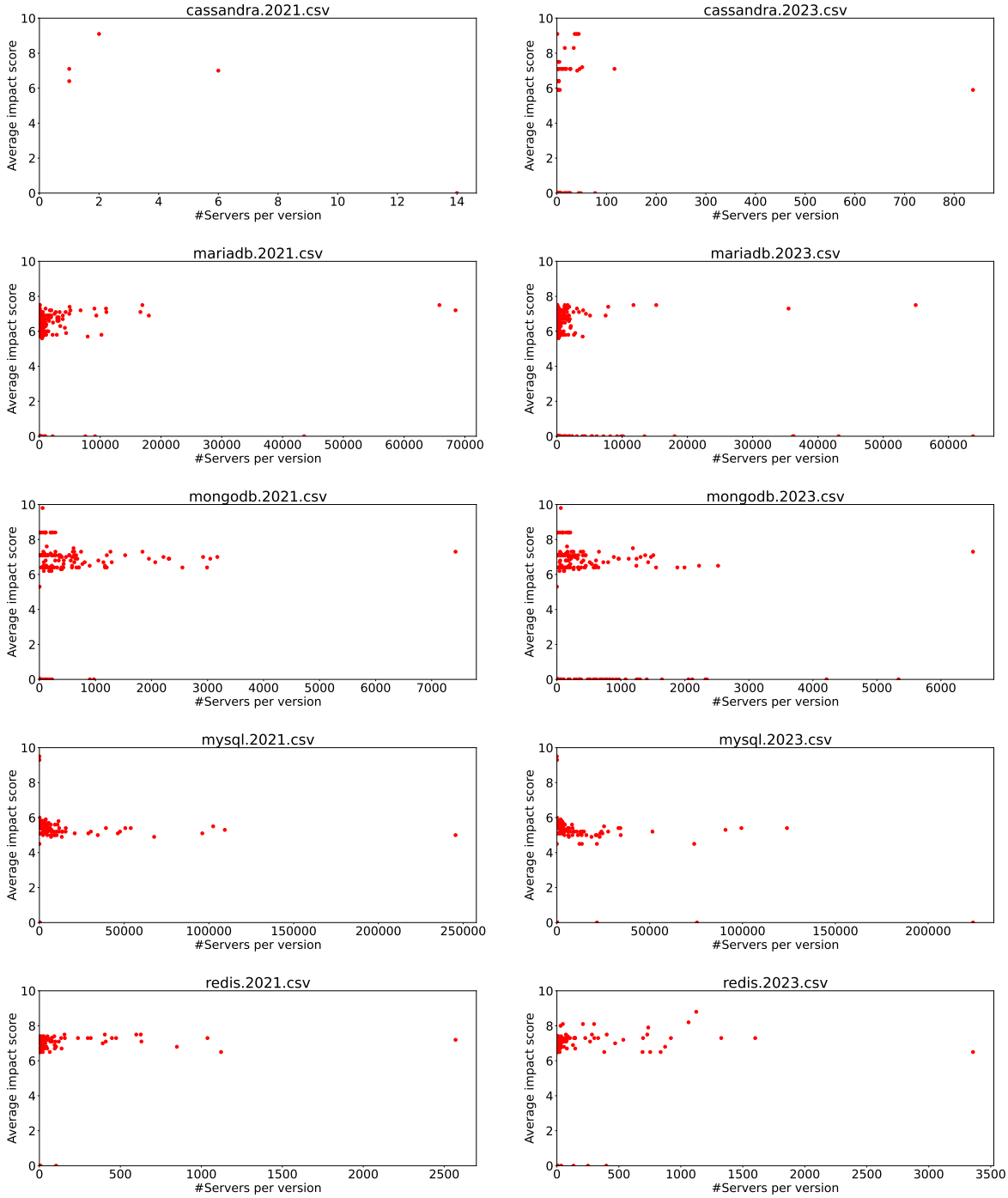


Figure 4.5: The average impact score of AV:N vulnerabilities present in database versions and the amount of servers found running that version.

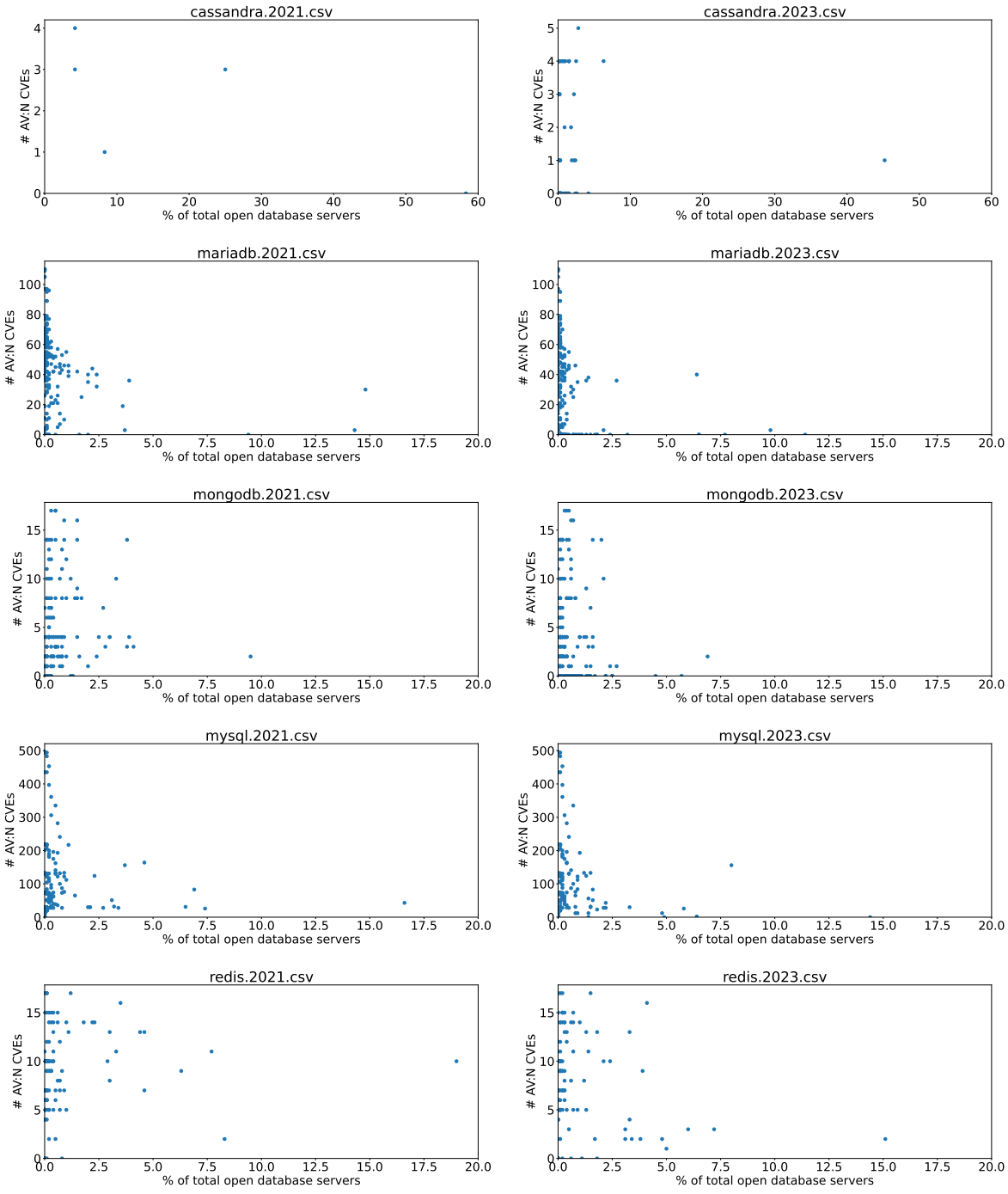


Figure 4.6: AV:N vulnerabilities present in database versions, and the percentage of servers found running that version.

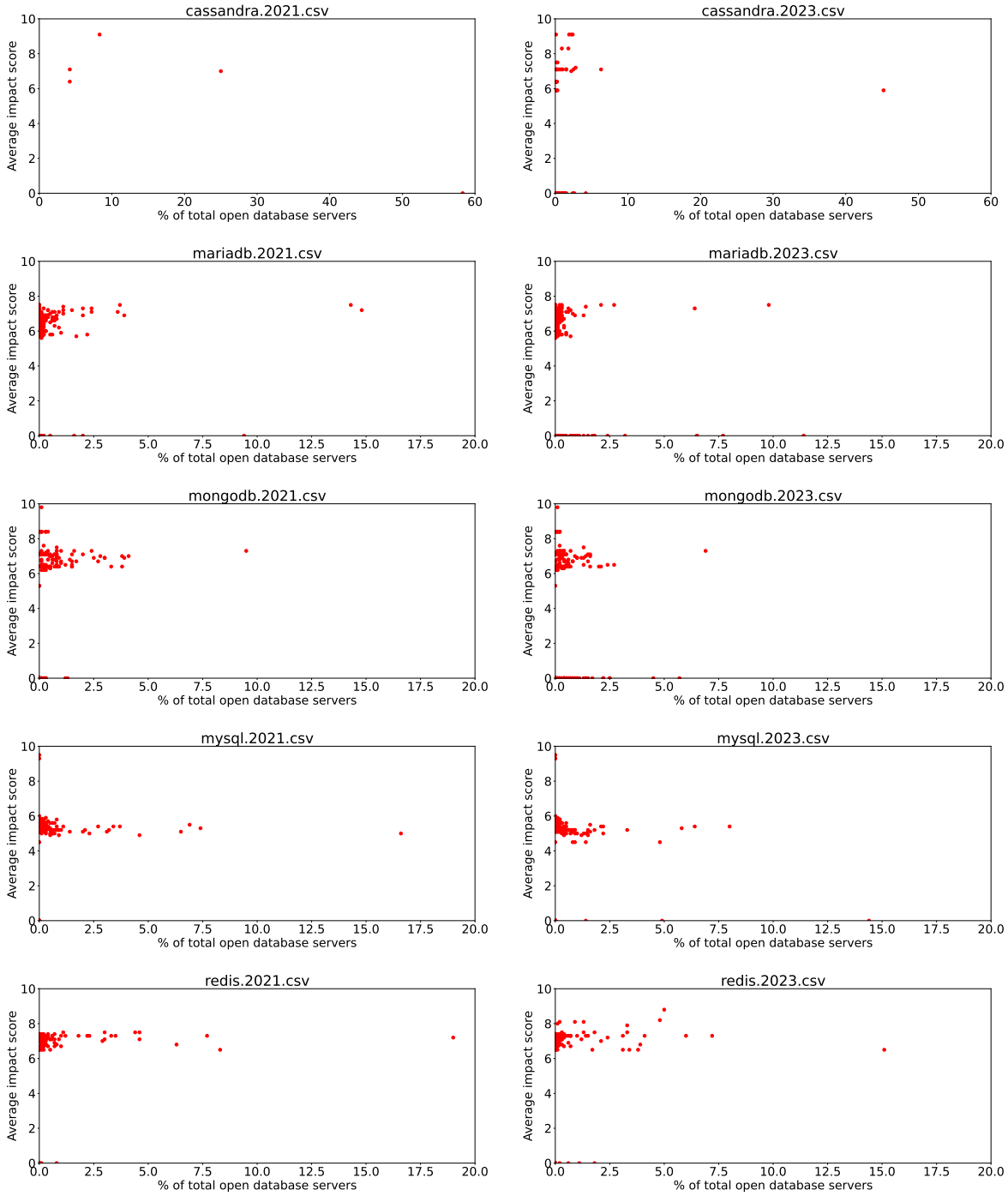


Figure 4.7: The average impact score of AV:N vulnerabilities present in database versions and the percentage of servers found running that version.

In appendix B we provide a range of scatter plots showing statistics for individual database solutions and individual classifications of vulnerabilities. We identify a number of vulnerability categories below to enable better comprehension of what those statistics mean.

- **AV:N** encompasses all vulnerabilities exploitable anywhere with an internet connection, meaning anyone with internet access is able to perform attack(s) enabled by the vulnerability. All vulnerabilities mentioned in this report share this property.
- **AV:N PR:None AV:Low** is the most dangerous category. Vulnerabilities with these properties can be exploited from anywhere, without any authentication or special circumstances, and are consistently reproducible.
- **AV:N PR:None AC:High** consists of vulnerabilities for which no authentication is needed, however, there need to be special conditions present outside of the attacker's control. These vulnerabilities are less dangerous then the previous category due to their inconsistency in exploitability.
- **AV:N PR:HL AC:Low** require some form of authentication, but no special circumstances. Usernames and passwords need to be guessed, stolen, or social engineered. This makes the likelihood of an attack lower then for PR:None vulnerabilities
- **AV:N PR:HL AV:High** require authentication, as well as complex circumstances outside of the attacker's control to be in such a way that the attack can succeed.

MySQL

In Figure B.1 are the scatter plots for vulnerability categories in MySQL 2021, and in Figure B.2 are the plots for MySQL 2023. We see that there are few of the most dangerous vulnerabilities, AV:N PR:None AV:Low, in MySQL versions. In both years the maximum amount of vulnerabilities per server is 12, and the average impact scores range from high to critical. We see a maximum of 14 vulnerabilities per server for the AV:N PR:None AC:High category and medium average scores in both years. AV:N PR:HL AC:Low vulnerabilities are very prevalent, occurring up to 420 times per MySQL version in 2021 and 2023. However, their average scores are all medium except for 1 version with very low deployment with a critical average score in 2021 and 2023. Their high complexity variant, AV:N PR:HL AV:High, occurs up to 55 times per version and has exclusively medium average impact scores except for 1 version with very low deployment with a high average score in 2023.

MongoDB

In Figure B.3 are the scatter plots for vulnerability categories in MongoDB 2021, and in Figure B.4 are the plots for MongoDB 2023. We see that there are few of the most dangerous vulnerabilities, AV:N PR:None AV:Low, in MongoDB versions. In both years the maximum amount of vulnerabilities per server is 4, and the average impact scores range from medium to critical. We see no servers with any AV:N PR:None AC:High vulnerabilities in both years. AV:N PR:HL AC:Low vulnerabilities occur up to 13 times per MongoDB version in 2021 and 2023. Their average scores are all medium. AV:N PR:HL AV:High vulnerabilities occur 0, 1, or 2 times per version and have medium to high average impact scores in 2021 and 2023.

Redis

In Figure B.5 are the scatter plots for vulnerability categories in Redis 2021, and in Figure B.6 are the plots for Redis 2023. We see that there are at most 4 of the most dangerous vulnerabilities, AV:N PR:None AV:Low, in Redis versions. In both years there is just one version with 4, the rest have 3 or less of these vulnerabilities. The average impact scores are medium to high in 2021, but range from medium to critical in 2023. We see no servers with any AV:N PR:None AC:High vulnerabilities in both years for Redis. AV:N PR:HL AC:Low vulnerabilities occur up to 9 times per Redis version in 2021 and 2023. Their average scores are all high in 2021 and in 2023, but in 2023 there are 2 versions with critical average impact scores. AV:N PR:HL AV:High vulnerabilities occur maximally 5 times per version and have high average impact scores in 2021 and 2023. In 2021 the version with the widest deployment has 4 AV:N PR:HL AV:High vulnerabilities, whereas in 2023 the widest deployed version is not affected by this category.

Cassandra

In Figure B.7 are the scatter plots for vulnerability categories in Cassandra 2021, and in Figure B.8 are the plots for Cassandra 2023. The statistics in 2021 are not comparable with 2023 due to the difference in scan implementation. One Cassandra version has 2 AV:N PR:None AV:Low vulnerability and the rest have 1 or 0 of those, all with a high impact score in 2023. The high complexity variant has many versions with 2 vulnerabilities, and the most widely deployed version with 1 of those. All have a medium average impact score. There is one authenticated vulnerability with low complexity in many Cassandra versions with critical impact score.

MariaDB

In Figure B.9 are the scatter plots for vulnerability categories in MariaDB 2021, and in Figure B.10 are the plots for MariaDB 2023. MariaDB has many AV:N PR:None AV:Low vulnerabilities with high average impact scores, up to 39 per version in 2021 and 2023. Only 6 versions do not have any of these dangerous vulnerabilities affecting them in 2021, which is much improved in 2023 with about half of the versions unaffected. Up to 7 AV:N PR:None AV:High vulnerabilities per version affect MariaDB versions, with impact scores ranging from medium to critical in both years. We see a substantial increase in unaffected versions in 2023 with respect to 2021.

4.2.2 Version changes over time

Machines responding on the same IP address and port in both 2021 and 2023 are compared in terms of software version. We cannot say with 100% certainty that it was the same machine and database server in both years, however the large amount of servers with the same IP address, open port and version string in both years does suggest this is a reasonable assumption to take. In Figure 4.8 the results are visible. Version not found means that the scan is not able to discover the software version of the database service found. Most servers are found to have their version unchanged, meaning they have the same version string in both scans. A very small portion of machines are found to have a version running in 2023 that is older than the version recorded in 2021.

49.8% of MySQL software on the same IP address and port has a newer version in 2023 with respect to 2021. MariaDB comes second with 38.7%, Ms-SQL third with 19.5%, MongoDB follows with 13%, and Redis showcases 0.7% of machines with newer versions in 2023.

Note that in the case of Oracle, the version compared is the TNS listener version (VSNNUM), which was the same in 2021 and in 2023.

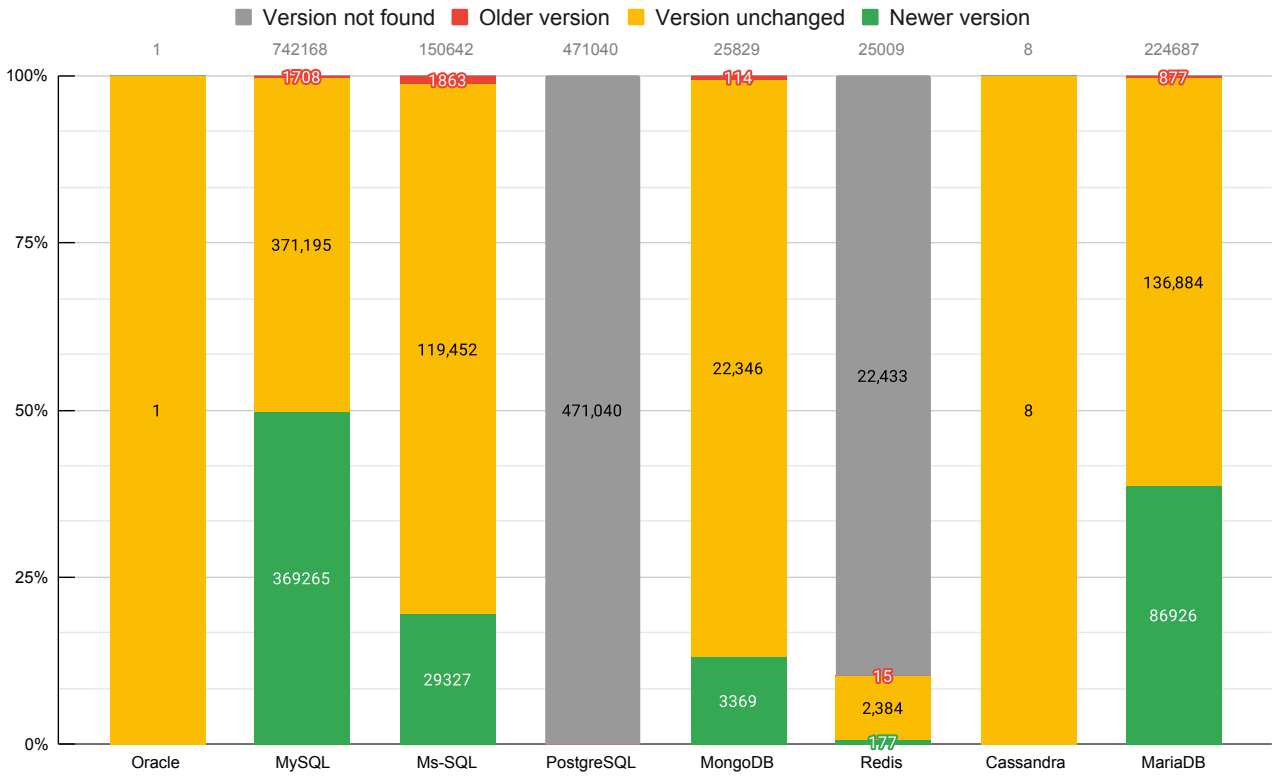


Figure 4.8: Version recorded of machines online in 2023 with respect to the version recorded in 2021.

Chapter 5

Discussion of results

Our presented results provide a broad insight into the security landscape of publicly accessible database servers, spanning multiple analyses including TLS configurations, vulnerabilities, certificate validations and software versions. The findings allow us to draw significant conclusions for the security posture of publicly accessible database servers running on their default IANA-assigned port of all 8 examined database solutions.

5.1 Individual database security postures

5.1.1 Oracle SQL

Our results highlight significant changes in Oracle's publicly accessible databases security posture between 2021 and 2023. Oracle shows a relatively low number of found servers compared to the other database solutions. A likely explanation for this is that databases in the Oracle Cloud Infrastructure (OCI) are by default not accessible from outside their virtual networks. To access a database in OCI it is required to first login to OCI before a connection with the database is setup within the virtual network. In addition, Oracle showcased a 96% reduction of publicly accessible databases between 2021 and 2023. These findings could be explained by the proprietary nature of Oracle and its focus on large-scale data and enhanced security features, which aligns with the characteristics of large enterprises. These companies typically prioritise security because of the sensitivity, scale and complexity of their operations.

In both 2021 and 2023 we see 100% TLS adoption in exposed Oracle databases, with a surprising discrepancy in configuration between the two scans. In 2021, 100% of servers used TLS 1.2 with a recommended cipher suite, 96.3% had a valid certificate, and 213 out of the 214 servers seen in 2021 are no longer publicly accessible in 2023. This suggests that Oracle database administrators in 2021 were very secu-

rity aware, and took steps to stop exposure of their databases to the internet. Furthermore, in 2023 only nine open Oracle databases were found, highlighting that the vast majority of Oracle databases already running before our first scan are no longer publicly accessible, and only a fraction of newly setup Oracle databases post-2021 are publicly accessible. A possible explanation could be the fact that in the Oracle Cloud Infrastructure (OCI), one must actively setup a rule in the configuration to allow Internet-wide incoming traffic on port 1521, and this rule already existed before our first scan in 2021. To the best of our knowledge there do not seem to be straightforward explanations as to why we see a drastic drop in publicly accessible Oracle databases. While a 96% reduction may sound drastic, in absolute numbers, the reduction from 214 to 9 is very small compared to the tens of thousands difference in other database solutions results.

In 2023, only 7 out of the 9 Oracle databases offer safe cipher suites, and none offer a verifiable certificate. These results show that security awareness

Oracle does not disclose software version information directly, however, its servers do advertise with their TNS (Transport Network Substrate) listener version number (VSNNUM), which can be used in certain cases to compute the Oracle server software version. Since the only solution is to restrict access to allowlisted IP addresses only, it is likely that the administrators of all Oracle servers found were not aware of this vulnerability at the time of the scans.

In conclusion, Oracle's approach to database security appears to be closely aligned with the requirements of large enterprises, which prioritise security due to their scale and sensitivity of operations. The large reduction in publicly accessible databases, combined with 100% TLS adoption, underscores Oracle's commitment to security best practices. However, the discrepancies in configuration between 2021 and 2023 highlight the potential challenges of maintaining consistent security practices, especially as new administrators enter the landscape.

5.1.2 MySQL

MySQL is also owned by Oracle Corporation, but differs from Oracle SQL in that it is mostly used for self hosting instead of in Oracle managed clouds. MySQL stands out with the highest number of publicly accessible databases among the eight database solutions studied. In 2021, there were 1.475 million servers, which increased to 1.55 million in 2023, indicating a 5% growth. This widespread adoption underscores MySQL's popularity as a database solution of choice for various applications. The increase in TLS adoption from 63% in 2021 to 65% in 2023 is a positive indicator of heightened security awareness among MySQL database administrators. The reduction in outdated TLS versions, from 3.5% in 2021 to 2% in 2023, showcases global

efforts to stop the usage of deprecated security protocols. What is concerning, however, is the slight increase in the use of MUST NOT cipher suites from 13.5% in 2021 to 15.7% in 2023. This trend could have many underlying reasons, such as some MySQL administrators being less concerned with security or less aware of the latest best security practices. The observed lack of verifiable TLS certificates from MySQL servers strengthens this notion, underscoring the need for improved certificate management knowledge among administrators.

In 2023, only six out of 1.55 million servers remain vulnerable to Heartbleed, signalling commendable efforts to manage known vulnerabilities within the MySQL community.

MySQL servers are especially vulnerable to authenticated attacks, with some encountered software versions affected by over 400 vulnerabilities in 2021 and 2023. On average these versions are impacted with a medium score, so most vulnerabilities are not particularly dangerous. More concerning is the maximum of 12 low complexity unauthenticated vulnerabilities per MySQL version in 2023, down from 13 in 2021. This category has a few average critical scores, but mostly high average impact scores. MySQL server administrators will need to become more aware of the vulnerabilities affecting the version their servers are running, and update to versions with no known vulnerabilities to increase security.

The statistics related to software version updates highlight the challenge of database maintenance. 47.7% of IP addresses running a MySQL server in 2023 were already seen in 2021 with a server on the MySQL port. While nearly half of those machines run newer software versions in 2023, half remained unchanged, and a small fraction is running an older version in 2023. This suggests that about half of MySQL administrators regularly update their software, which is significantly higher then for all other investigated database solutions. It is concerning on the other hand to witness more than half of the databases without signs of regular software version maintenance, assuming our assumption that the same IP addresses in both scans will belong to the same server holds.

In conclusion, MySQL's security trends demonstrate a mixed landscape of improvements and challenges. The increase in TLS adoption and reduced use of outdated TLS versions are positive signs, but the persistence of insecure cipher suites and a majority of self-signed or unverifiable certificates highlight areas for improvement. Continuous security education with attention to best practices, and ongoing software maintenance are essential for enhancing the security posture of MySQL databases, especially since the popularity of MySQL databases amplifies the potential impact of security incidents.

5.1.3 Ms-SQL

Microsoft SQL server runs publicly accessible on 357,665 machines in 2021, and that number is increase by 5% in 2023. Ms-SQL comes with TLS by default, which is likely the explanation for the observed 100% TLS adoption in both years. At first glance this looks well configured, however, upon examining which TLS versions Ms-SQL servers offer, we found that 49.7% of servers in 2021 and 45.2% in 2023 offer a deprecated TLS version. This goes hand in hand with 48.9% SHOULD NOT and 12.4% MUST NOT cipher suite usage in 2021, slightly improved to 45.3% SHOULD NOT and 8.2% MUST NOT in 2023. This is likely the result of the default settings which enable TLS 1.0 and TLS 1.1 in SQL Server 2019 and older versions with TLS support [41], which enable backwards compatibility between Ms-SQL versions. Administrators need to actively disable older TLS versions to bring their servers up to the latest security standards. There is possibly a long way to go for Ms-SQL to improve the security of data in transit between client and server, which is made difficult by the trade-off with backwards compatibility for old versions of Ms-SQL server running on older Windows versions like Windows XP and Windows 7 which are still in use.

Ms-SQL servers have very low rates of valid verifiable certificate usage for proving their identity. 5.6% in 2021, slightly improved to 6.5% in 2023. The vast majority of certificates are self-signed. Certificate usage is poor with most other database solutions as well. It is cheaper and easier to setup to use self-signed certificates, which is the most likely explanation for the amounts of self-signed certificates we encountered.

40.1% of IP addresses with Ms-SQL servers online in 2023, also had a Microsoft SQL server running in 2021. Of these, 1.2% shows an older version, 79.3% the same version, and 19.5% is running newer version in 2023 with respect to 2021. Maintaining servers to always run the newest software version does not consistently happen, and this is likely part of the reason why backwards compatibility continues to be needed.

In conclusion, Ms-SQL scores poorly on the security of their publicly accessible databases. This shows mostly in the choice made to tackle the trade off between security and backwards compatibility, with the default settings favouring backwards compatibility.

5.1.4 PostgreSQL

PostgreSQL is with 873,734 servers in 2021 and 767,887 in 2023 the second most widely publicly deployed database solution. PostgreSQL shows a 12% decline in publicly accessible databases in 2023, despite DB-Engines [30] showing an in-

crease in popularity for PostgreSQL between 2021 and 2023. This can be seen as a positive development showcasing efforts to reduce exposure of PostgreSQL databases.

In 2021, 37% offered TLS, but in 2023 this was reduced to 27%. TLS is not enabled by default in PostgreSQL, and it has a relatively easy setup process. Many people not fully aware of how to securely setup a database using PostgreSQL could be an explanation as to why so many publicly accessible PostgreSQL databases lack TLS support.

The PostgreSQL databases that do support TLS use TLS 1.2 overwhelmingly, with 99% in both years. Whereas in 2021 only 45.7% of servers offered a RECOMMENDED cipher suite, in 2023 this is drastically improved to 98.4%. An explanation for this could be that there is growing awareness of TLS best practices among PostgreSQL database administrators. PostgreSQL uses OpenSSL on Linux installations, they have changed their TLS default settings to be a little more secure in march 2023 with the release of OpenSSL version 3.1, the default security level increased from 1 to 2, however, level 2 still allows for SHOULD NOT cipher suites. The extra focus on security could have had a positive effect on security awareness amongst PostgreSQL database administrators.

TLS certificate practices within the PostgreSQL community are similarly meaningless as most other databases, with just 1.3% valid certificates in 2021, and a minor improvement to 5.8% in 2023.

The authentication barrier before the server version can be found, helps PostgreSQL servers hide their potential vulnerabilities during scans. This is a good example of security by design and has prevented us from identifying potential vulnerabilities on the publicly accessible PostgreSQL servers.

To conclude, PostgreSQL faces its biggest challenge with the declining number of TLS enabled servers. More awareness could be raised about the risks of plaintext communication and how TLS mitigates the potentially harmful effects of it.

5.1.5 MongoDB

MongoDB has a moderate publicly accessible presence of 77,796 servers in 2021, significantly increased with 21% to 94,435 in 2023. According to DB-engines, the popularity of MongoDB has decreased by a small amount, but this is not reflected in our results.

The MongoDB scan did not check whether a server offered TLS connections, so unfortunately we do not have TLS configuration statistics to analyse.

It is quite feasible to figure out which vulnerabilities affect MongoDB servers, as software version information is available to anyone who connects to publicly acces-

sible MongoDB databases. More than half of MongoDB servers are affected by at least one vulnerability. A majority of running versions is affected by at least one very dangerous no authentication low complexity vulnerability, including the most deployed version in both years. With impact scores up to 10, the critical level, and on average a score of 7, the medium impact level, MongoDB servers are exposed to quite some risks. A possible reason for this is that MongoDB administrators do not actively monitor the CVE nor update their software versions regularly.

MongoDB servers on IP addresses with the MongoDB port open in both 2021 and 2023 have the lowest newer version statistic of all investigated databases. 13% are found to run a newer version in 2023 compared to 2021, while 86.5% advertised the same version in both years. This is more reason to believe that MongoDB servers are not regularly updated.

5.1.6 Redis

Redis has 99,050 publicly accessible servers in 2021, significantly increased with 78% to 176,553 in 2023. According to DB-engines, the popularity of Redis has decreased by a small amount, but this is not reflected in our results.

The Redis scan did not check whether a server offered TLS connections, so unfortunately we do not have TLS configuration statistics to analyse.

It is quite feasible to figure out which vulnerabilities affect Redis servers, as software version information is available to anyone who connects to publicly accessible Redis databases. More than half of Redis servers are affected by at least one vulnerability. A majority of running versions is affected by at least one very dangerous no authentication low complexity vulnerability, including the most deployed version in 2021, but excluding the four most widely deployed versions in 2023. With impact scores up to 10, the critical level, in 2023, and on average a score of 7, the medium impact level, Redis servers are exposed to quite some risks. Redis sees a fair reduction of the most dangerous category of vulnerabilities affecting widely deployed versions from 2021 to 2023, which is a result of newer Redis versions not being affected by these vulnerabilities, and Redis database administrators installing the newest versions.

With this in mind, it is a good feature that most Redis servers do not publicise their version string to unauthenticated connections, 82.6% in 2021 and 84.7% in 2023. Our version change over time check found that 89.7% of Redis ports open on the same IP address in 2021 and 2023 do not advertise server version, and 9.5% remained on the same version.

5.1.7 MariaDB

MariaDB stands out with the lowest TLS adoption in servers around the world. Of the 460,786 servers found in 2021, just 5% offered TLS. In 2023, the amount of servers increased by 21%, and the TLS adoption increased, to 15%. MariaDB uses unencrypted connections by default, and TLS needs to be manually enabled during server installation. MariaDB administrators might be unaware of the risks that come with unencrypted connections or do not take the effort needed to configure their database with TLS.

The MariaDB administrators that do configure TLS for their server, do rarely allow for TLS versions older than 1.2 to be used, 4.9% in 2021, shrunk to 1.2% in 2023. The configured servers in 2021 saw 27.8% verifiable certificates, which improved greatly to 95.4% in 2023. MariaDB is the only database solution with a minority of just 8.6% self-signed certificates in 2021, which decreased to 2.5% in 2023. These good MariaDB security statistics are however accompanied by very poor cipher suite usage. In 2021, 20% MUST NOT cipher suite usage is registered, worryingly, this has increased to a stunning 59.7% in 2023. In the certificate domain, MariaDB administrators are well aware in 2023 how to setup their certificates securely, but at the same time they do allow their servers to choose old deprecated encryption ciphers to encrypt connections. This could be explained by the MariaDB procedure to configure TLS, a certificate is required, and active setup is hence needed to make the server install. However, the server will install just fine without any cipher suite restrictions. It is not a required step to look at those. It seems like many MariaDB administrators have not taken the cipher suite restriction option into account when configuring their server. It is highly recommended for MariaDB to raise more awareness about recommended cipher suites.

In 2021, MariaDB versions are affected by up to 38 unauthenticated low complexity vulnerabilities. Just six versions are not affected by this category of vulnerabilities. Impact scores average at high. In 2023, this has improved somewhat with almost half of versions not affected by any unauthenticated low complexity vulnerability, and impact scores remaining at the high average level. Yet there are some widely deployed versions in both years with quite a lot of dangerous vulnerabilities affecting them. We see that MariaDB shows 38.7% of IP addresses with the MariaDB port open operating a newer version in 2023 compared to in 2021. This shows that MariaDB administrators do update their software, but not not enough to really phase out older more vulnerable versions staying deployed. More awareness on the importance of software maintenance and updates could help to reduce the weak security posture of MariaDB servers.

5.1.8 Cassandra

With 24 TLS offering servers found in 2021 and 9 in 2023, it is difficult to say whether the Cassandra TLS statistics are very meaningful. The TLS configurations all use TLS 1.2 and only offered RECOMMENDED cipher suites in 2023, however none offered a verifiable certificate. These results are better than Oracle's 9 TLS results in 2023.

8 IP addresses in 2023 were also found to have the Cassandra port open in 2021, and all remained on exactly the same version. So there is no indication of server maintenance done on Cassandra servers running for a long time. Again, just 8 results makes this not too meaningful.

The most deployed version in 2021 has zero vulnerabilities. All other versions have a few, with just one server vulnerable to one unauthenticated low complexity vulnerability with a high impact score. In 2023, the maximum amount of vulnerabilities in a server version is 5. Cassandra administrators would do good to update their servers to versions without known vulnerabilities to increase security.

5.2 Overall database security

In total, 3.5 million databases among our 8 investigated solutions are publicly accessible in 2023 with 51% of those servers offering TLS. We've looked at TLS configurations and version vulnerability, and presented insights based on our results.

TLS version support for deprecated versions is minimal except for Ms-SQL. The configuration of TLS is however still a concern, with self-signed certificates being very common, and unsafe cipher suite usage occurring too often. Herein lies a task for database software publishers to change their default configurations to safe and up to date security best practice standards, and ensure that sufficient information on security best practices is available with the database products. In that way, administrators will need to make a conscious choice to enable deprecated configuration options, for example to make a server backwards compatible if that is a requirement.

Software version vulnerability appears to be a major concern for publicly accessible databases, as established with our CVE database and server version cross-referencing. The scatterplots in Section 4.2.1 and in Appendix B tell a story of vulnerable servers seemingly easy to target. We do note that there are limitations with our approach, for example it is possible that servers have been manually patched without their version string being changed or updated to reflect the security updates. To verify our claims it would be necessary to conduct an experiment to check whether identified vulnerabilities in publicly accessible database servers are exploitable in practice, however such evaluation would present major ethical and legal challenges.

We have identified a massive amount of publicly accessible database servers running versions with vulnerabilities affecting them. Software maintenance is proving to be very difficult, many database administrators do not seem to update their software to the latest version regularly or move to newer releases when they come out. Knowledge about the number of vulnerabilities affecting database software might not be very common unfortunately. We highly recommend database software to not reveal their version information to anyone who happens to connect. This could be achieved in multiple ways, for example, Ms-SQL version information does not disclose whether vulnerability fixes have been applied to the version running.

This report has highlighted the challenges and risks that database software publishers deal with. It sheds light on the many publicly accessible databases online worldwide, and the ease with which they can be scanned and targeted.

Chapter 6

Conclusions

6.1 Public database connection security posture

Connections with publicly accessible databases are not secured by default in a majority of publicly accessible databases. About half of publicly accessible databases offer the possibility to establish a secure connection with the TLS protocol. TLS is in a minority of cases configured in such a way that an agreement on deprecated security protocols can be established, but most servers direct clients to TLS version 1.2. Most concerning is the tendency of unverifiable certificate usage amongst the majority of publicly accessible databases. In conclusion, database administrators could do more to help secure their clients connections and ensure that data remains confidential in transit, by configuring their database servers to offer the currently recommended TLS versions, with RECOMMENDED cipher suites, and investing in a CA signed certificate for encrypted connections.

6.2 Public database software security posture

We have learned that MySQL, MongoDB, Redis, Cassandra and MariaDB allow their version string to be acquired by anyone who attempts a connection to a publicly accessible database. From this information it is possible to deduce how vulnerable the software running on the server is by cross referencing version information with the CVE database [40]. From our analysis we can conclude that a majority of servers run DBMS versions affected by vulnerabilities, and that many servers do not get updated to newer versions regularly. We have to conclude that the security posture of publicly accessible databases is poor, and it is important to raise awareness about this issue to database vendors and administrators. We recommend database administrators to hide version information as an added measure of security. Note that security by obscurity does not make a server safe, attacks can be launched in an

automatic manner to see whether they succeed, hidden version or not.

6.3 Recommendations

6.3.1 Improvements to our study

To improve this research were it repeated, we discuss two key recommendations here. The first is to make the scan process as equal as possible for all the different databases. This includes enabling TLS scanning for all databases for example, which has the advantage of a more complete overview and better comparability. The second recommendation is that more analysis could have been done on the gathered data, for example IP address geographical location could be taken into account to study differences between countries. This might shed light on the effectiveness of attention to security within tech education in different countries, and help national policymakers and researchers gain insights for evaluation of current policies and data to base new policies and research on.

6.3.2 Future work

For future work building and improving on our method and results, we discuss two key recommendations here. The first recommendation is to find a way to verify whether servers with a particular version are provably vulnerable to identified vulnerabilities affecting that version. This could for example be done by analysing the vulnerabilities to find out if some of them can be checked for in an ethical way to confirm their presence without any impact. This would need a sharp ethical review, but would be necessary to prove our claims about server vulnerability. The second recommendation would be to increase the amount of database solutions to scan, which would give a more complete overview of the publicly accessible database landscape.

Bibliography

[1] (2020) Zgrab 2.0. [Online]. Available: https://github.com/zmap/zgrab2

[2] K. Moriarty and S. Farrell, “Deprecating TLS 1.0 and TLS 1.1,” RFC 8996, Mar. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8996

[3] R. Barnes, M. Thomson, A. Pironti, and A. Langley, “Deprecating Secure Sockets Layer Version 3.0,” RFC 7568, Jun. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7568

[4] (2023) Cve. [Online]. Available: https://cve.org/

[5] D. Leonard and D. Loguinov, “Demystifying service discovery,” in *Proceedings of the 10th annual conference on Internet measurement - IMC '10*. ACM Press, 2010. [Online]. Available: https://doi.org/10.1145/1879141.1879156

[6] Z. Durumeric, E. Wustrow, and J. A. Halderman, “ZMap: Fast Internet-wide scanning and its security applications,” in *Proceedings of the 22nd USENIX Security Symposium*, Washington, D.C., United States, Aug. 2013.

[7] M. Smith and D. Loguinov, “Enabling high-performance internet-wide measurements on windows,” in *Proceedings of the 11th International Conference on Passive and Active Measurement*, ser. PAM’10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 121–130.

[8] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson, “Target generation for internet-wide ipv6 scanning,” in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 242–253. [Online]. Available: https://doi.org/10.1145/3131365.3131405

[9] P. Foremski, D. Plonka, and A. Berger, “Entropy/ip: Uncovering structure in ipv6 addresses,” in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 167–181. [Online]. Available: https://doi.org/10.1145/2987443.2987445

[10] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczyński, S. D. Strowes, L. Hendriks, and G. Carle, “Clusters in the expanse: Understanding and unbiasing ipv6 hitlists,” in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 364–378. [Online]. Available: https://doi.org/10.1145/3278532.3278564

[11] G. Wan, L. Izhikevich, D. Adrian, K. Yoshioka, R. Holz, C. Rossow, and Z. Durumeric, “On the origin of scanning,” in *Proceedings of the ACM Internet Measurement Conference*. ACM, Oct. 2020. [Online]. Available: https://doi.org/10.1145/3419394.3424214

[12] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, “A search engine backed by internet-wide scanning,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 542–553. [Online]. Available: https://doi.org/10.1145/2810103.2813703

[13] L. Izhikevich, R. Teixeira, and Z. Durumeric, “LZR: Identifying unexpected internet services,” in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/izhikevich

[14] D. Ferrari, M. Carminati, M. Polino, and S. Zanero, “NoSQL breakdown: A large-scale analysis of misconfigured NoSQL services,” in *Annual Computer Security Applications Conference*. ACM, Dec. 2020. [Online]. Available: https://doi.org/10.1145/3427228.3427260

[15] (2020) Nmap network scanning: A quick port scanning tutorial. [Online]. Available: https://nmap.org/book/port-scanning-tutorial.html

[16] A. Continella, M. Polino, M. Pogliani, and S. Zanero, “There’s a hole in that bucket! a large-scale analysis of misconfigured s3 buckets,” in *Proceedings of the 34th Annual Computer Security Applications Conference*, ser. ACSAC ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 702–711. [Online]. Available: https://doi.org/10.1145/3274694.3274736

[17] J. Cable, L. Izhikevich, D. Gregory, and Z. Durumeric, “Stratosphere: Finding vulnerable cloud storage buckets,” 2021. [Online]. Available: https://lizizhikevich.github.io/assets/papers/stratosphere.pdf

[18] Public buckets by grayhatwarfare. [Online]. Available: https://buckets.grayhatwarfare.com/

[19] (2020) Masscan: Mass ip port scanner. [Online]. Available: https://github.com/robertdavidgraham/masscan

[20] D. Springall, Z. Durumeric, and J. A. Halderman, "Measuring the security harm of TLS crypto shortcuts," in *Proceedings of the 2016 Internet Measurement Conference*. ACM, Nov. 2016. [Online]. Available: https://doi.org/10.1145/2987443.2987480

[21] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz, "Mission accomplished? https security after diginotar," ser. IMC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 325–340. [Online]. Available: https://doi.org/10.1145/3131365.3131401

[22] Goscanner. [Online]. Available: https://github.com/tumi8/goscanner

[23] J. Rüth, C. Bormann, and O. Hohlfeld, "Large-scale scanning of tcp's initial window," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 304–310. [Online]. Available: https://doi.org/10.1145/3131365.3131370

[24] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the https certificate ecosystem," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 291–304. [Online]. Available: https://doi.org/10.1145/2504730.2504755

[25] 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 *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, Nov. 2014. [Online]. Available: https://doi.org/10.1145/2663716.2663755

[26] M. Williams, M. Tüxen, and R. Seggelmann, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension," RFC 6520, Feb. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6520.txt

[27] R. Holz, J. Hiller, J. Amann, A. Razaghpanah, T. Jost, N. Vallina-Rodriguez, and O. Hohlfeld, "Tracking the deployment of tls 1.3 on the web: A story of experimentation and centralization," *SIGCOMM Comput. Commun. Rev.*, vol. 50, no. 3, p. 3–15, Jul. 2020. [Online]. Available: https://doi.org/10.1145/3411740.3411742

[28] E. Kenneally and D. Dittrich, “The menlo report: Ethical principles guiding information and communication technology research,” *SSRN Electronic Journal*, 2012. [Online]. Available: https://doi.org/10.2139/ssrn.2445102

[29] C. Partridge and M. Allman, “Addressing ethical considerations in network measurement papers: Abstract,” in *Proceedings of the 2015 ACM SIGCOMM Workshop on Ethics in Networked Systems Research*, ser. NS Ethics ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 33. [Online]. Available: https://doi.org/10.1145/2793013.2793014

[30] “DB-Engines Ranking — db-engines.com,” https://db-engines.com/en/ranking, 2023, [Accessed 05-08-2023].

[31] “pyasn,” https://catalog.caida.org/software/pyasn, accessed: 12-09-2023.

[32] (2023) University of oregon routeviews project. [Online]. Available: https://www.routeviews.org/routeviews/

[33] (2020) tcpdump. [Online]. Available: https://www.tcpdump.org/

[34] “Gocql,” 2016. [Online]. Available: https://github.com/gocql/gocql

[35] (2020) Zcrypto. [Online]. Available: https://github.com/zmap/zcrypto

[36] Y. Sheffer, P. Saint-Andre, and T. Fossati, “Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS),” RFC 9325, Nov. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9325

[37] “Openssl,” 1999. [Online]. Available: https://www.openssl.org/

[38] “Common ca database.” [Online]. Available: https://www.ccadb.org/

[39] Z. Ma, J. Mason, M. Antonakakis, Z. Durumeric, and M. Bailey, “What’s in a name? exploring CA certificate control,” in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 4383–4400. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/ma

[40] A. Dulaunoy, P.-J. Moreels, P. Tikken, and R. Vinot, “Search main page: Cve-search - tool-set to perform local searches for known vulnerabilities.” [Online]. Available: https://www.cve-search.org/

[41] (2023) Tls 1.2 support for microsoft sql server. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/sql/database-engine/connect/tls-1-2-support-microsoft-sql-server

[42] S. O. Bradner, “Key words for use in RFCs to Indicate Requirement Levels,” RFC 2119, Mar. 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2119

[43] (2023) Heartbleed, bluekeep and other vulnerabilities that didn’t disappear just because we don’t talk about them anymore. [Online]. Available: https://isc.sans.edu/diary/26798

[44] openssl security level. [Online]. Available: https://www.openssl.org/docs/man1.1.1/man3/SSL_CTX_set_security_level.html

[45] (2023) Oracle tns listener vsnnum version remote information disclosure. [Online]. Available: https://www.tenable.com/plugins/nessus/110053

[46] (2023) Common vulnerability scoring system sig. [Online]. Available: https://www.first.org/cvss/

Appendix A

TLS complete data tables

TLS version	MariaDB 2021	MariaDB 2023	MySQL 2021	MySQL 2023
TLSv1.2	24,130	80,703	890,819	992,464
TLSv1.1	1,241	938	17,502	18,447
TLSv1.0	11	13	15,182	1,440
SSLv3	0	0	0	0

TLS version	Cassandra 2021	Cassandra 2023	Oracle 2021	Oracle 2023
TLSv1.2	24	9	214	7
TLSv1.1	0	0	0	0
TLSv1.0	0	0	0	2
SSLv3	0	0	0	0

TLS version	Ms-SQL 2021	Ms-SQL 2023	PostgreSQL 2021	PostgreSQL 2023
TLSv1.2	179,788	205,693	324,071	204,565
TLSv1.1	9	12	0	0
TLSv1.0	177,652	169,741	2,367	1,986
SSLv3	216	155	0	0

Table A.1: Full TLS version data

Cipher suite	MariaDB 2021	MariaDB 2023	MySQL 2021	MySQL 2023
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	12	141	13	20
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	0	0	0	0
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	0	1	570	29
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	20,043	32,605	798,567	853,785
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	244	176	0	0
TLS_ECDHE_RSA_WITH_RC4_128_SHA	0	0	0	1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	1	9	0	0
TLS_RSA_WITH_AES_128_CBC_SHA	894	2,274	102,138	138,748
TLS_RSA_WITH_AES_256_CBC_SHA	1,249	939	20,837	19,375
TLS_RSA_WITH_RC4_128_SHA	2,939	45,509	1,378	393

Cipher suite	Cassandra 2021	Cassandra 2023	Oracle 2021	Oracle 2023
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	0	0	0	0
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	0	0	0	0
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	1	0	0	0
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	19	9	7	214
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	0	0	0	0
TLS_ECDHE_RSA_WITH_RC4_128_SHA	0	0	0	0
TLS_RSA_WITH_3DES_EDE_CBC_SHA	0	0	0	0
TLS_RSA_WITH_AES_128_CBC_SHA	0	0	0	0
TLS_RSA_WITH_AES_256_CBC_SHA	0	0	0	0
TLS_RSA_WITH_RC4_128_SHA	4	0	2	0

Cipher suite	Ms-SQL 2021	Ms-SQL 2023	PostgreSQL 2021	PostgreSQL 2023
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	5	8	209	765
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	0	0	4	13
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	74	85	0	23
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	138,387	174,472	148,875	202,457
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	174,954	170,211	164,839	1,017
TLS_ECDHE_RSA_WITH_RC4_128_SHA	0	0	0	0
TLS_RSA_WITH_3DES_EDE_CBC_SHA	242	202	2	0
TLS_RSA_WITH_AES_128_CBC_SHA	38,251	26,748	730	4
TLS_RSA_WITH_AES_256_CBC_SHA	550	194	43	33
TLS_RSA_WITH_RC4_128_SHA	5,202	3,681	11,736	2,239

Table A.2: Full TLS ciphersuite data

Validation	MariaDB 2021	MariaDB 2023	MySQL 2021	MySQL 2023
Certificate is not yet valid	0	0	2	0
Certificate expired	67	340	523	627
Unable to get local issuer certificate	16,027	1,768	344,286	192,074
Self signed certificate	2,163	1,992	402,653	643,891
Level 2 security (server certificate key too weak)	6,180	10,044	13,722	16,626
Level 2 security (CA certificate key too weak)	827	67,315	156	128
Level 3 security (server certificate key too weak)	0	13	0	0
Level 4 security (server certificate key too weak)	0	1	1	0

Validation	cassandra1	cassandra2	oracle1	oracle2
Certificate is not yet valid	0	0	0	0
Certificate expired	0	0	0	0
Unable to get local issuer certificate	0	1	2	3
Self signed certificate	24	8	6	6
Level 2 security (server certificate key too weak)	0	0	206	0
Level 2 security (CA certificate key too weak)	0	0	0	0
Level 3 security (server certificate key too weak)	0	0	0	0
Level 4 security (server certificate key too weak)	0	0	0	0

Validation	Ms-SQL 2021	Ms-SQL 2023	PostgreSQL 2021	PostgreSQL 2023
Certificate is not yet valid	0	0	2	0
Certificate expired	110	185	500	399
Unable to get local issuer certificate	36,112	35,833	30,826	31,403
Self signed certificate	301,083	315,025	288,929	161,095
Level 2 security (server certificate key too weak)	20,120	24,271	4,104	11,619
Level 2 security (CA certificate key too weak)	58	91	93	192
Level 3 security (server certificate key too weak)	0	2	2	6
Level 4 security (server certificate key too weak)	0	0	0	1

Table A.3: Full certificate validation data.

Appendix B

Vulnerability statistics

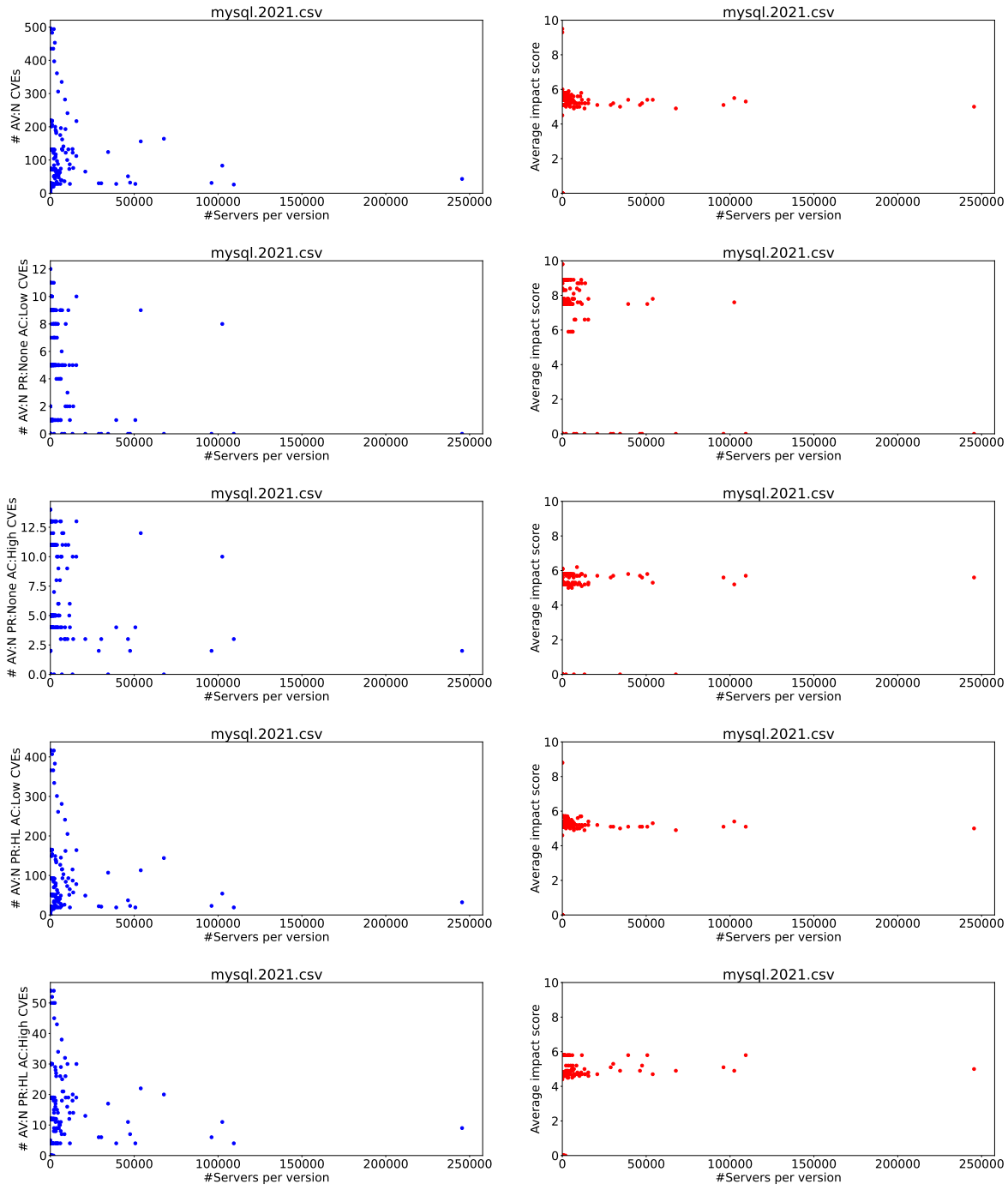


Figure B.1: Categorised vulnerabilities in MySQL versions and their impact scores in 2021.

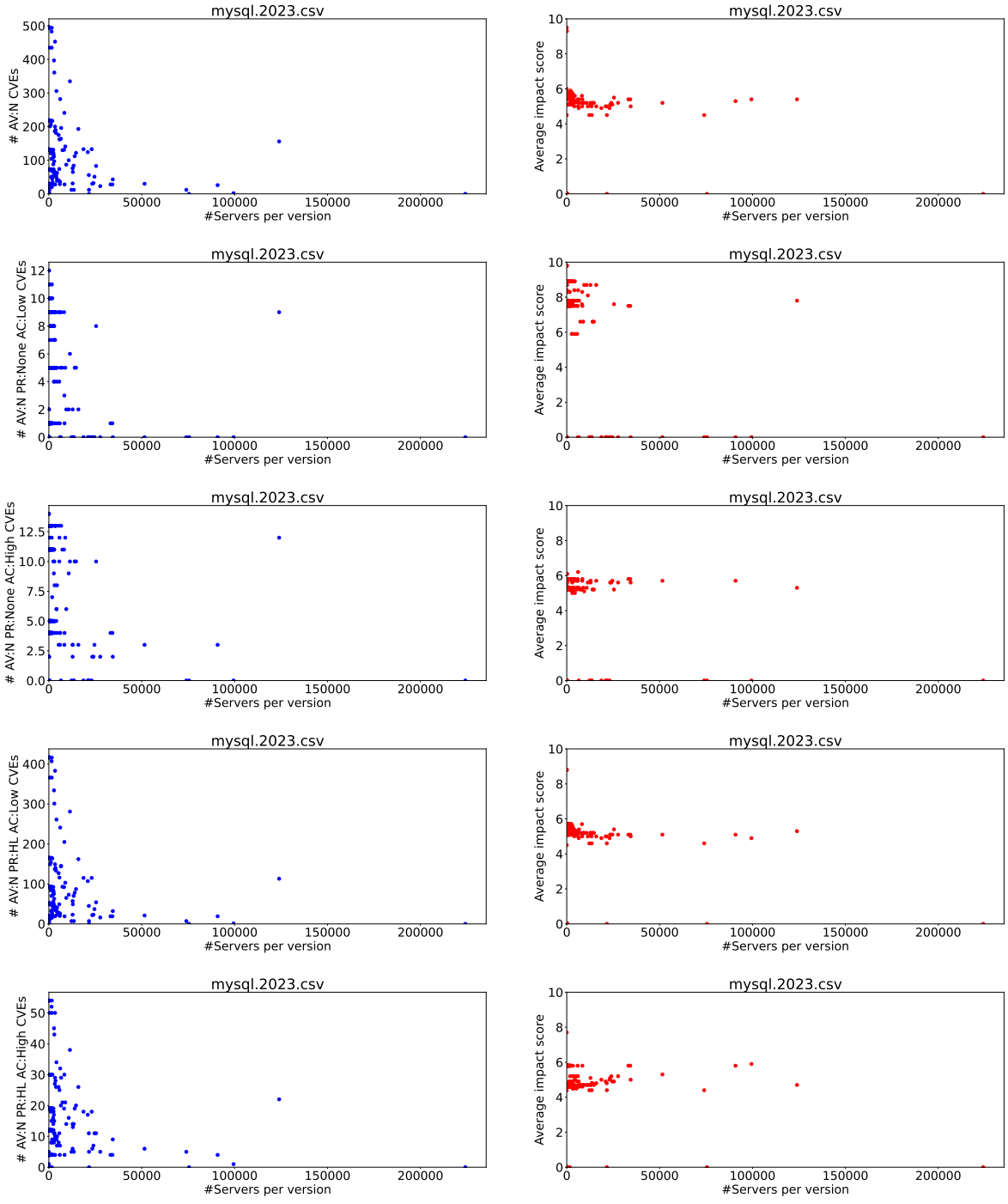


Figure B.2: Categorised vulnerabilities in MySQL versions and their impact scores in 2023.

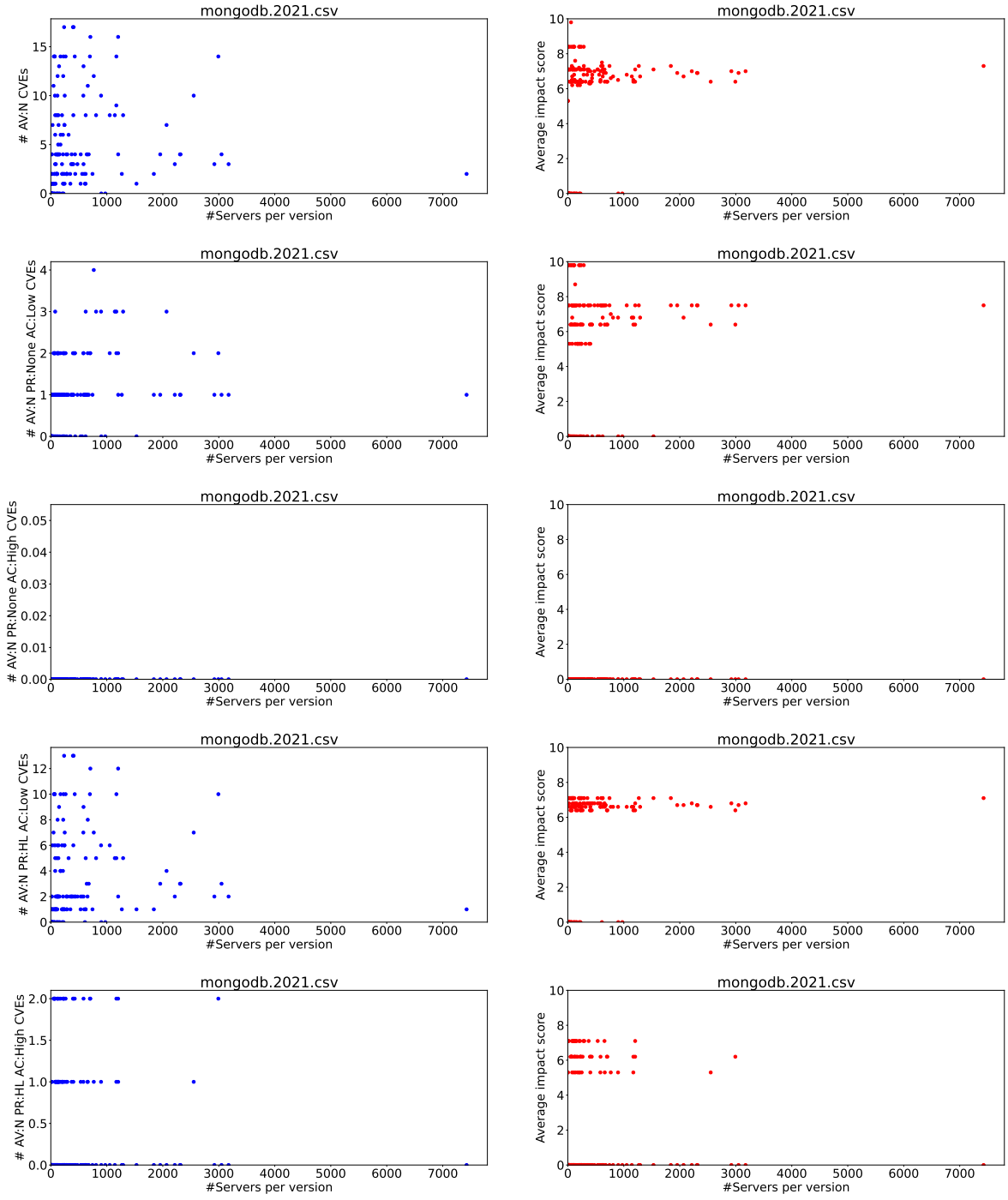


Figure B.3: Categorised vulnerabilities in MongoDB versions and their impact scores in 2021.

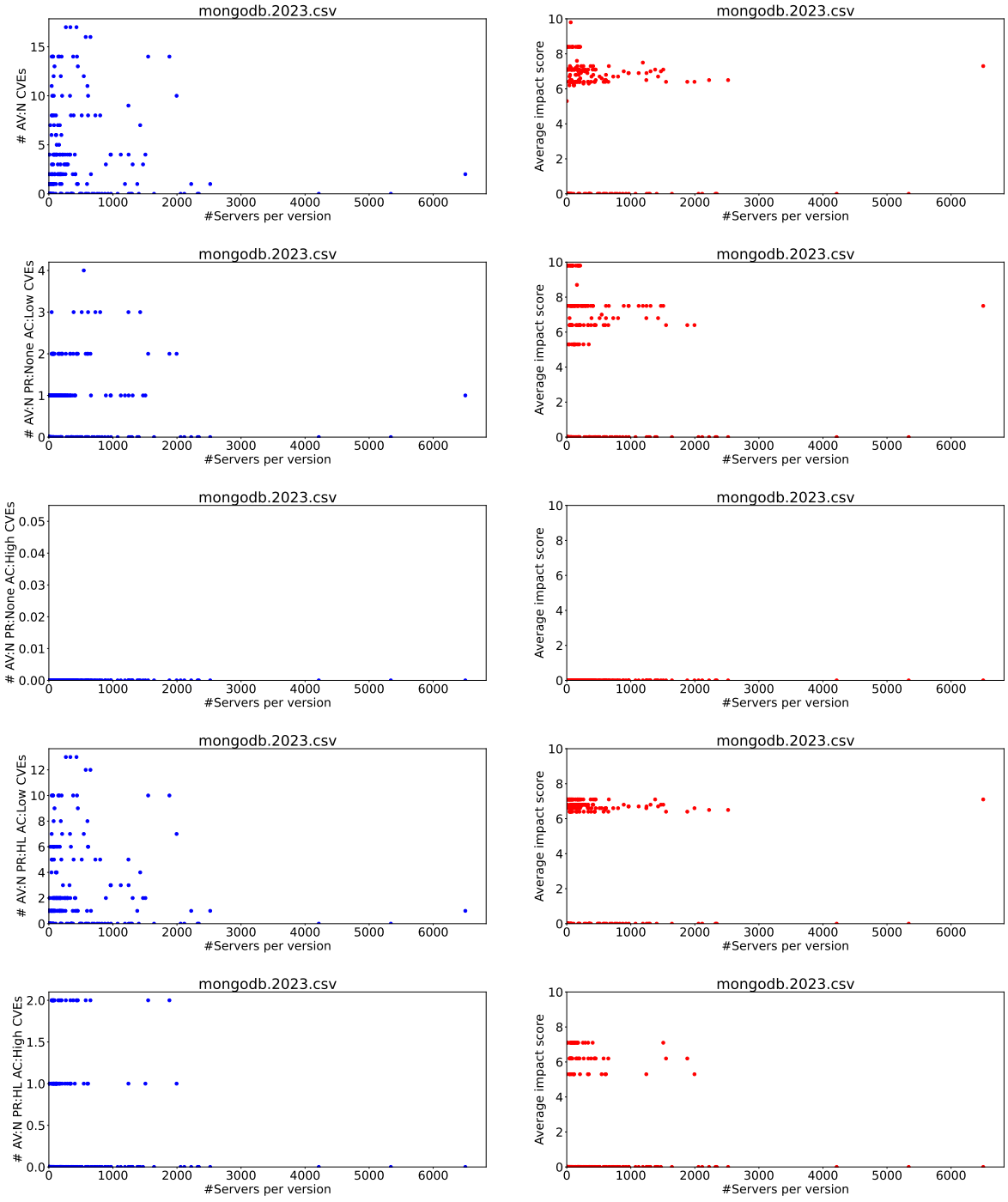


Figure B.4: Categorised vulnerabilities in MongoDB versions and their impact scores in 2023.

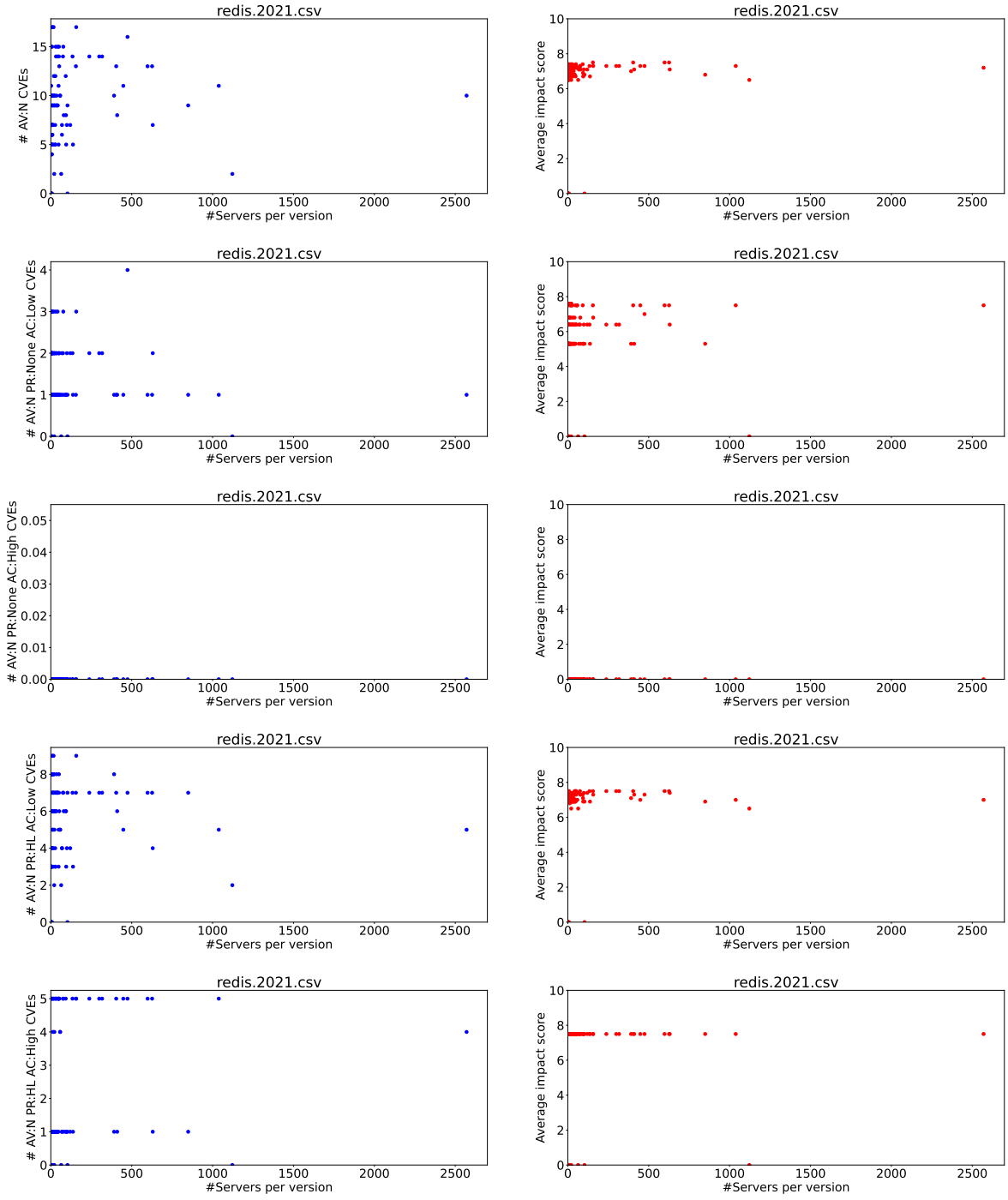


Figure B.5: Categorised vulnerabilities in Redis versions and their impact scores in 2021.

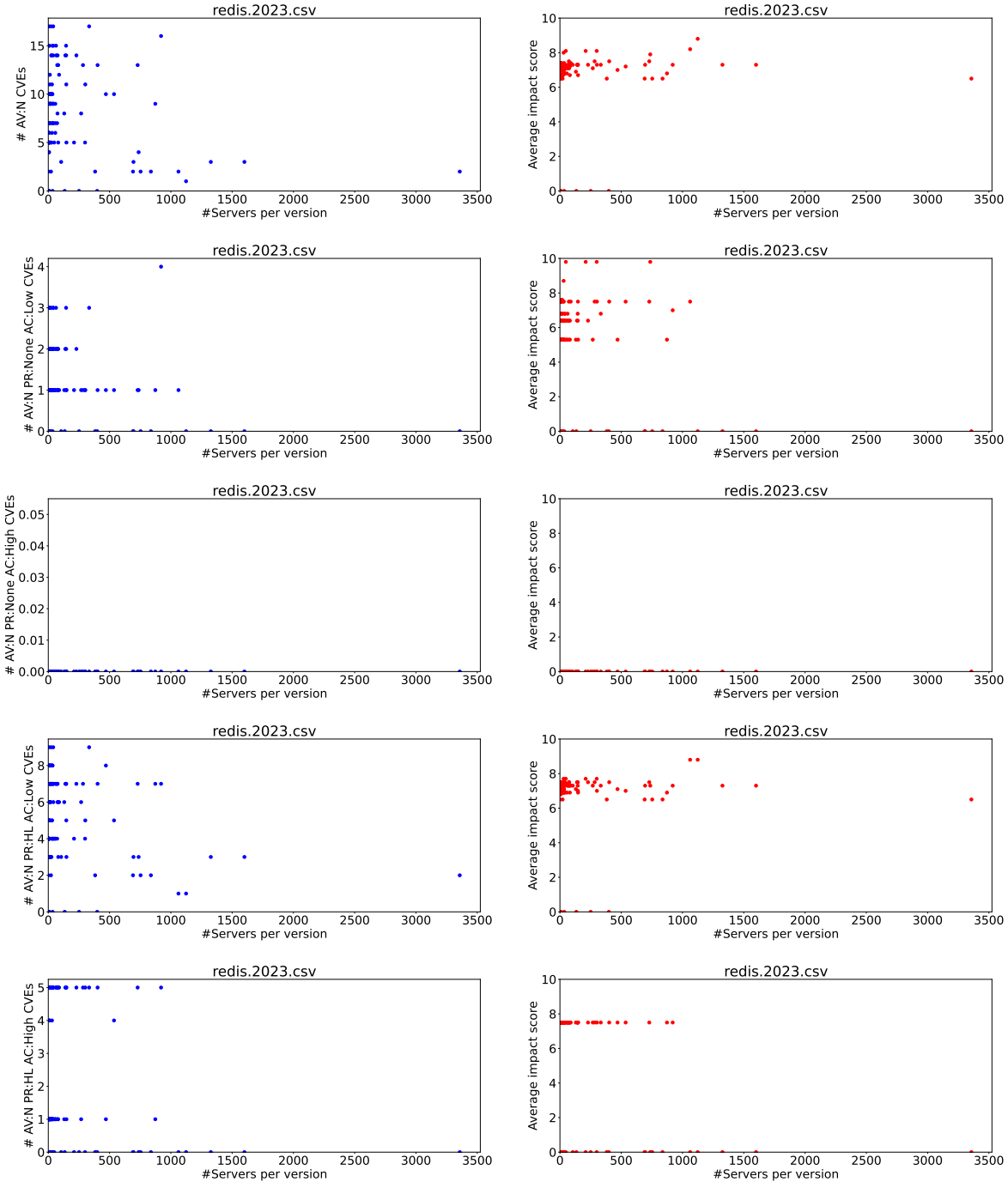


Figure B.6: Categorised vulnerabilities in Redis versions and their impact scores in 2023.

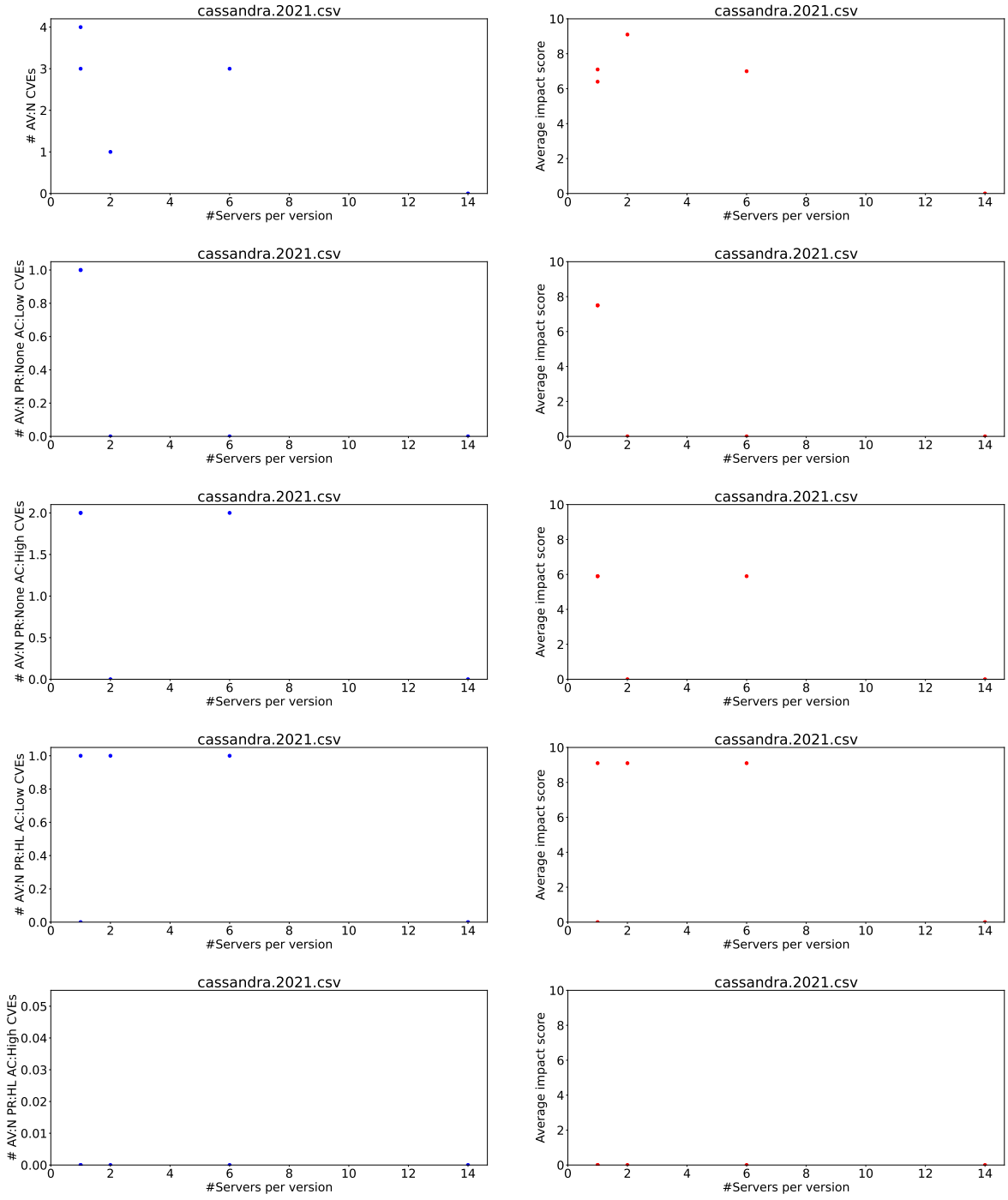


Figure B.7: Categorised vulnerabilities in Cassandra versions and their impact scores in 2021.

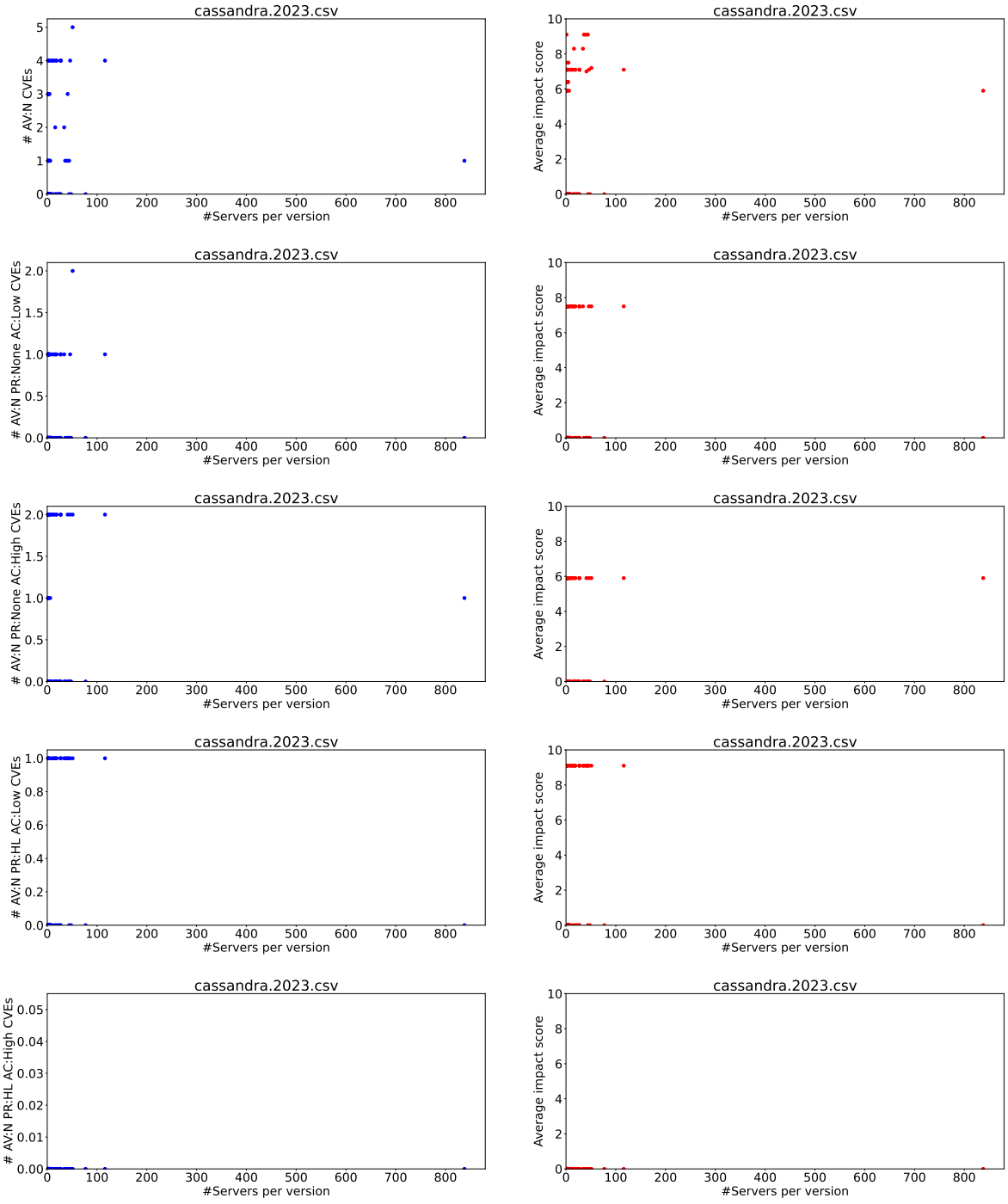


Figure B.8: Categorised vulnerabilities in Cassandra versions and their impact scores in 2023.

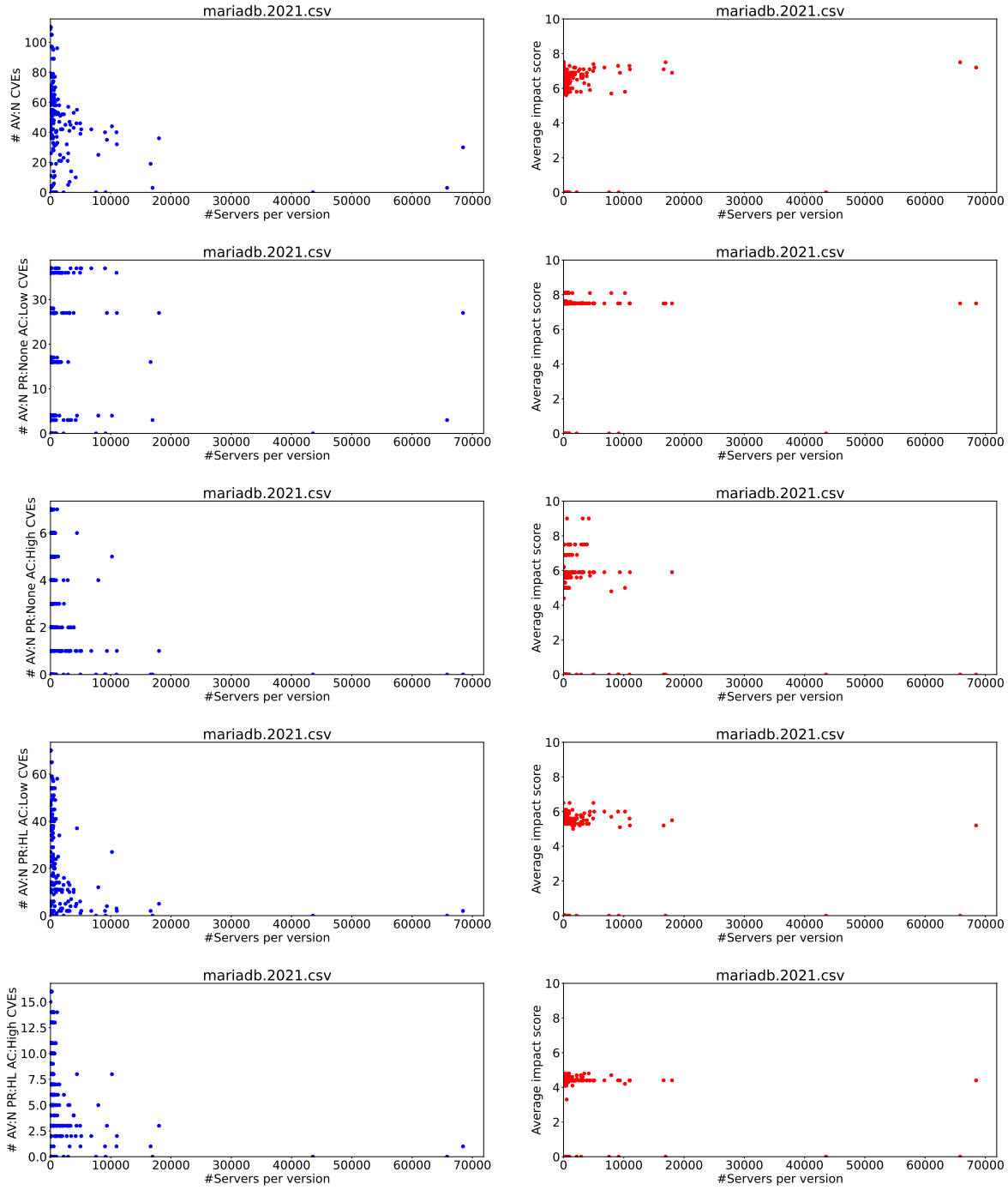


Figure B.9: Categorised vulnerabilities in MariaDB versions and their impact scores in 2021.

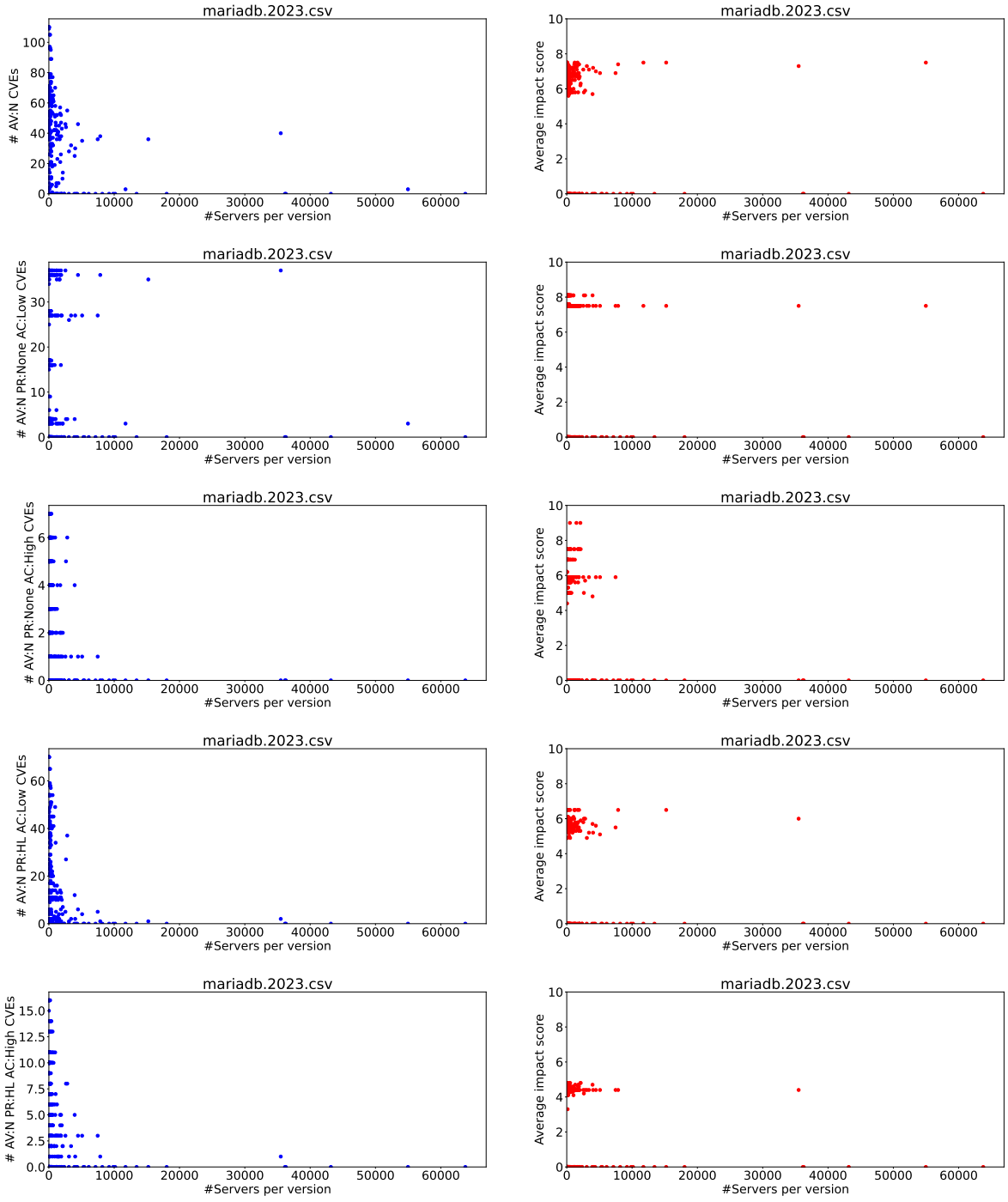


Figure B.10: Categorised vulnerabilities in MariaDB versions and their impact scores in 2023.

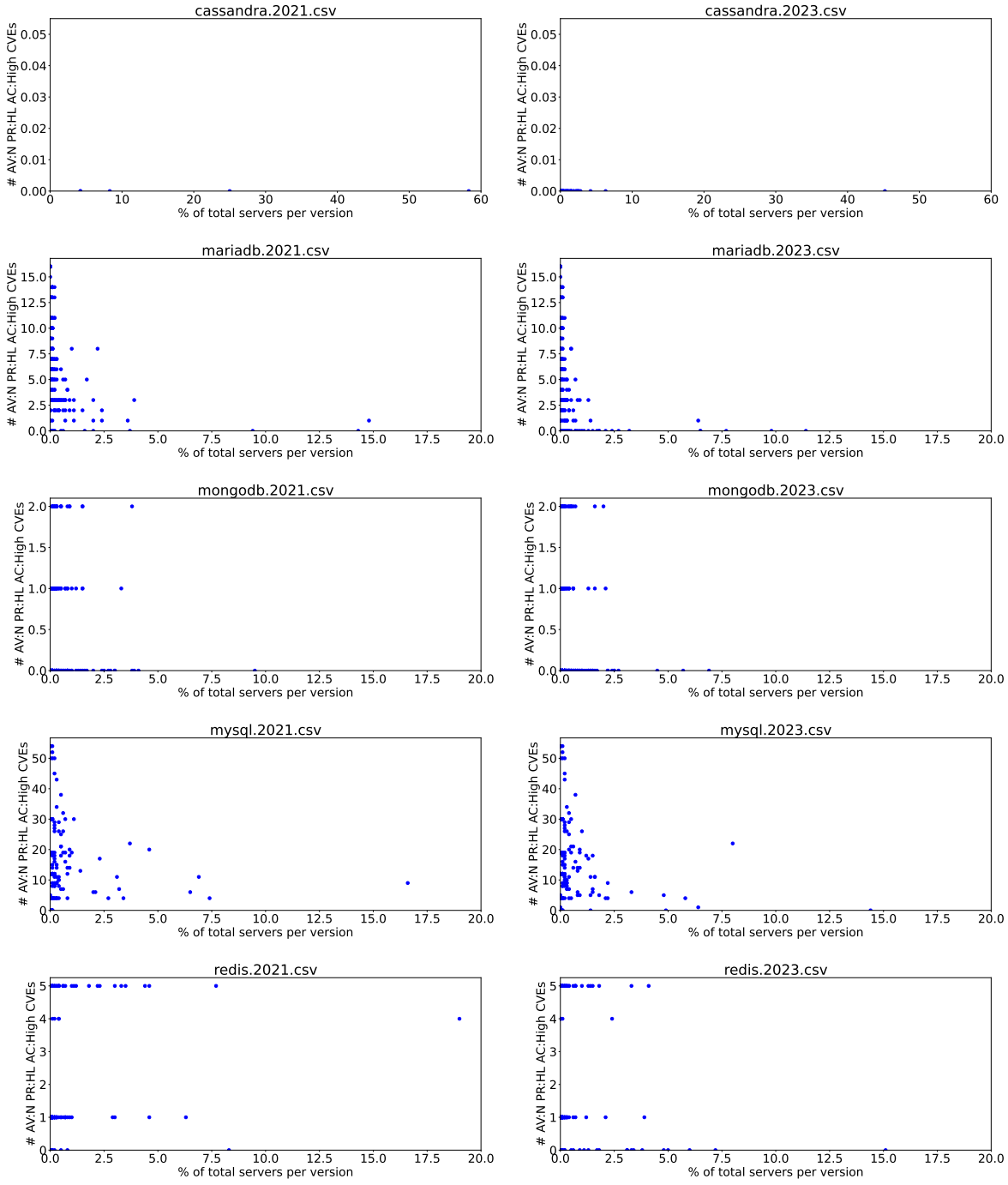


Figure B.11: AV:N PR:HL AC:High vulnerabilities across all database solutions

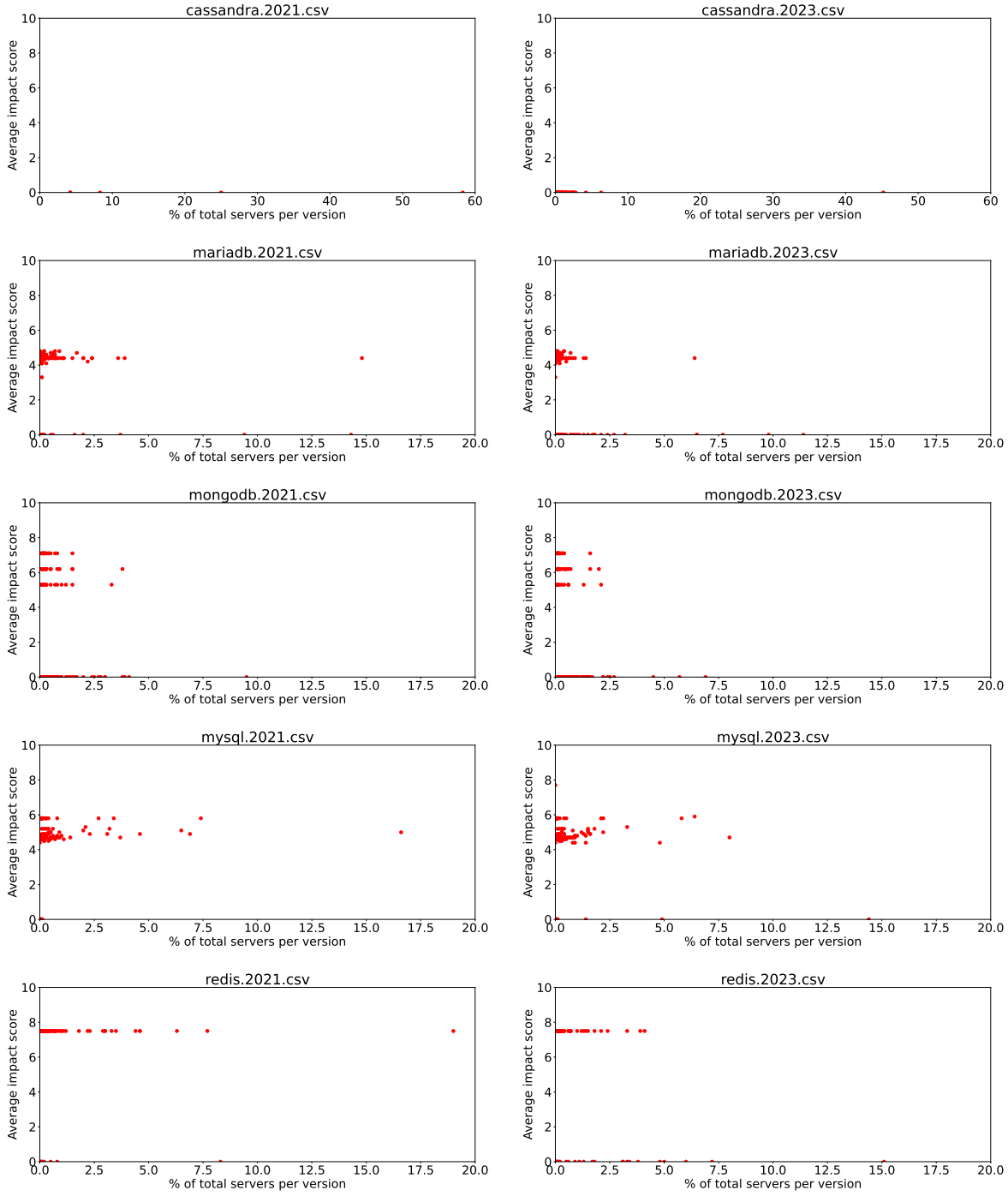


Figure B.12: Average impact score of AV:N PR:HL AC:High vulnerabilities across all database solutions

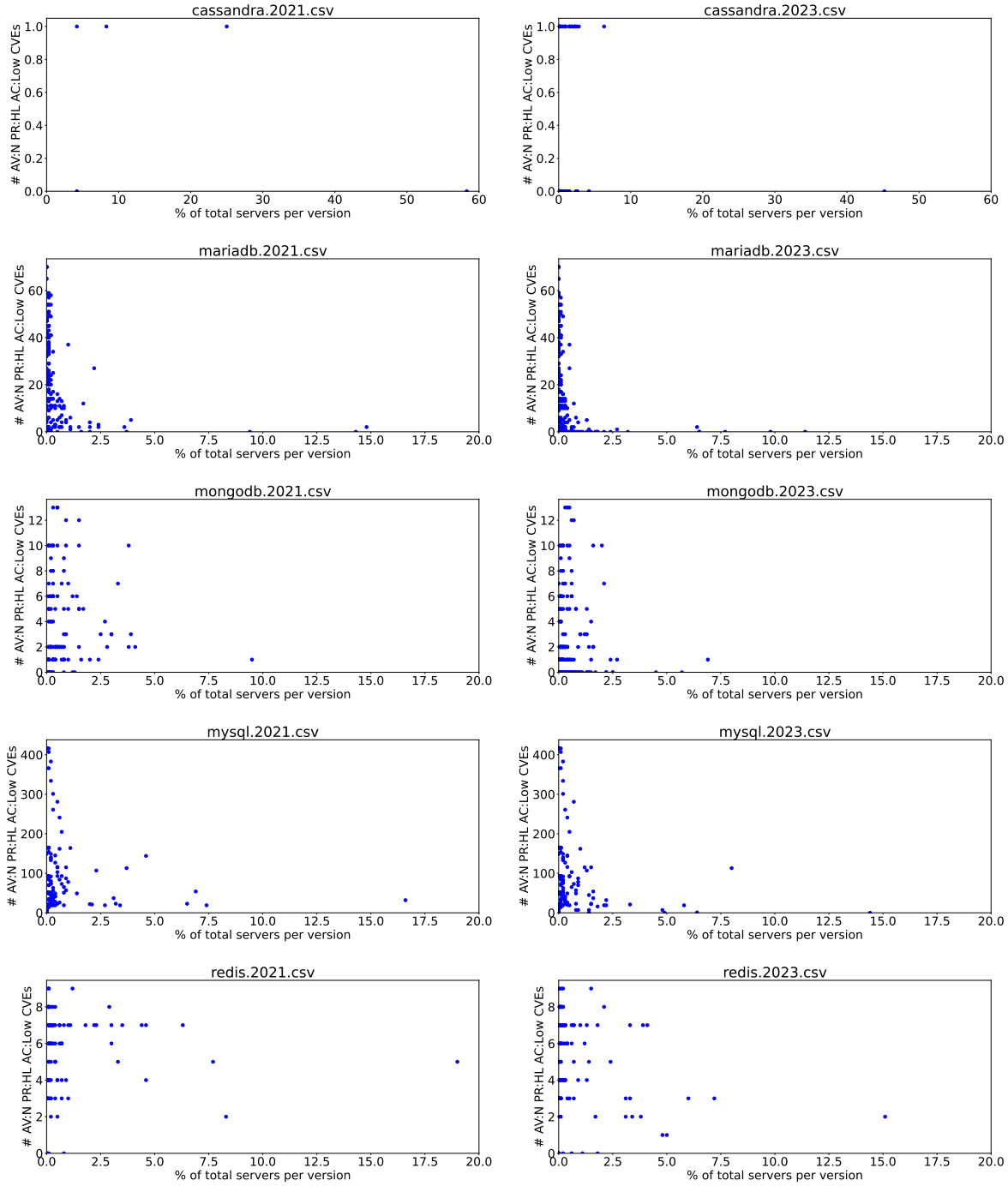


Figure B.13: AV:N PR:HL AC:Low vulnerabilities across all database solutions

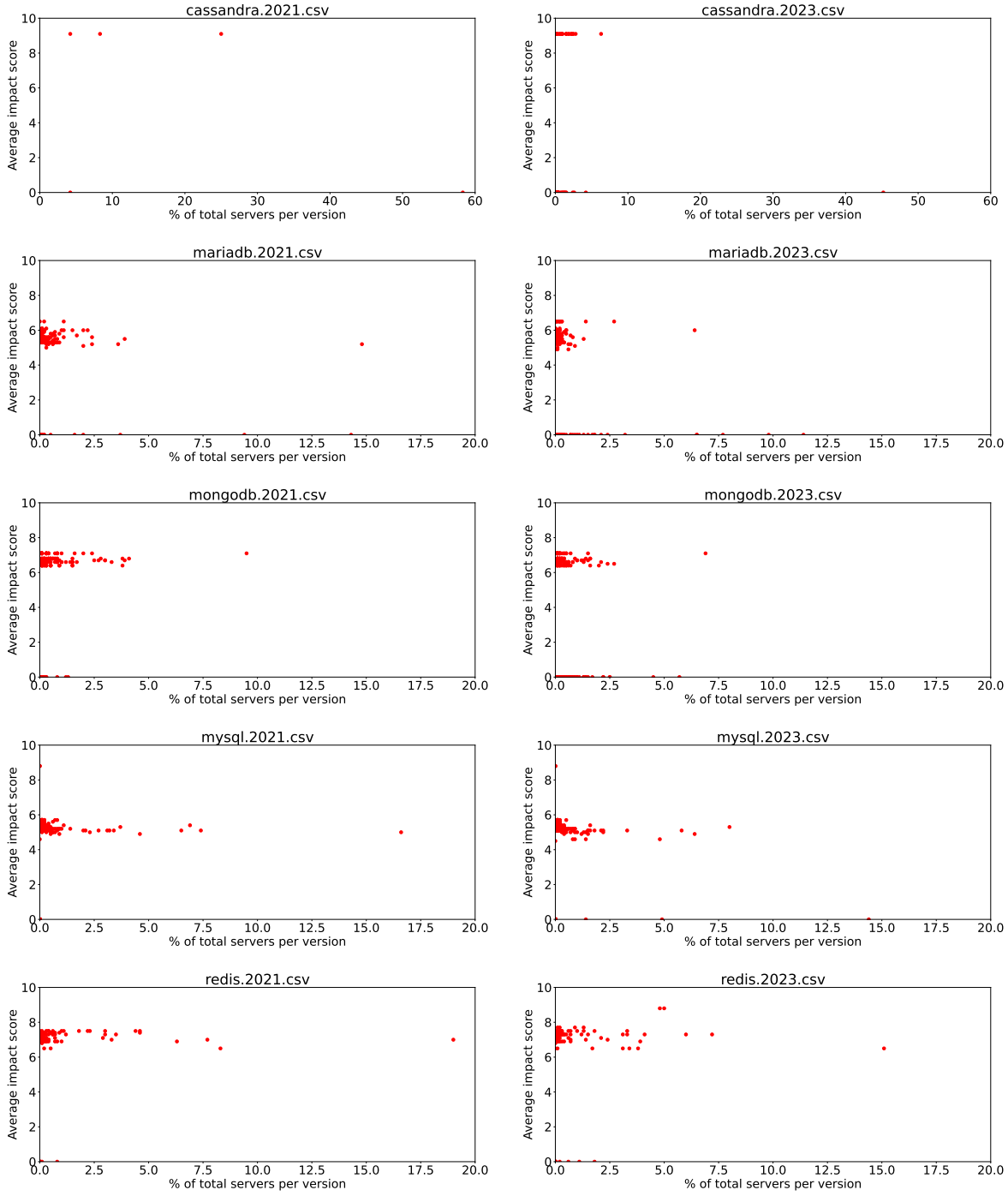


Figure B.14: Average impact score of AV:N PR:HL AC:Low vulnerabilities across all database solutions

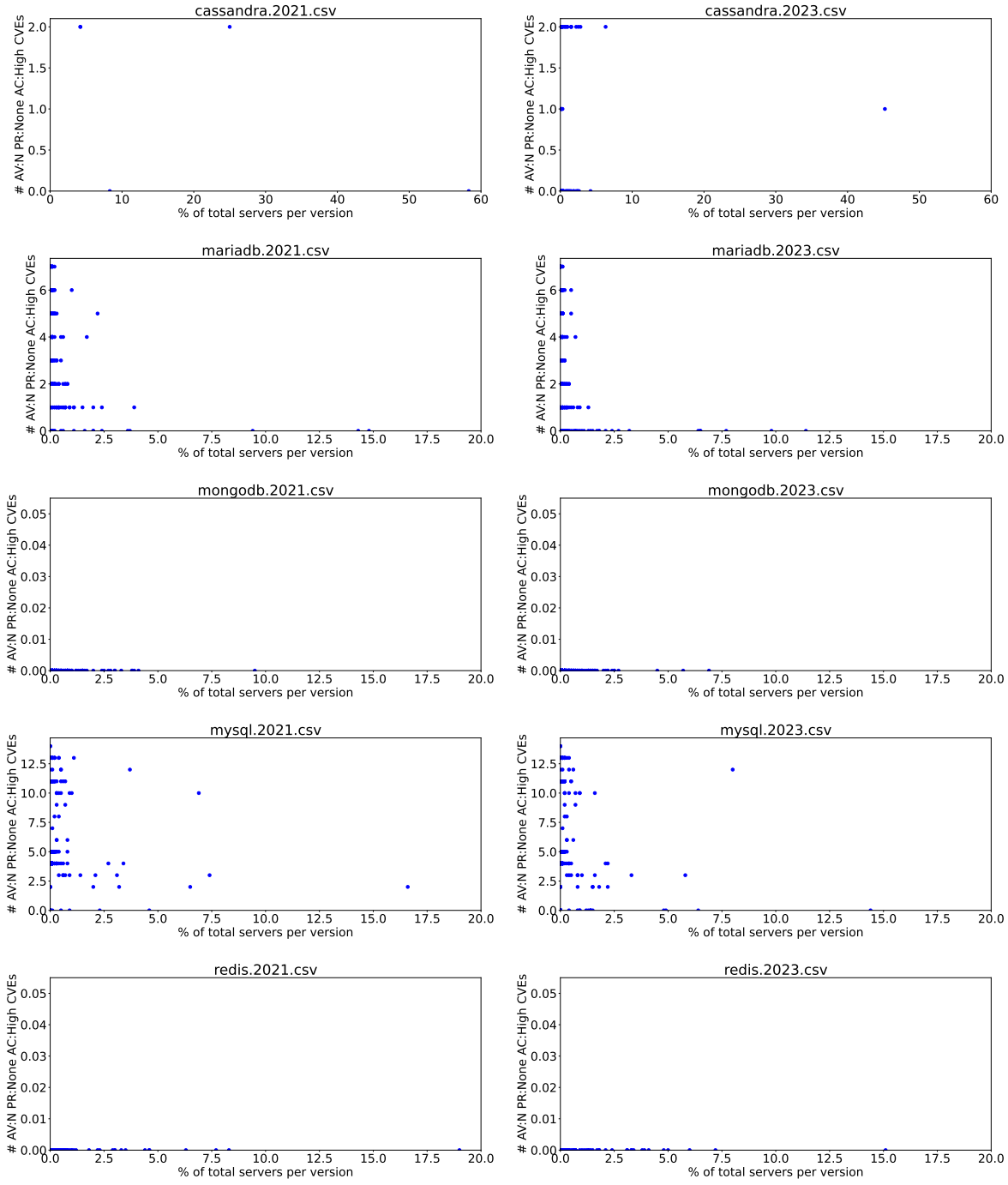


Figure B.15: AV:N PR:None AC:High vulnerabilities across all database solutions

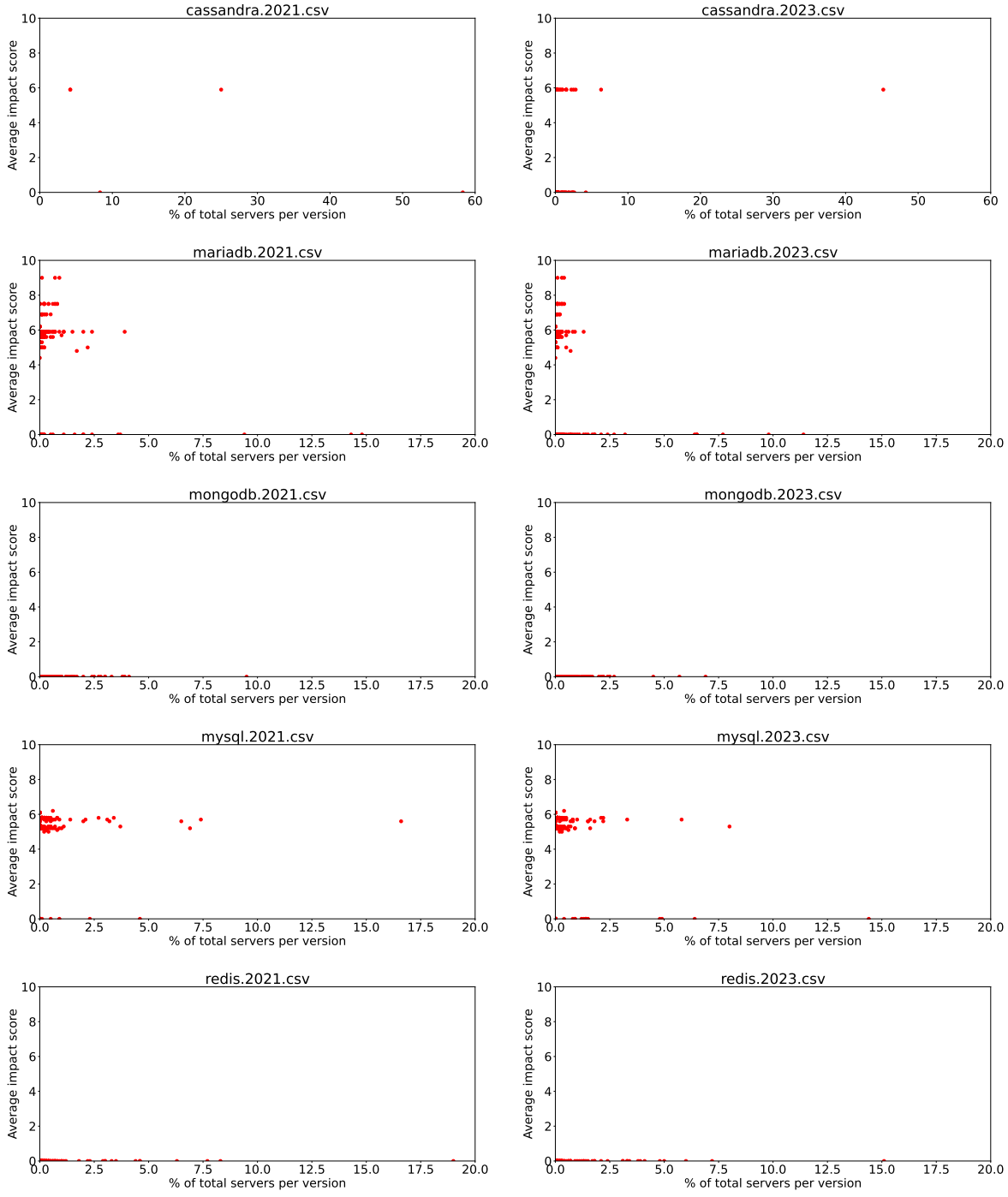


Figure B.16: Average impact score of AV:N PR:None AC:High vulnerabilities across all database solutions

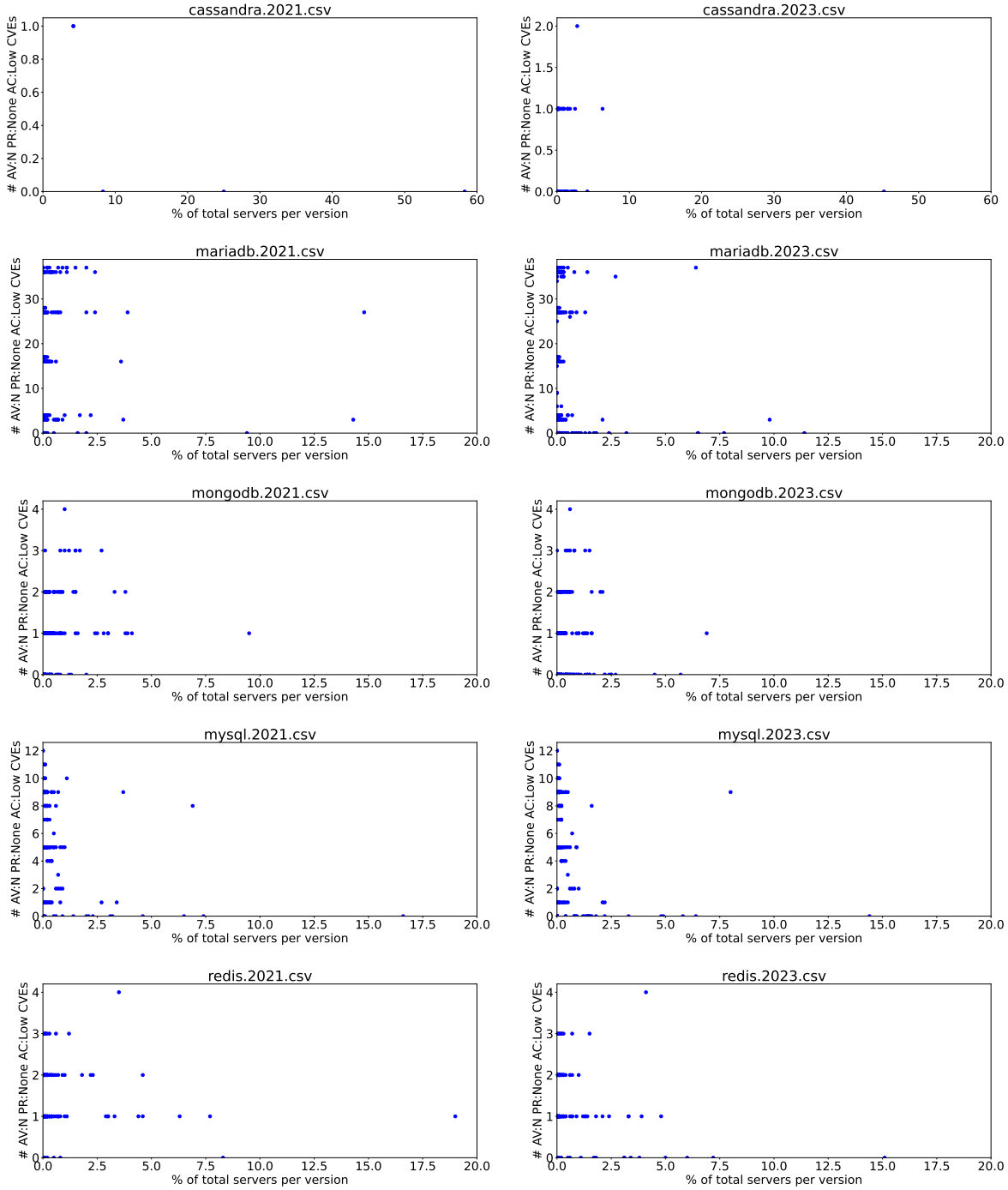


Figure B.17: AV:N PR:None AC:Low vulnerabilities across all database solutions

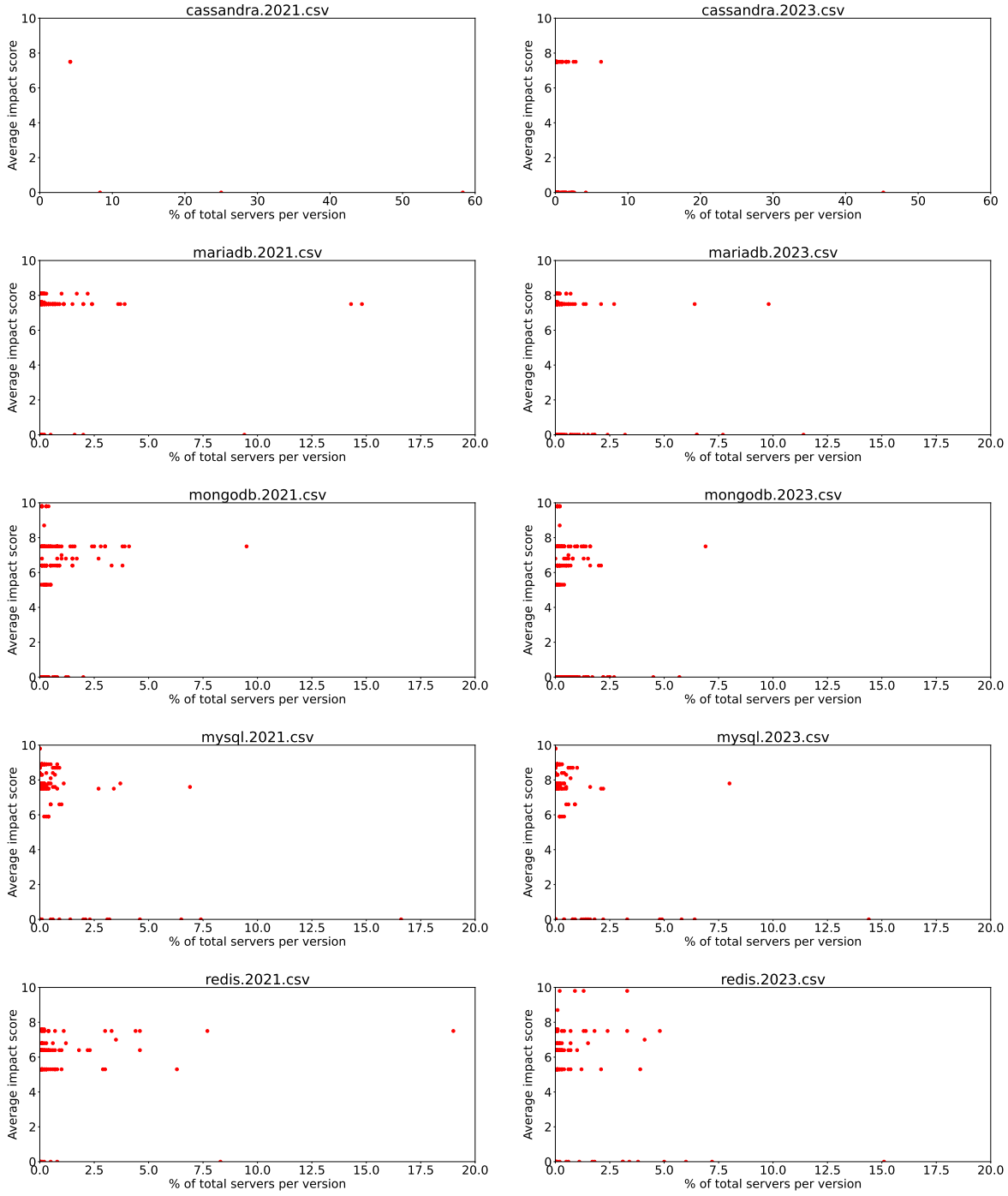


Figure B.18: Average impact score of AV:N PR:None AC:Low vulnerabilities across all database solutions