The State of Vulnerabilities in Linux Distributions

FOSSA

Containers have exploded in popularity within the last decade. They offer developers and organizations agility in the development cycle, accelerating application development, simplifying deployment, and consuming fewer computing resources.

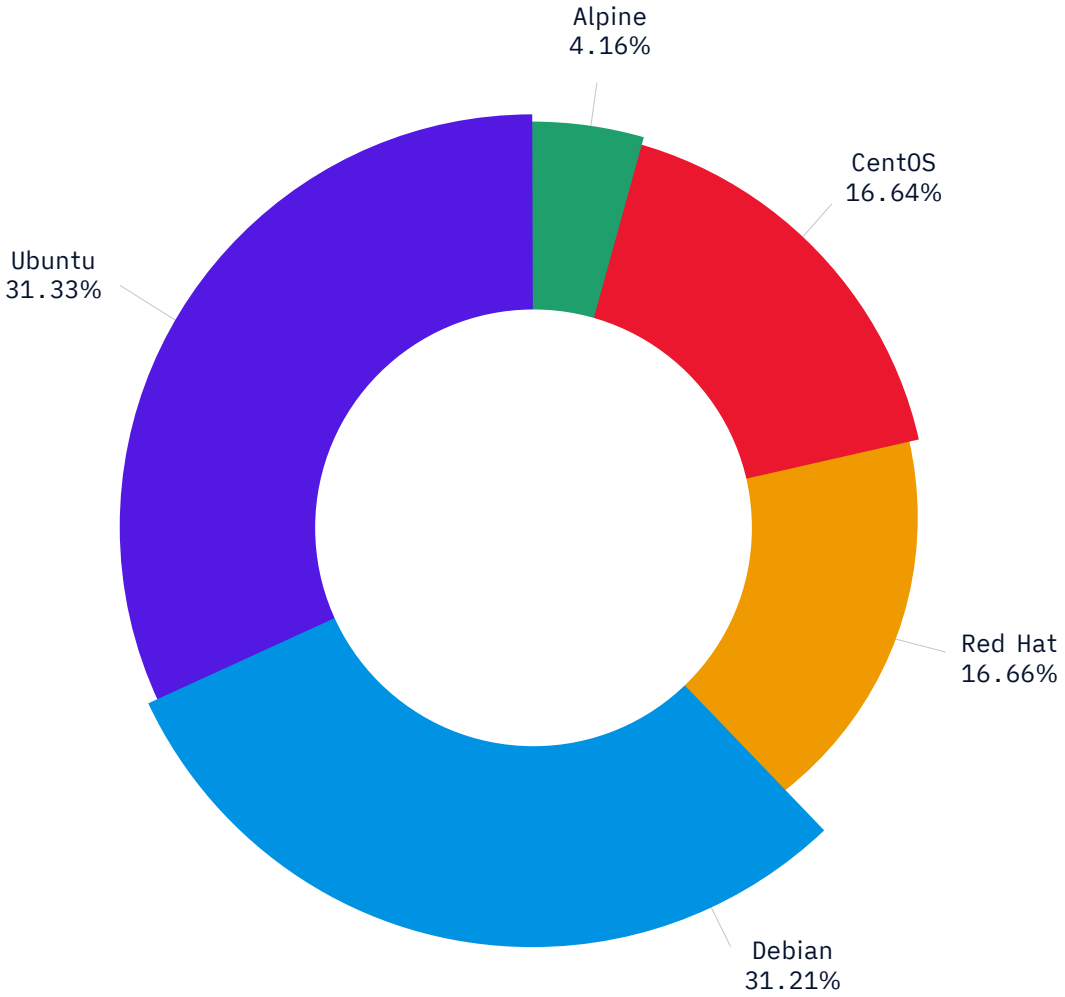
However, as container usage has grown, container image security has become more of a priority. In fact, a recent report revealed that over two million container images hosted on the Docker Hub repository are afflicted with at least one critical vulnerability.

Containers are composed of a base image and additional layers which may contain dependencies, applications, or other filesystem changes. Most developers start with a base image and then build/add layers on top of it. Popular base images are basic or minimal Linux distributions, such as Debian, Ubuntu, Red Hat, Centos, or Alpine.

In this report, we will take an in-depth look at the security vulnerabilities found in the most popular Linux distributions, including analysis of:

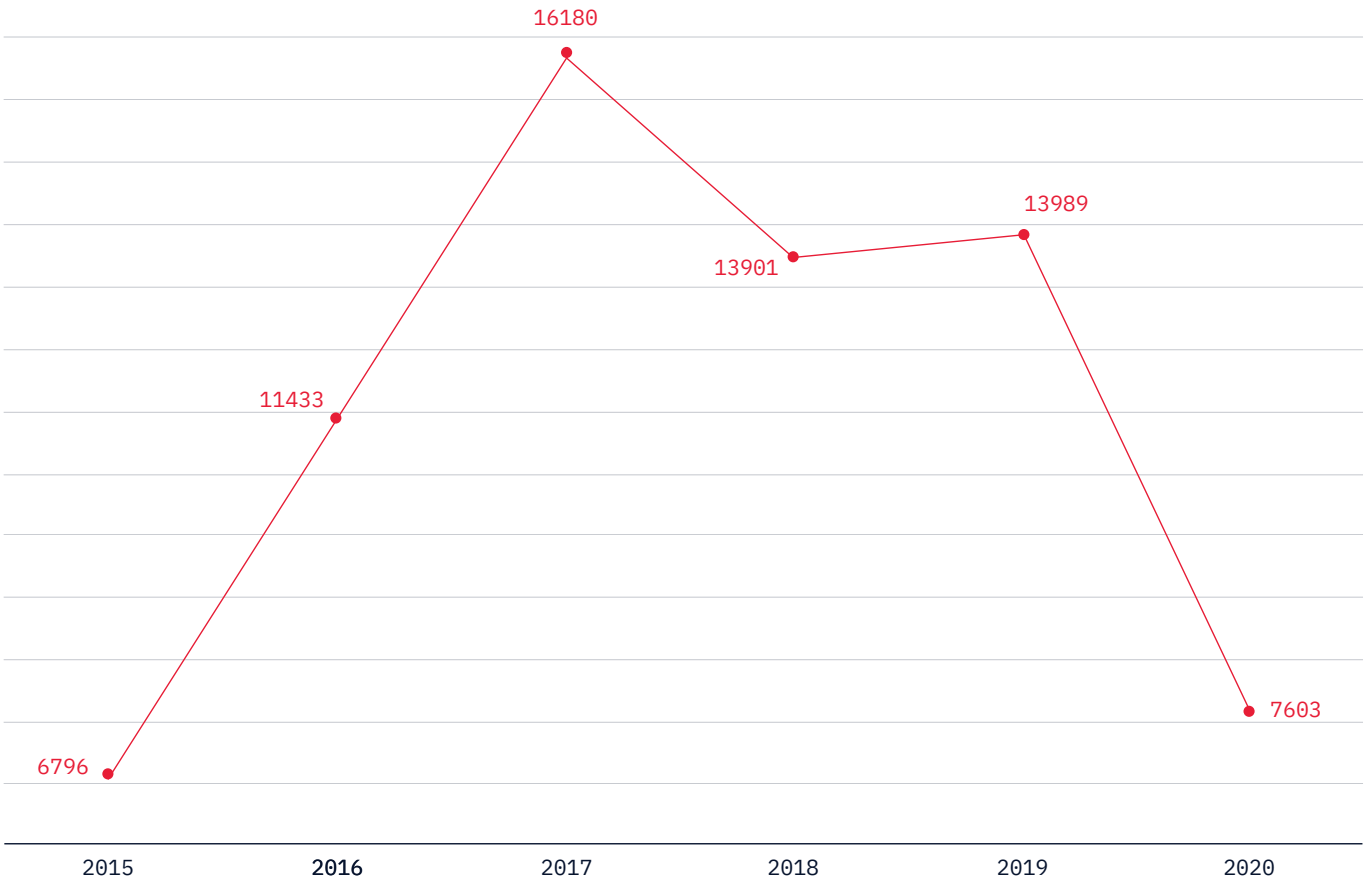
- The most common vulnerabilities
- The distribution of vulnerabilities
- Longitudinal trends in vulnerabilities over several years
- The most prevalent CWEs in each Linux distribution

Vulnerabilities Across Linux Distributions



Alpine has the fewest number of vulnerabilities across the above Linux distributions. Considering that Alpine has a smaller footprint and comes with built-in security protections, this is not too surprising. Debian is one of the most popular distributions, so the high number of vulnerabilities might be expected. Ubuntu is based on Debian, so it should also come as no shock that the two distributions have similar vulnerability counts. CentOS is based on Red Hat and both of these distributions have a similar number of vulnerabilities as well.

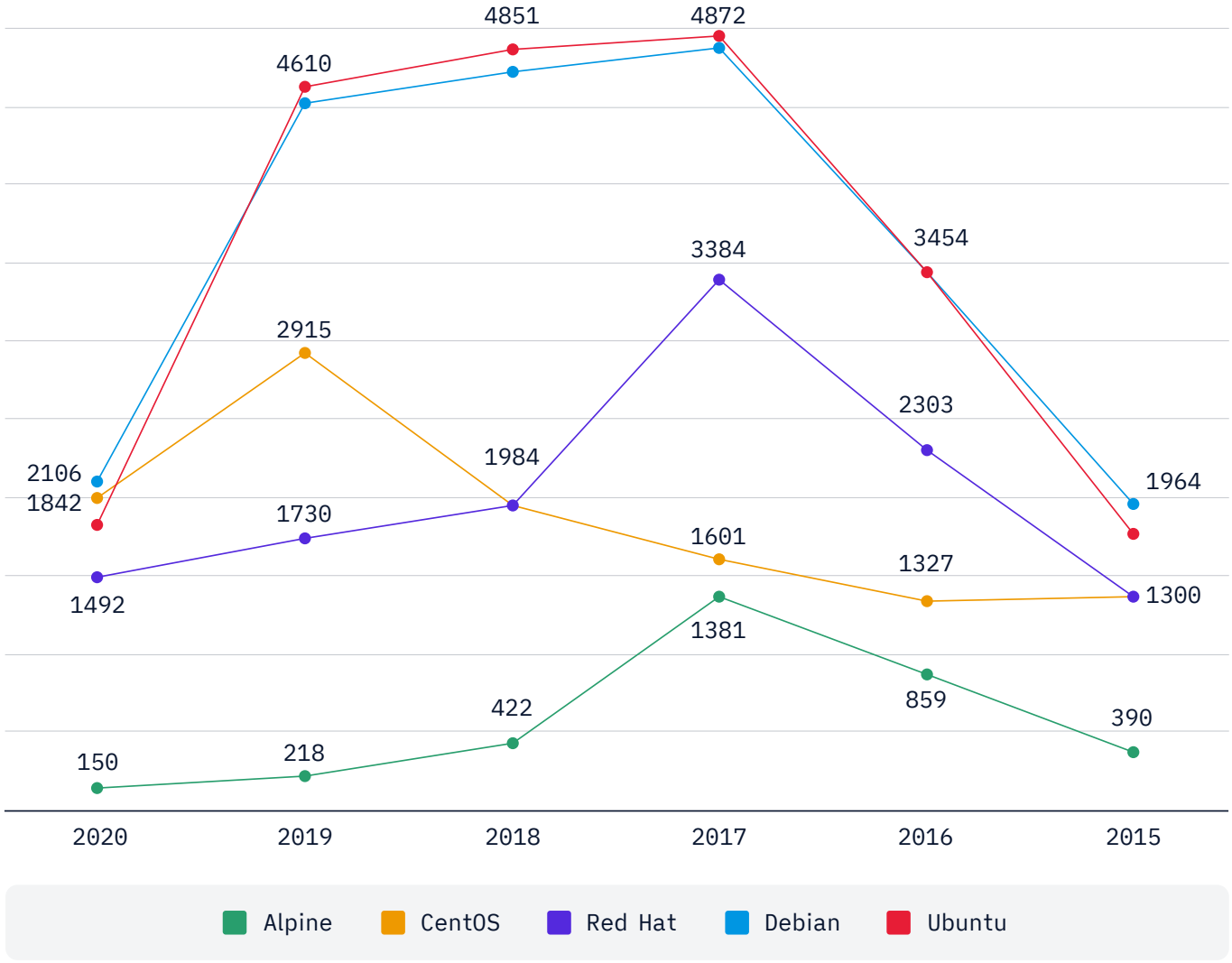
Vulnerability Counts Over Time



Alpine, CentOS, Red Hat Debian & Ubuntu distributions only

As container and Docker adoption has grown over the past decade, so too has the number of vulnerabilities in the distributions. Container adoption picked up in 2015/2016, and we can see that the number of vulnerabilities peaked in 2017. 2020 saw a steep drop in vulnerabilities, but we need more data before we can say if it was the start of a trend or an anomaly.

Vulnerabilities in Container Distributions by Year



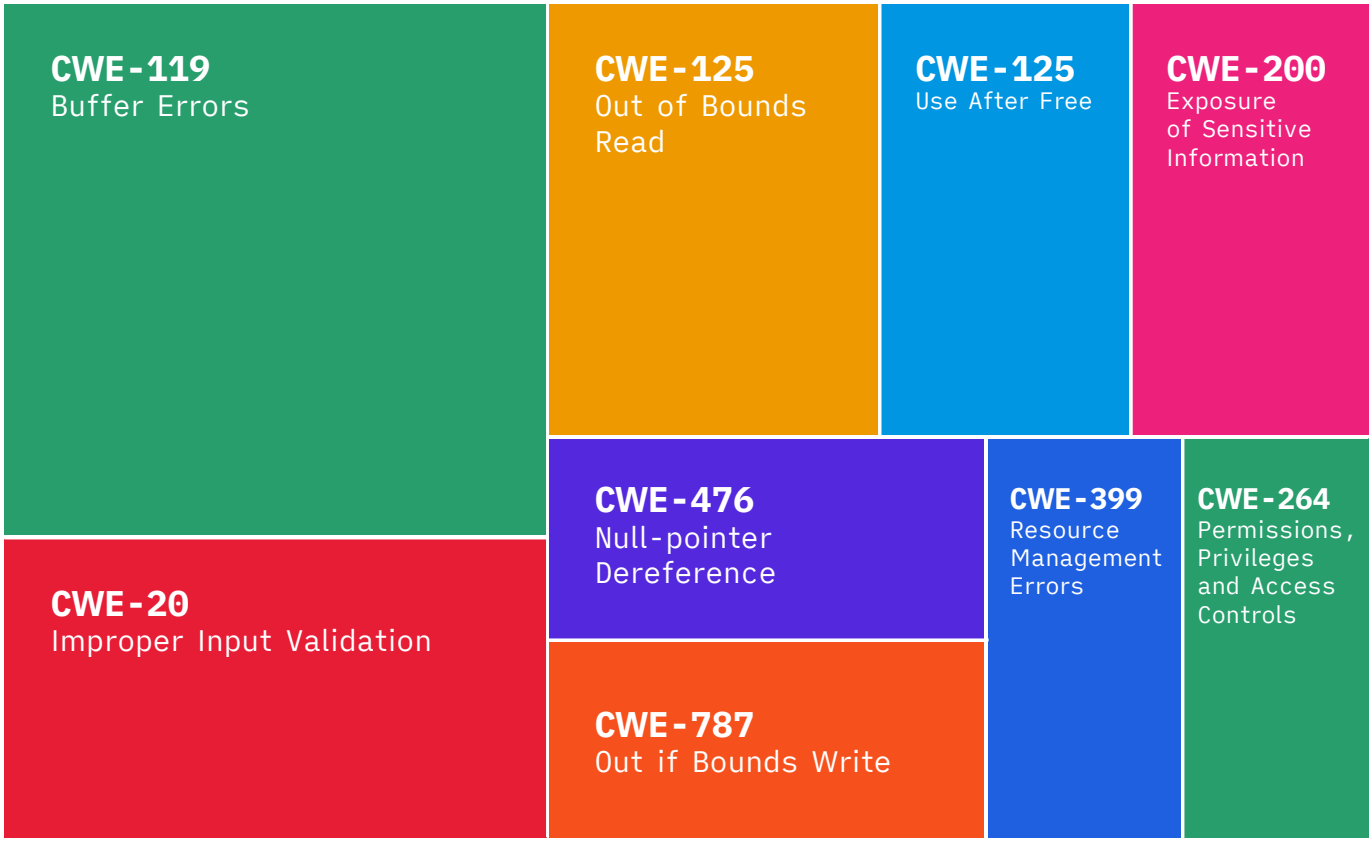
All distributions saw fewer vulnerabilities in 2020, though more data is needed to determine if this was the start of a lasting trend or more transitory. CentOS experienced a consistent upward trend in the past 5 years, while Red Hat saw fewer vulnerabilities starting in 2017. Debian and Ubuntu had similar vulnerability counts, which is understandable given that Ubuntu is based on Debian.

Most Frequently Seen Vulnerabilities

Now, let's look at some of the most commonly found vulnerabilities in these Linux distributions.

Buffer Errors (CWE-119) topped our list of most frequently seen vulnerabilities by a large amount. We also discovered numerous instances of CWE-125 (Out-of-bounds Read) and CWE-787 (Out-of-bounds Write) across the Linux distributions.

It is important for developers to be aware of frequently observed coding errors that result in vulnerabilities and use best practices to avoid such issues.



CWE 119: Improper Restriction of Operations within the Bounds of a Memory Buffer

CWE 119 is one of the most prevalent vulnerabilities out there and is even included in MITRE's top 25 most dangerous software weaknesses. MITRE describes CWE-119 as follows: "The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer." Developers using programming languages without memory management support should be extra cautious and incorporate tools that detect and mitigate these buffer overflows.

Related weaknesses include Range Error (CWE-118), Classic Buffer Overflow (CWE-120), and Expired Pointer Dereference (CWE-825).

CWE 125: Out-of-bounds Read

This ranks among the top 3 in MITRE's top 25 most dangerous software weaknesses and belongs to the same family of vulnerabilities as CWE-119. In fact, CWE-125 is classified as a child of CWE-119. Per MITRE's description, this vulnerability occurs when "the software reads data past the end, or before the beginning, of the intended buffer."

Typically, this can allow attackers to read sensitive information from other memory locations or cause a crash. It's not surprising that this is a common issue, as many vulnerabilities are from system libraries, often written in C or C++, which may have less memory safety built-in.

Related weaknesses include Buffer Over-read (CWE-126), Buffer Under-read (CWE-127), and Untrusted Pointer Dereference (CWE-822).

CWE 20: Improper Input Validation

This is a class-level weakness where the product does not validate or incorrectly validates the input. Attackers who exploit this vulnerability can craft an input that is not expected by the rest of the application, leading to unexpected consequences. CWE-20 is among the most commonly found vulnerabilities across all languages.

Related weaknesses include Improper Neutralization (CWE-707), Injection Errors (CWE-74), and Path Traversal errors (CWE-22).

CWE 787: Out-of-bounds Write

This sits on top of MITRE's top 25 most dangerous software weaknesses list. According to MITRE, this vulnerability occurs when "software writes past the end, or before the beginning, of the intended buffer."

Related weaknesses include Stack-based buffer overflow (CWE-121), Heap-based Buffer Overflow (CWE-122), and Access of Uninitialized Pointer (CWE-824).

CWE 476: Null Pointer Dereference

This occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit. Developers can avoid such errors by sanity checking each pointer before use.

Related weaknesses include Unchecked Return Value (CWE-252) and Pointer Issues (CWE-465).

CWE 200: Exposure of Sensitive Information to an Unauthorized Actor

CWE-200 describes the issue where an application unintentionally exposes private or sensitive information to users not explicitly authorized. MITRE advises developers to build compartmentalization into applications so that sensitive data can stay within those boundaries. As a general rule, developers should adopt principles of least privilege when granting access to any data within applications.

Related weaknesses include Observable Discrepancy (CWE-203) and Insertion of Sensitive Information Into Sent Data (CWE-201).

CWE-416: Use After Free

This vulnerability, which occurs when applications reference memory after it has been freed, can result in corruption of valid data, application crash, or execution of unauthorized code. The simplest way data corruption may occur involves the system's reuse of the freed memory. One potential mitigating practice would be to set all pointers to NULL as they are freed.

Related weaknesses include Double Free (CWE-415) and Single Handler Race Conditions (CWE-364).

CWE-399: Resource Management Errors

Per MITRE, CWE-399 refers to a category of weaknesses "related to improper management of system resources."

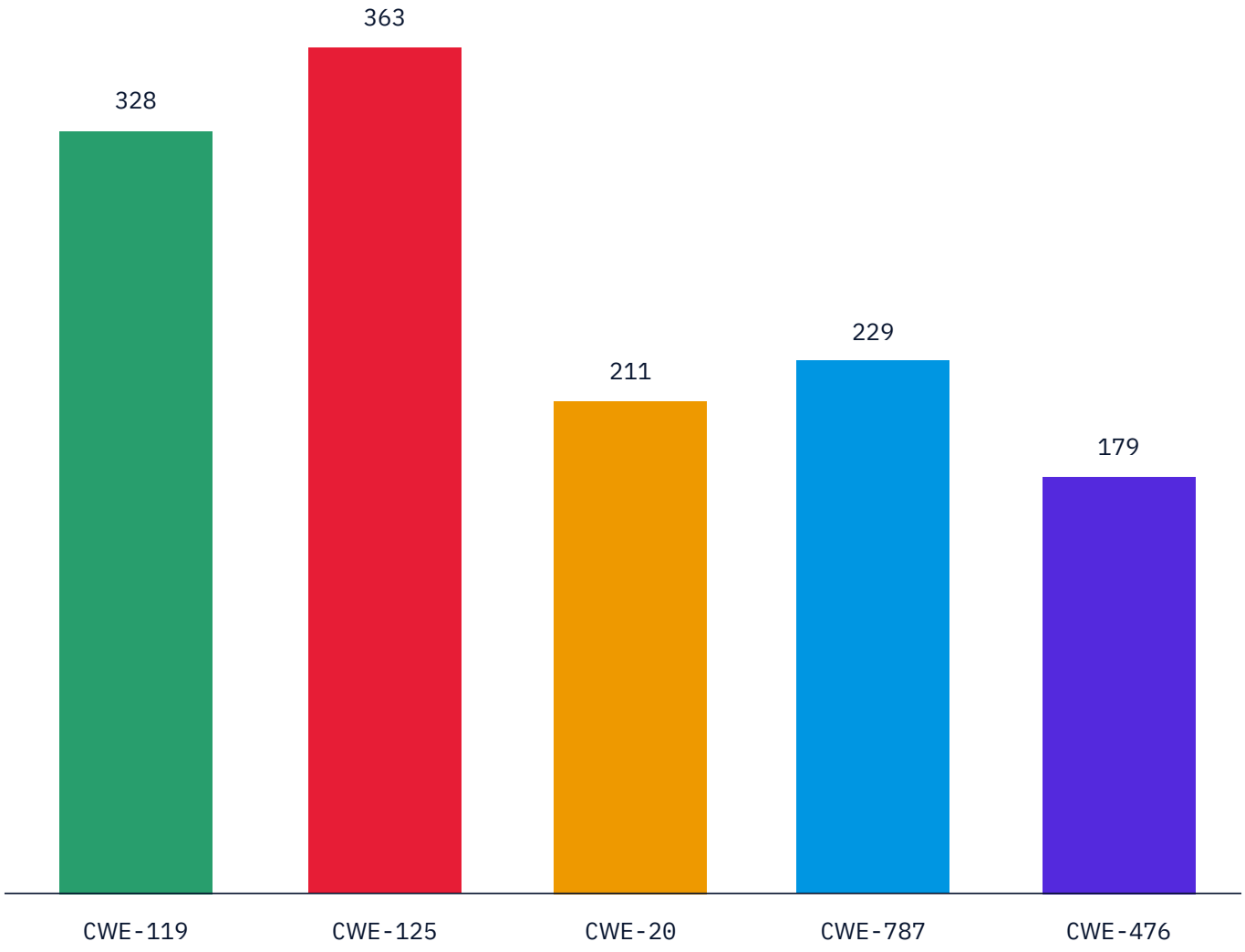
Related weaknesses include Insufficient Resource Pool (CWE-410) and Use of Uninitialized Resource (CWE-908).

CWE-264: Permissions, Privileges, and Access Controls

According to MITRE, “weaknesses in this category are related to the management of permissions, privileges, and other security features that are used to perform access control.”

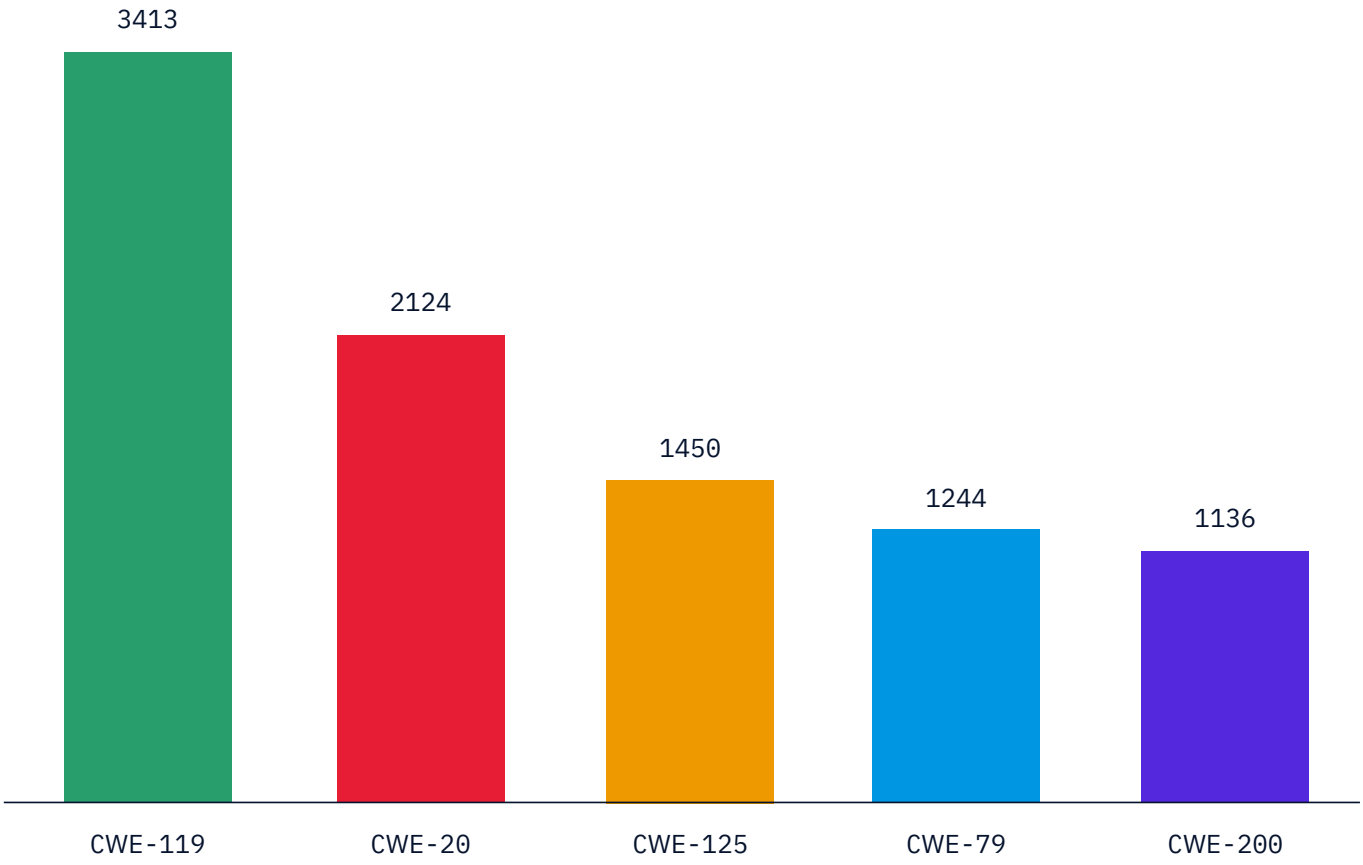
Related weaknesses include Permission Issues (CWE-275) and Incorrect User Management (CWE-286).

Most Common Vulnerabilities: Alpine



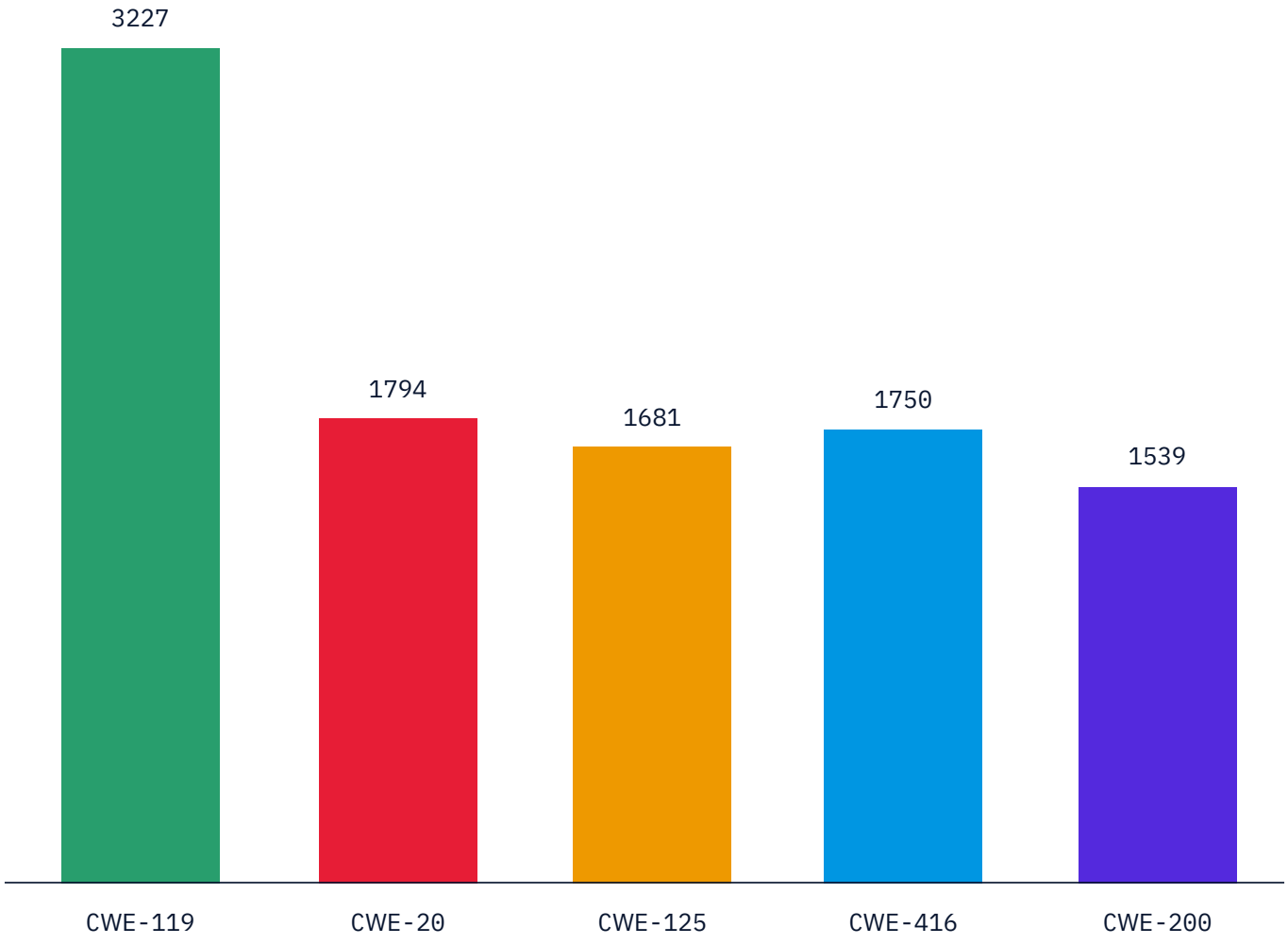
Out-of-bounds Read (CWE-125) and Buffer errors (CWE-119) were the two most common vulnerabilities in the Alpine distribution, both belonging to the family of buffer overflows. Out-of-bounds Write (CWE-787) errors were uniquely present in large numbers in Alpine distributions, so teams using Alpine should proactively check the size of the buffers they are writing into.

Most Common Vulnerabilities: Debian



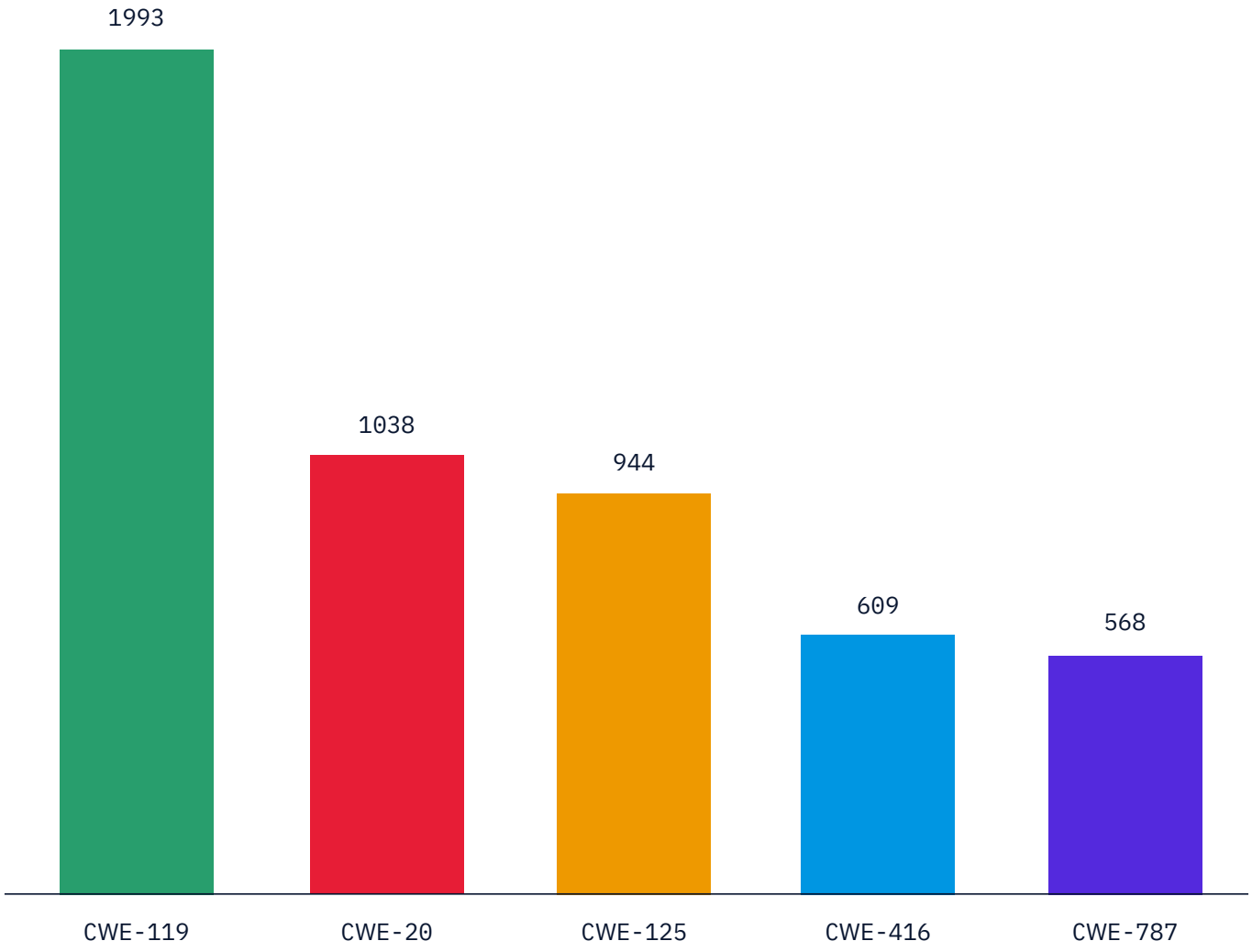
Debian had the highest number of vulnerabilities by count, and, like the other distributions we've discussed, CWE-119 (Buffer errors) was the most commonly found vulnerability.

Most Common Vulnerabilities: Ubuntu



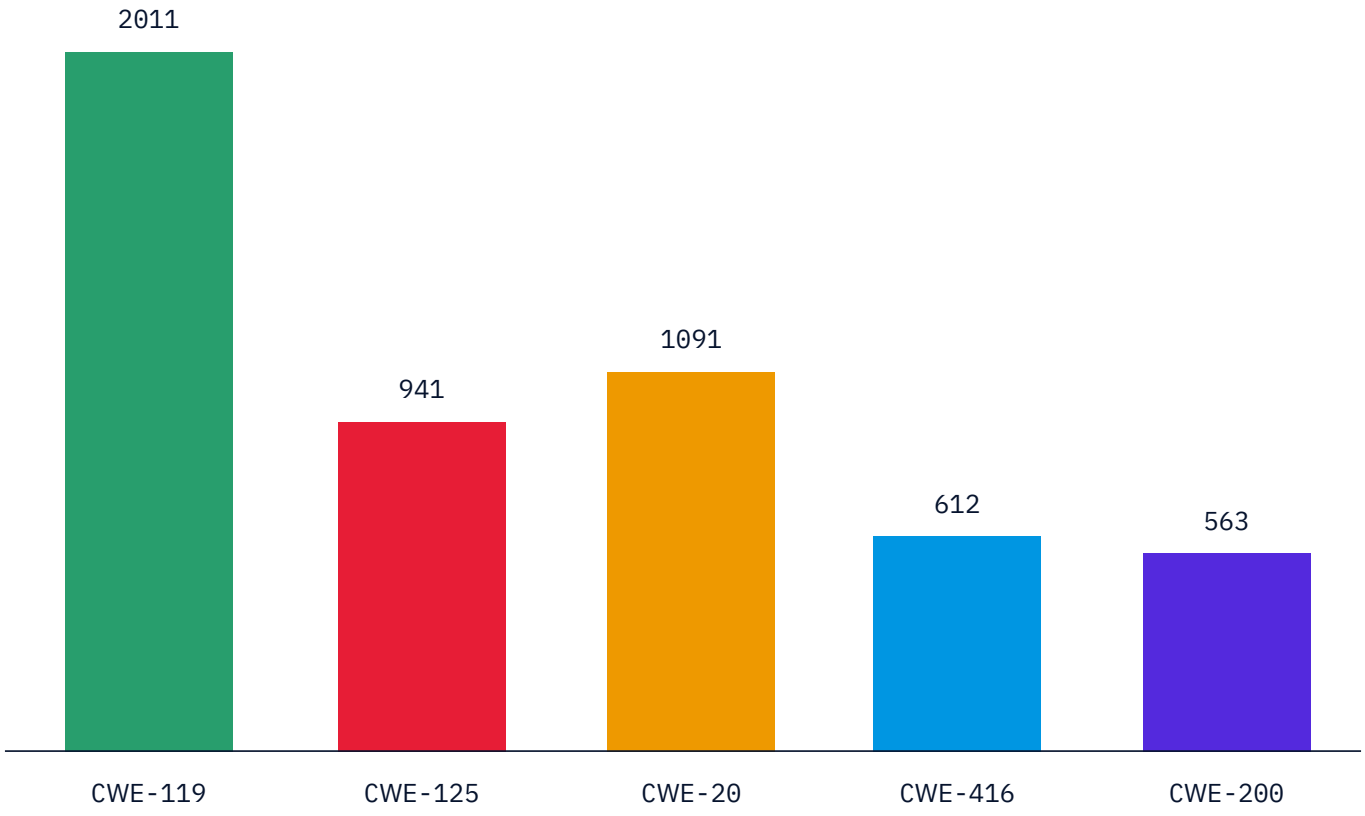
As expected, Ubuntu and Debian were impacted by similar numbers and types of vulnerabilities. This is most likely because of the extremely similar package architecture shared between both distributions. But Ubuntu had a higher number of Use After Free errors (CWE-416) vulnerabilities than Debian. In light of the large number of buffer errors (CWE-119) discovered in Ubuntu, developers should consider using static analysis tools to detect out-of-bound memory operations.

Most Common Vulnerabilities: CentOS



CentOS also saw a very high number of Buffer errors (CWE-119), which is expected. Out-of-bounds Write errors (CWE-787) were more prevalent in CentOS than in similar distributions (such as Red Hat).

Most Common Vulnerabilities: Red Hat

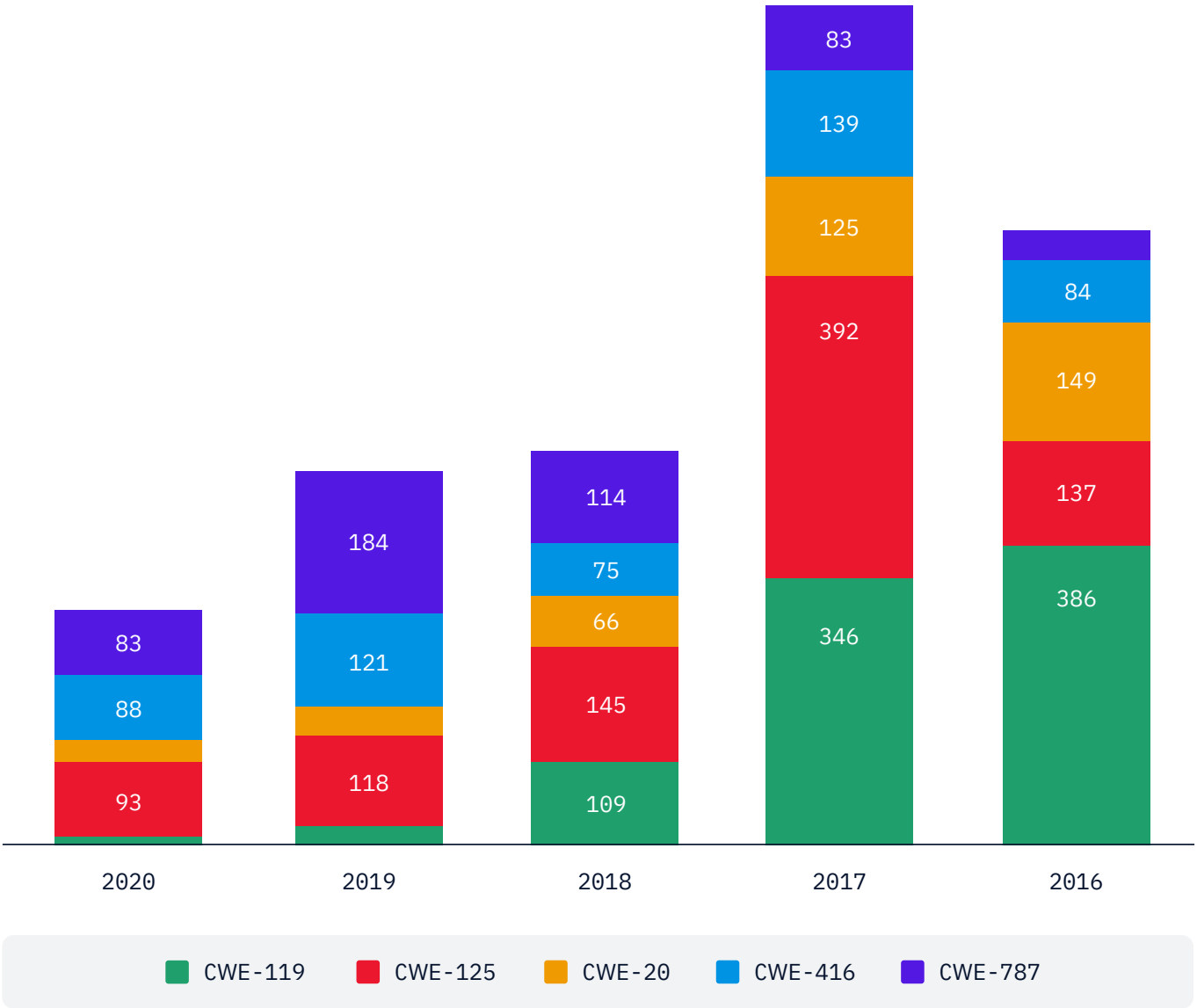


Red Hat and CentOS, not surprisingly, have a similar mix of vulnerabilities. The biggest difference is that Red Hat sees more Exposure of Sensitive Information errors (CWE-200), whereas CentOS sees more Out-of-bounds Write errors (CWE 787).

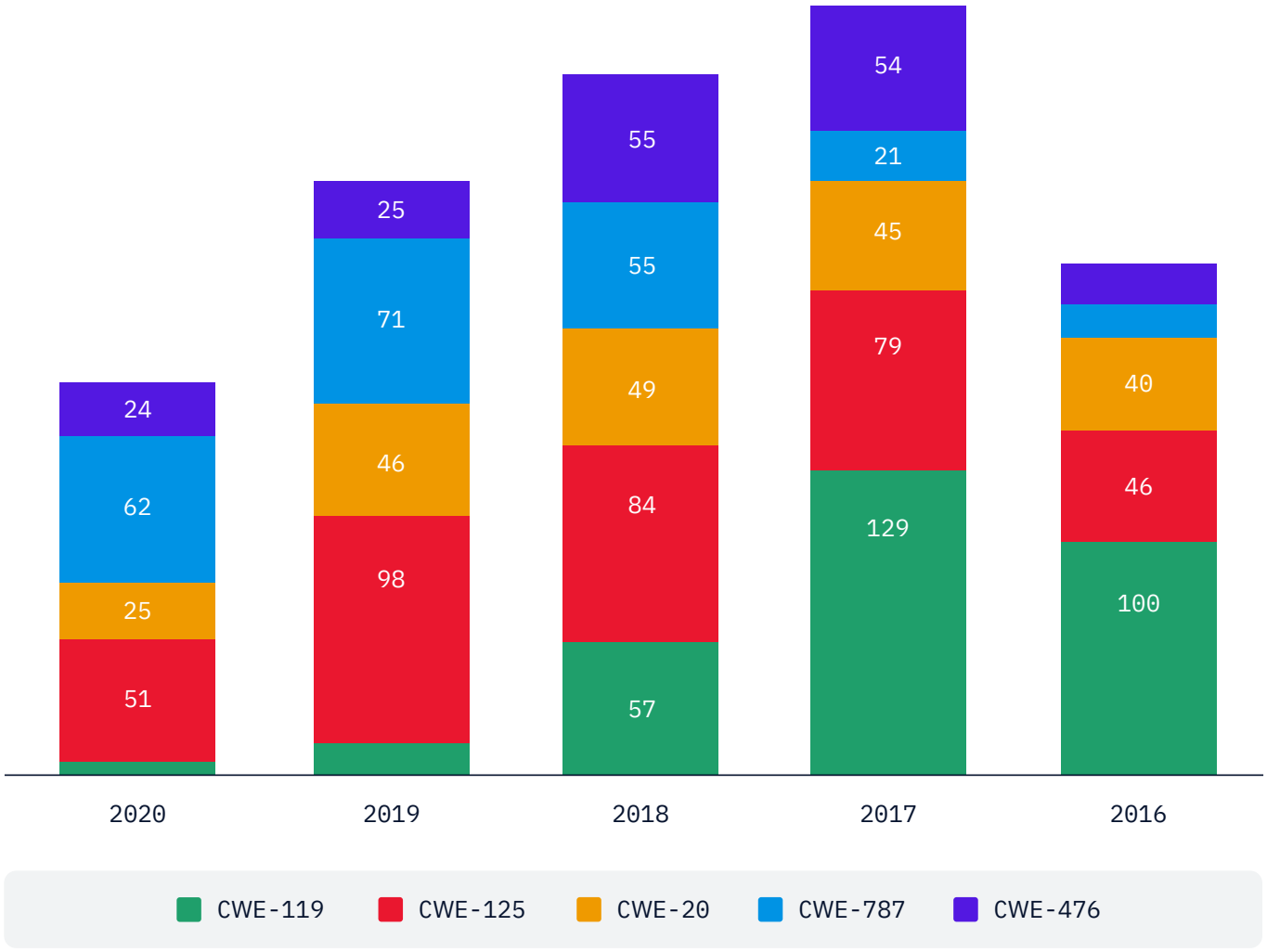
Compared to vulnerabilities in popular languages, the vulnerabilities found in the above Linux distributions lean more toward buffer overflows, improper input validation, exposure of sensitive information, and null pointer dereference. This is because there is less memory safety built into the C and C++ languages, the language of choice for many Linux system libraries.

Let's take a deeper look at the top vulnerabilities in these distributions and see how they trended over our five-year sample.

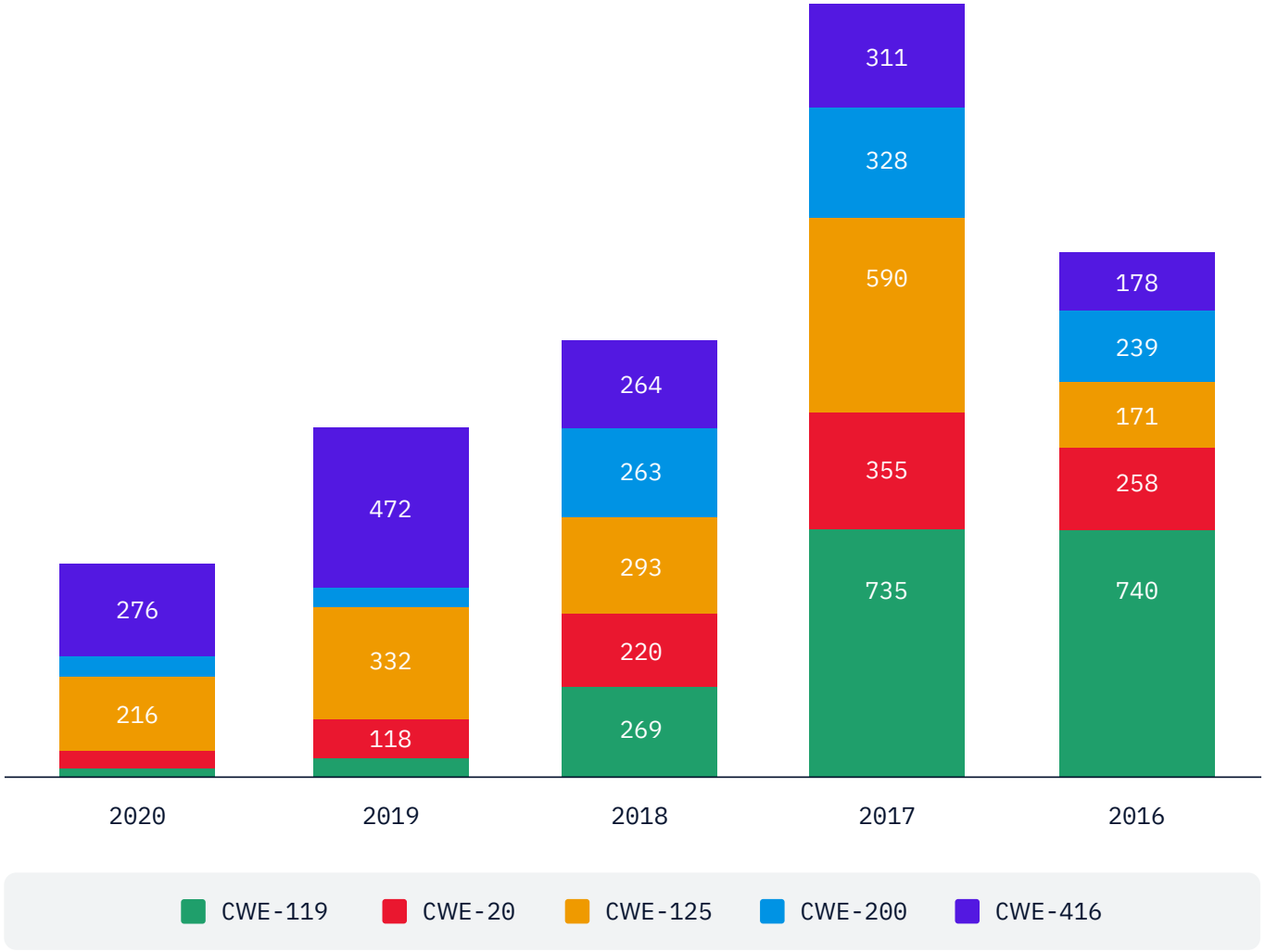
CentOS: Trends Over the Years



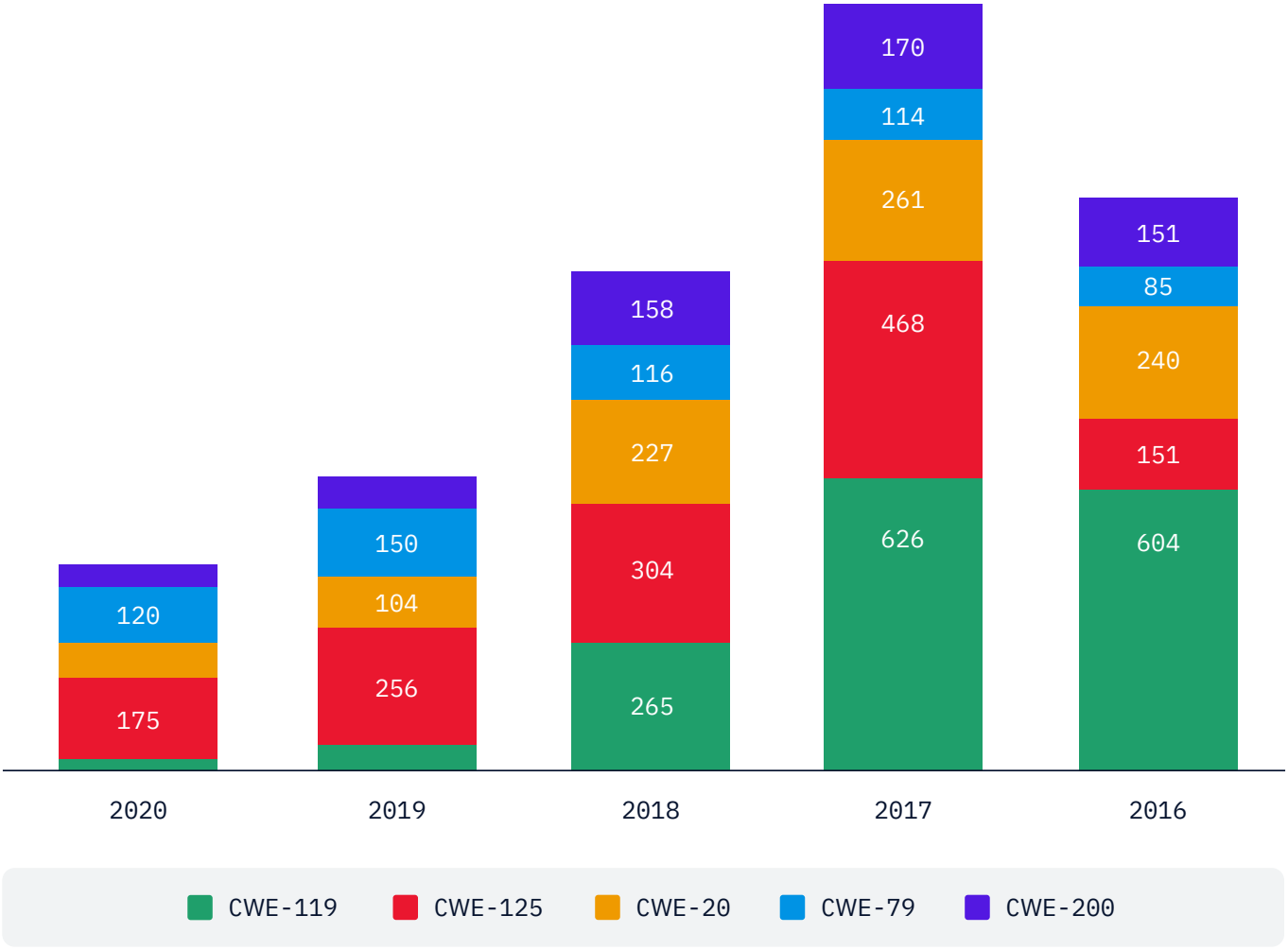
Alpine: Trends Over the Years



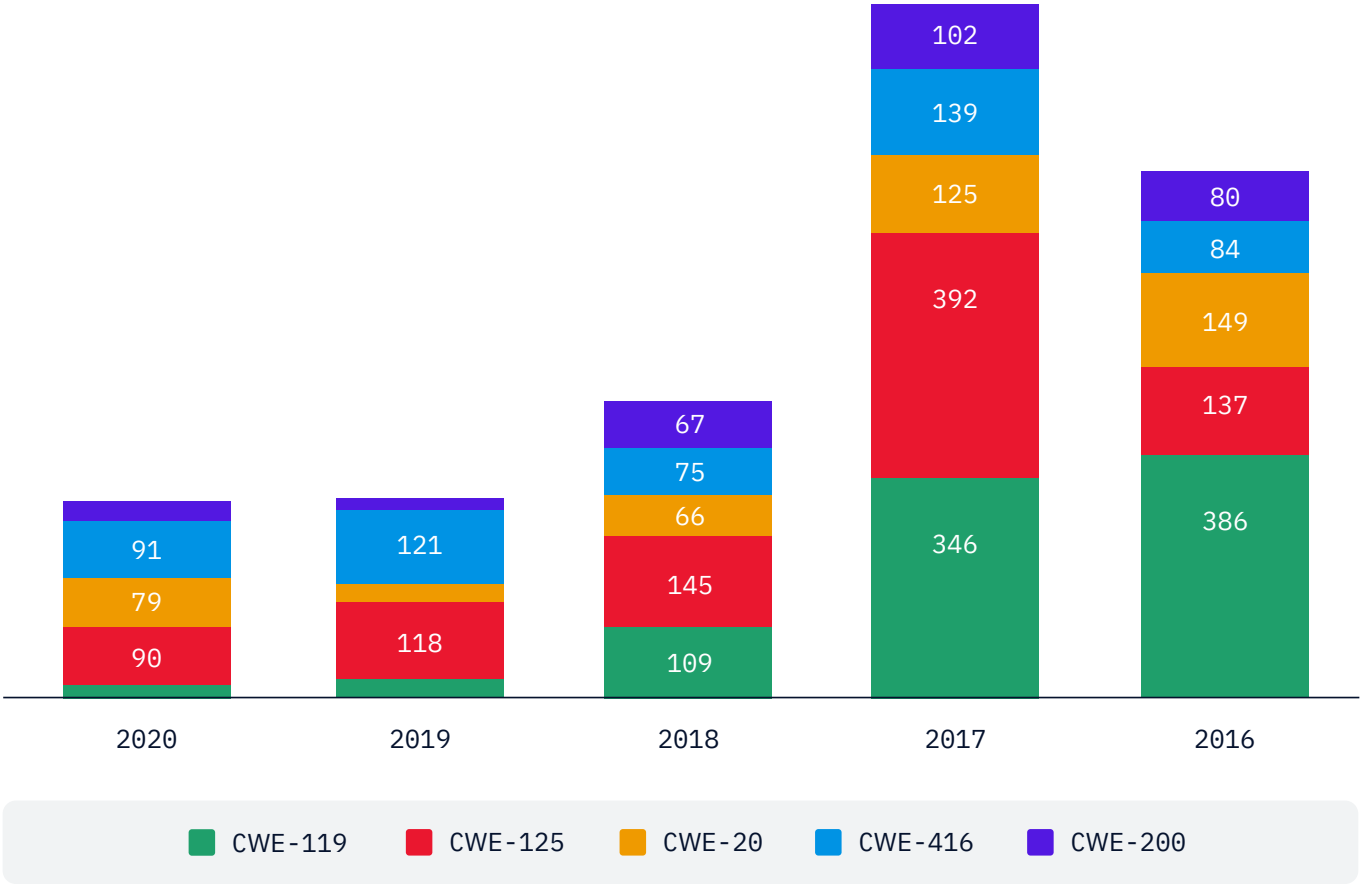
Ubuntu: Trends Over the Years



Debian: Trends Over the Years



Red Hat: Trends Over the Years



Clearly, one of the larger storylines from this data is the drop or change in the counts of the top vulnerabilities in 2019 and 2020. Also, in 2019 and 2020, CWE-125, CWE-787, and CWE-416 were the most commonly found vulnerabilities. One possible reason could be that as containers get used in newer workloads and newer applications, the types of vulnerabilities found might also fluctuate.

Conclusion

Companies across the globe have embraced containerization for a host of reasons, including increased efficiency, cost-effectiveness, and reduced overhead. But as the data in this report illustrates, it's important to take precautions to reduce the risks posed by vulnerable container images.

One important best practice is to scan your container images for potential vulnerabilities. It's generally wise to do this earlier than later, so consider scanning your images before you even package them.

Another benefit of incorporating container scanning into your regular workflows is that scanning tools (like FOSSA) not only detect vulnerabilities, but also inventory the components in your application. This offers visibility into potential software supply chain threats.

About FOSSA

Up to 90% of any piece of software is from open source, creating countless dependencies and areas of risk to manage. FOSSA is the most reliable automated policy engine for security management, license compliance, and code quality across the open source stack. With FOSSA, engineering, security, and legal teams all get complete and continuous risk mitigation for the entire software supply chain, integrated into each of their existing workflows. FOSSA enables organizations like Uber, Zendesk, Twitter, Verizon, Fitbit, and UiPath to manage their open source at scale and drive continuous innovation. Learn more at https://fossa.com.