SPOTLIGHT

# The Dark Reality of Open Source

## Through the Lens of Threat and Vulnerability Management

RISKSENSE®

# Executive Summary

Open source software (OSS) has quickly transformed both how modern applications are built and the underlying code they rely on. Access to high-quality and powerful open source software projects has allowed developers to quickly integrate new capabilities into their applications without having to reinvent the wheel. As a result, it is now estimated that between 80% and 90% of the code in most modern applications is made up of open source components. Likewise, many of the very tools that have enabled the growth of DevOps and CI/CD such as Jenkins, Kubernetes, and Docker are themselves open source projects.

OSS also allows organizations to reduce their software costs, and is often key to digital transformation efforts and the transition of services to the cloud. It is no surprise then that a 2020 report from Red Hat found that 95% of organizations view open source software as strategically important to their business.

However the open source revolution also comes with its own potential pitfalls. Open source code is often considered to be more secure than private code, as OSS can be reviewed by many more people to find problems. However, some weaknesses will always make it through the review process, and when countless developers reuse the same code, any vulnerabilities in that code can likewise be replicated. In short, while open source projects can rapidly go viral, so can their vulnerabilities. As a result, the same Red Hat report also found that the security of the code was the #1 barrier to adopting open source in the enterprise.

The now infamous Heartbleed bug demonstrated how a relatively simple vulnerability in the OpenSSL library, could reach around the globe and put roughly 17.5% of the world's SSL-enabled sites at risk. And while Heartbleed showed the breadth of open source weaknesses, the Equifax breach showed the severity, when a vulnerability in the open source Apache Struts framework led to one of the biggest data breaches in U.S. history.

And while Heartbleed and the Apache Struts vulnerabilities are the household names of open source vulnerabilities, they are far from the only examples. Open source software is increasingly being targeted by cryptominers, ransomware, and leveraged in DDoS attacks. Unfortunately, OSS vulnerabilities are often a blind spot for many enterprises, who may not always be aware of all the open source projects and dependencies that are used in their applications.

With this in mind, we have focused this version of the RiskSense Spotlight report on vulnerabilities in some of today's most popular open source software, including more than 50 OSS projects and over 2,600 vulnerabilities. We then used this dataset to provide a risk-based analysis of open source software to reveal the following:

- **Which vulnerabilities pose the most risk based on their susceptibility to real-world attack**
- **Which open source projects have the most vulnerabilities and risk**
- **What are the most significant vulnerabilities for individual open source projects**
- **How open source vulnerabilities are growing year over year**
- **Gaps and lags in how OSS vulnerabilities are added to the U.S. National Vulnerability Database**
- **What underlying weaknesses caused the vulnerability and how attackers could use them**
- **How the vulnerabilities were scored and categorized by CVSS and other models**

# Key Findings

### 2019 was a Record Year for OSS Vulnerabilities

In 2019 the overall number of published open source CVEs (968) more than doubled compared to any previous year. Vulnerabilities grew by 130% between 2018 and 2019 (421 CVEs to 968 CVEs), and was 127% higher than 2017 (435), which had the second most CVEs in the study. This increase does not appear to be a flash in the pan as the discovery of new CVEs also remains at historically high levels through the first three months of 2020. This volume increases the complexity of managing an organization's attack surface for developers, IT, and security teams alike.

### Widespread Problems in NVD Disclosure Latency

Vulnerabilities in open source software are taking a very long time to be added to the U.S. National Vulnerability Database (NVD). The average time between the first public disclosure of a vulnerability and its addition to the NVD was 54 days. The longest observed lag was 1,817 days for a critical PostgreSQL vulnerability. 119 CVEs had lags of more than 1 year, and almost a quarter (24%) had lags of more than a month. These lags were consistent across all severities of vulnerabilities, with critical severity vulnerabilities having some of the longest average lag times. This latency creates a dangerous lack of visibility for organizations who rely on the NVD as their main source of CVE data and context information.

### Jenkins & MySQL Generate the Most Vulnerabilities

The Jenkins automation server had the most CVEs overall with 646 and was closely followed by MySQL with 624. These projects likewise tied for the most weaponized vulnerabilities with 15 (vulnerabilities for which exploit code exists). By contrast, HashiCorp's Vagrant only had 9 total CVEs, but 6 of them were weaponized, making it one of the most weaponized open source projects in terms of percentage. Apache Tomcat, Magento, Kubernetes, Elasticsearch, and JBoss all had vulnerabilities that were trending or popular in real-world attacks.

### Cross-Site Scripting & Input Validation are the Most Weaponized Weaknesses

Cross-Site Scripting (XSS) and Input Validation weaknesses were both some of the most common and most weaponized types of weaknesses in the study. XSS issues were the second most common type of weakness, but were the number one most weaponized. Likewise Input Validation issues were the third most common and second most weaponized. Input Validation and Access Control issues were both common and were seen trending in real-world attacks.

### Rare Weaknesses Matter in the Real World

Some weaknesses were far less common, yet remained very popular in active attack campaigns. Deserialization Issue (28 CVEs), Code Injection (16 CVEs), Error Handling Issues (2 CVEs), and Container Errors (1 CVE) were all seen trending in the wild. The fact that these issues are rare in open source projects is a positive sign for the security of open source code, but also serve as a reminder that when problems do pop in OSS, they can be attacked quite broadly.

# Table of Contents

# 1. Overview of Vulnerabilities

Open source software has existed for decades reaching back to the earliest days of the GNU Project and the subsequent release of Linux. And while Linux has become one of the most widely used operating systems today, it was historically one of the few success stories of open source. However, over the past several years, OSS has experienced somewhat of a revolution with much of the most popular software being based on open source.

We focused our analysis on some of the most popular open source projects from the past five years. We used a variety of factors to build the list including popularity on GitHub, market value of companies based on specific open source projects (e.g., Elastic and Elasticsearch), as well as various OSS software lists such as the BOSS index. In total, the resulting dataset included 54 open source projects. We analyzed each project over the past five years from 2015 through the first three months of 2020, which yielded a total of 2,694 CVEs.

Of note, we intentionally excluded Linux and its many offshoots. While Linux vulnerabilities are obviously significant, they have been well documented in other analysis, and our goal was to focus on more recent, smaller projects. We also excluded open source web frameworks such as Drupal and WordPress, which we analyzed in the recent Spotlight report, "Cracks in the Foundation: Web and Application Framework Vulnerabilities."

## Increased Risk Funnel Analysis

While all vulnerabilities matter, those used by attackers in the real world naturally pose a much more immediate risk to an organization. To this end, we analyzed the dataset in terms of a variety of real-world threat contexts. This type of threat-centric analysis provides a powerful way to quickly hone in on the most important vulnerabilities for remediation. The analysis yielded a progressively focused funnel of vulnerabilities based on the following contexts:

- **Weaponized vulnerabilities:** Of the 2,694 total vulnerabilities, only 89 or 3.3% of them are weaponized, meaning that known exploit code exists that can take advantage of the vulnerability.
- **Strategic Vulnerabilities:** 18 vulnerabilities were found to enable remote code execution (RCE) or privilege escalation (PE). Such vulnerabilities are highly valuable to attackers and increase the likelihood and impact of an attack.
- **Trending in the Wild:** 6 vulnerabilities were noted by RiskSense research to be Trending based on their being targeted in active attack campaigns or having the potential for widespread impact.

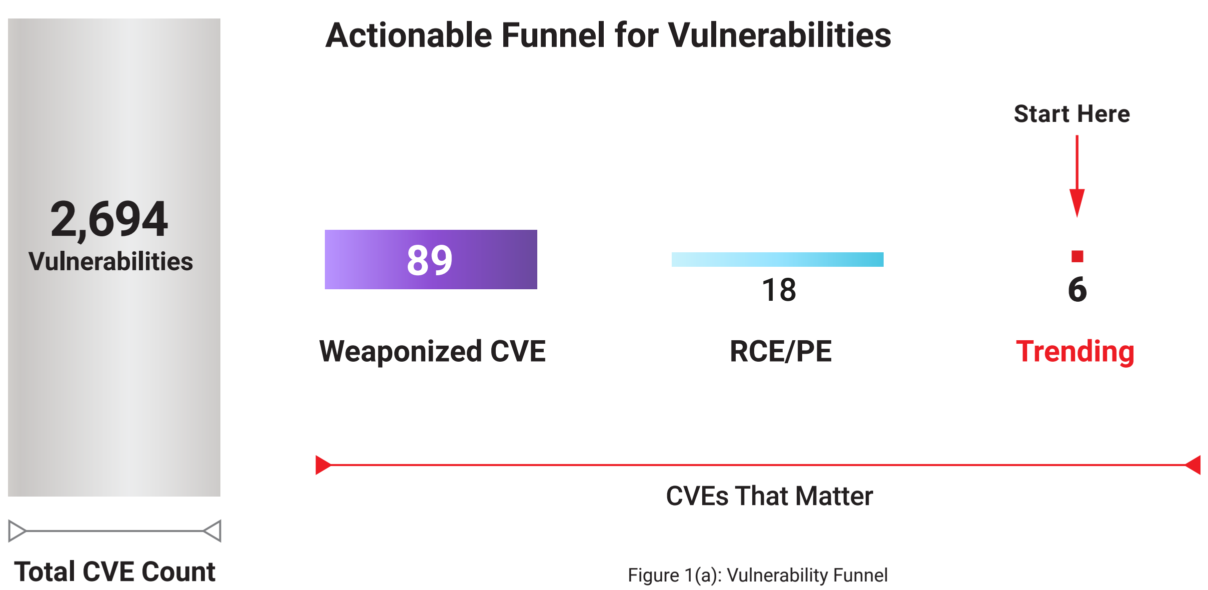


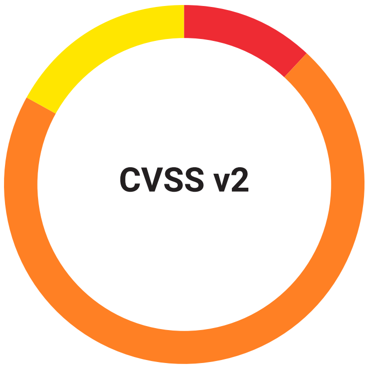

Figure 1(a): Vulnerability Funnel

# 1. Overview of Vulnerabilities (Continued)

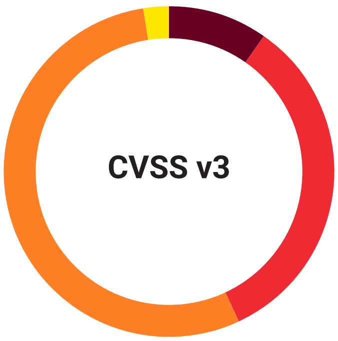
## CVEs by Severity

We also analyzed the data set in terms of CVSS v2, CVSS v3, and RiskSense's Vulnerability Risk Rating (VRR). It should be noted that not all CVEs had a CVSS v3 score as that scoring model was not introduced until later in 2015. As a result the CVSS analysis only contains 2,474 CVEs as opposed to 2,694.

Figure 1(b) shows that the majority of CVEs were rated as Medium in the CVSS v2 scoring model. In all, 11.9% CVEs were High, 71.1% were rated Medium, and 17% were Low. CVSS v3 introduces the "Critical" severity rating along with a more sophisticated scoring model. In terms of the OSS dataset, this translated to 9.6% classified as Critical CVEs, 33.3% as High, 54.5% as Medium, and 2.6% Low.




| Rating | CVEs | Weaponized | RCE/PE | Trending |
|---|---|---|---|---|
| High | 320 | 34 | 10 | 4 |
| Medium | 1917 | 46 | 8 | 2 |
| Low | 457 | 9 | 0 | 0 |
| Total | 2,694 | 89 | 18 | 6 |




| Rating | CVEs | Weaponized | RCE/PE | Trending |
|---|---|---|---|---|
| Critical | 237 | 16 | 4 | 3 |
| High | 824 | 39 | 10 | 2 |
| Medium | 1349 | 15 | 0 | 0 |
| Low | 64 | 0 | 0 | 0 |
| Total | 2474 | 70 | 14 | 5 |


| Rating | CVEs | Weaponized | RCE/PE | Trending |
|---|---|---|---|---|
| Critical | 18 | 18 | 13 | 3 |
| High | 43 | 43 | 5 | 3 |
| Medium | 2207 | 23 | 0 | 0 |
| Low | 426 | 5 | 0 | 0 |
| Total | 2,694 | 89 | 18 | 6 |

Figure 1(b): Comparison of CVE Severity Classification Models

# 1. Overview of Vulnerabilities (Continued)

**A Note About RiskSense VRR:** The RiskSense VRR has inherent advantages over CVSS in that it takes into account a variety of more advanced vulnerability analytics as well as real-world contexts such as the popularity of a vulnerability in attack campaigns or hacker forums. However, the OSS data demonstrates the value of this approach in terms of efficiency. The VRR model produced only 18 Critical CVEs and 43 High severity CVEs. However, these combined 61 CVEs (2.3%) accounted for all of the RCE/PE capable and trending vulnerabilities in the dataset.

This means Security or IT teams could address the highest risk vulnerabilities by addressing only 2.3% of the CVEs in the dataset.

By contrast, teams would need patch 2,237 (83%) of the CVEs to achieve the same efficacy using CVSS v2 or 1,098 (42.9%) using CVSS v3. It is also important to note that CVSS v3 scores did not cover all RCE/PE and trending vulnerabilities simply due to not being applied to all CVEs in 2015.

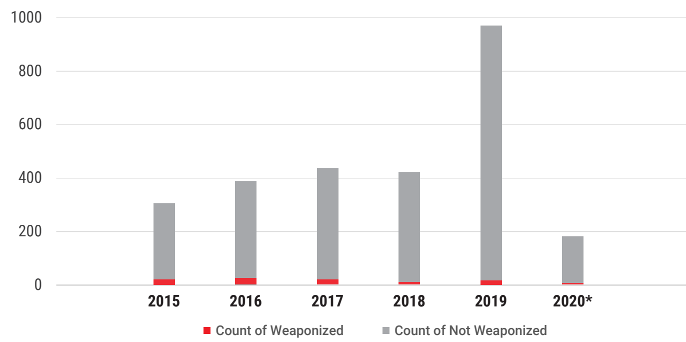
# 2. Vulnerabilities by Year

Based on the sheer volume of CVEs, 2019 was a watershed year for open source vulnerabilities. While the total number of CVEs has been steadily rising in recent years, 2019 more than doubled the previous high-water mark set in 2017. 2019 showed a 130% increase compared to 2018, rising from 421 CVEs to 968. This rate was far higher than the overall NVD, which grew at a rate of only 8% in the same time frame.

Through the first three months of 2020, the rate of CVEs discovery remains historically high with a total of 178 CVEs. While this puts 2020 on a slower pace than 2019, it is well above all other years.

The increase in vulnerabilities was seen across a variety of projects, although Magento, GitLab, and Jenkins showed the most pronounced increases. Magento, an open source e-commerce platform, had no vulnerabilities in 2018 but had 137 in 2019. GitLab vulnerabilities rose by almost 400% between 2018 and 2019, jumping from 40 to 198 CVEs, while Jenkins rose by 174%, from 120 to 329 CVEs. Conversely, Hive, Puppet, and OpenShift showed decreases in CVEs from 2018 to 2019.

As a point of good news, the weaponization of vulnerabilities in 2019 remains low. Of the 978 vulnerabilities in 2019, only 15 or 1.5% were weaponized. This is notably lower than the average NVD weaponization rate for 2019, which was 4.1%.




| Year | Total CVEs | Weaponized |
|---|---|---|
| 2015 | 303 | 19 |
| 2016 | 388 | 23 |
| 2017 | 435 | 18 |
| 2018 | 421 | 9 |
| 2019 | 968 | 15 |
| 2020* | 179 | 5 |
| Total | 2694 | 89 |

Figure 2(a): Open Source CVEs by Year
*2020 data only includes January through March

# 3. Vulnerabilities by Open Source Project

Next we analyzed the vulnerabilities in terms of the open source project that they were associated with. It should be noted that some vulnerabilities applied to more than one project.

It is important to note that simply having a large number of vulnerabilities should not necessarily be a negative mark against a particular OSS project. For example, Jenkins, which had the most vulnerabilities of any OSS project, is also a CVE Numbering Authority (CNA) and has a robust process for the collection and submission of CVEs. As a result, such projects may be more efficient at discovering and reporting vulnerabilities than others. This makes it important to track additional real-world contexts such as vulnerabilities that are weaponized or popular (trending) in the wild.

Figure 3(a) shows the top 30 OSS projects both in terms of total CVEs as well as weaponized CVEs. The list is sorted in terms of decreasing number of weaponized CVEs.

The top projects in terms of total vulnerabilities were Jenkins, MySQL, GitLab, OpenStack, and Magento. The total CVEs for these projects are shown below in Figure 3(b).

| Product | Total CVEs | Weaponized |
|---|---|---|
| Jenkins | 646 | 15 |
| MySQL | 624 | 15 |
| JBoss | 88 | 8 |
| OpenStack | 165 | 7 |
| Tomcat | 72 | 7 |
| Hive | 90 | 6 |
| Vagrant | 9 | 6 |
| Elasticsearch | 58 | 4 |
| Ansible | 32 | 4 |
| Magento | 154 | 3 |
| Alfresco | 9 | 3 |
| GitLab | 306 | 2 |
| OpenShift | 76 | 2 |
| PostgreSQL | 47 | 2 |
| Docker | 30 | 2 |
| Redis | 16 | 2 |
| Chef | 10 | 2 |
| Kubernetes | 44 | 1 |
| Nginx | 22 | 1 |
| Spark | 16 | 1 |
| LifeRay Portal | 10 | 1 |
| Odoo | 10 | 1 |
| Kaltura | 5 | 1 |
| SVN | 5 | 1 |
| Artifactory | 4 | 1 |
| Puppet | 72 | 0 |
| Cloud Foundry | 42 | 0 |
| Kibana | 29 | 0 |
| MongoDB | 18 | 0 |
| Hbase | 12 | 0 |

Figure 3(a): CVEs by Project, Highest to Lowest Weaponized

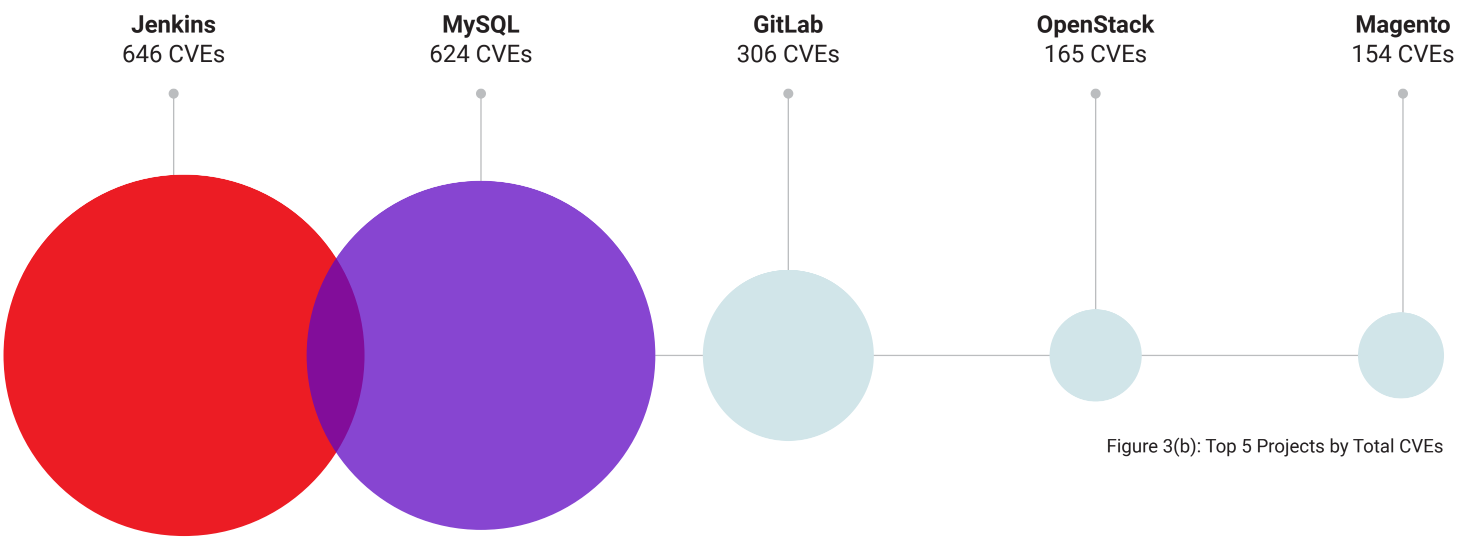



Figure 3(b): Top 5 Projects by Total CVEs

# 3. Vulnerabilities by Open Source Project (Continued)

However, large numbers of CVEs don't necessarily translate to equally large amounts of weaponized vulnerabilities. While Jenkins and MySQL were tied with the most weaponized CVEs with 15, their overall weaponization rate remains relatively low at around 2.3% and 2.4% respectively. In strong contrast, HashiCorp's Vagrant only had 9 CVEs, yet 6 of them (66.7%) were weaponized. While this is admittedly a very small sample size, it is worth noting and monitoring going forward, as organizations should be aware of their software that is the most likely to be weaponized. Figure 3(c) provides a list of projects in terms of their weaponization rates. Many examples, such as Artifactory, have a very low number of CVEs overall (4), which contribute to their high weaponization rates.

However, Apache Tomcat, JBoss, Elasticsearch, and Hive all have comparatively large numbers of CVEs and have weaponization rates significantly higher than the average of the dataset, which is 3.4%.

In contrast, several open source projects and very low or non-existent weaponization rates. GitLab had the 3rd most total CVEs, yet only two were weaponized. Puppet and Cloud Foundry had 72 and 43 CVEs respectively, yet had no weaponization.

| Product | Total CVEs | Weaponized | % Weaponized |
|---|---|---|---|
| Vagrant | 9 | 6 | 66.7% |
| Alfresco | 9 | 3 | 33.3% |
| Artifactory | 4 | 1 | 25% |
| Chef | 10 | 2 | 20% |
| Kaltura | 5 | 1 | 20% |
| SVN | 5 | 1 | 20% |
| Ansible | 32 | 4 | 12.5% |
| Redis | 16 | 2 | 12.5% |
| LifeRay Portal | 10 | 1 | 10% |
| Odoo | 10 | 1 | 10% |
| Tomcat | 72 | 7 | 9.7% |
| JBoss | 88 | 8 | 9.1% |
| Elasticsearch | 68 | 4 | 6.9% |
| Hive | 90 | 6 | 6.7% |
| Docker | 30 | 2 | 6.7% |

Figure 3(c): Projects by Percent of Weaponized CVEs

Organizations should be aware of software

most likely to be weaponized

# 3. Vulnerabilities by Open Source Project (Continued)

## Summary of Key Projects and Vulnerabilities

The following section highlights 12 important vulnerabilities that organizations should be aware of in their risk management practices. These include vulnerabilities that are trending in active threat campaigns and/or pose a high risk due to their exploitability and potential impact.

- **Jenkins:** Jenkins is an automation server which enables developers to build, test, and deploy software. As described above, Jenkins shows the most total CVEs and the third most weaponized CVEs. While not currently being exploited in the wild, the recently discovered CVE-2020-2100 is particularly significant in that it can be used in reflective DDoS attacks.

- **JBoss:** JBoss is an open source application platform. JBoss had the 8th most CVEs with 88, and the 6th most weaponized CVEs with 8. JBoss is susceptible to the recent Ghostcat vulnerability (CVE-2020-1938), which primarily affects Apache Tomcat and is currently trending in the wild. Other notable JBoss vulnerabilities include CVE-2017-12149, a deserialization vulnerability that has been attacked in the wild, as well as the older CVE-2010-0738 which continues to be targeted by ransomware. Refer to our previous Spotlight report, "Enterprise Ransomware – Through the Lens of Threat and Vulnerability Management" for more analysis on JBoss and other vulnerabilities that are targeted by ransomware.

- **Apache Tomcat:** Apache Tomcat had the 10th most CVEs and was the 7th most weaponized. The most notable vulnerability was the aforementioned Ghostcat vulnerability (CVE-2020-1938), which allows attackers to plant backdoors on Tomcat servers. Other notable vulnerabilities include CVE-2019-0232 and CVE-2017-12617, which enable remote code execution and are rated as Critical by RiskSense's Vulnerability Risk Rating.

- **Magento:** Magento is a popular open source e-commerce platform. While only 3 out of Magento's 154 vulnerabilities were weaponized, they were all significant. CVE-2019-7932 enables injection of arbitrary code and is currently trending in the wild. Additionally CVE-2019-7139 enables SQL injection and has likewise been previously attacked in the wild. Likewise CVE-2016-4010 has been used in "magecart" attacks as a way to skim credit card information from Magento-based sites.

- **Docker:** Docker accounted for 30 CVEs and only had two that were weaponized. However, this included CVE-2019-5736, which can allow a malicious Docker container to gain root-level control over the host by attacking the host's runC binary. This same vulnerability also affects Kubernetes and OpenShift. Additionally, improperly secured Docker deployments have been found to be attacked by a worm known as "Graboid."

- **Kubernetes:** In addition to being affected by the runC vulnerability described above (CVE-2019-5736), Kubernetes was also affected by CVE-2018-1002105. This vulnerability is also trending in the wild and allows an attacker to send fully authorized requests to a Kubernetes API server with the privileges of any user.

- **Elasticsearch:** Elasticsearch had the 12th most total vulnerabilities with 58 CVEs. Four vulnerabilities were weaponized with two enabling remote code execution. Notably, CVE-2015-1427 has been used in a variety of attacks over the past year including cryptomining, botnet, and DDoS campaigns.

- **Git:** Git came up very little in our dataset with only two CVEs, but that doesn't mean that it hasn't been the target of attacks. Instead of going after CVEs, attackers have targeted weakly secured GitHub, GitLab, and BitBucket accounts to gain access and hold Git repositories for ransom.

# 4. NVD Latency Analysis

The U.S. National Vulnerability Database (NVD) is a critical resource of vulnerability information, and for many organizations is the primary way of consuming CVE information. The NVD ingests CVE entries from MITRE, vendors, and trusted security researchers and enhances each CVE entry with a variety of information including severity scores, weakness information and a variety of additional contexts. However, not all CVEs disclosed by vendors get into the NVD in a timely manner leading to CVE disclosure latency with respect to the NVD release date. This lag can cause serious problems for organizations that rely solely on the NVD for their vulnerability information.

**Unfortunately, the NVD lag observed in the OSS dataset was exceptionally high, with the average lag being 54 days.**


This should be a concerning finding for organizations that rely on the NVD as their primary source of CVE data. It is not unusual to see some amount of lag between a CVE first being published and being added to the NVD as the NVD will validate and enrich a CVE with a variety of information. However, many CVEs are added to the NVD on the same day they are published, and for most popular software, lags are typically no more than 1 to 2 days.

Digging deeper into the data, it became clear that lags were both fairly common and could be particularly long. The longest observed lag was 1,817 days. This was tied to a PostgreSQL vulnerability (CVE-2015-0244), which enabled SQL injection and is a CVSS v3 Critical vulnerability with a score of 9.8. The vulnerability was originally published on 2/6/15 and was not added to the NVD until 1/27/20.

And while the Postgres vulnerability is the most egregious example, there were many additional causes for concern. In the overall dataset of 2,694 CVEs:

- **119 CVEs (4.4%) had lags of a year or more**
- **660 CVEs (24%) had lags of 30 days or more**
- **1,286 CVEs (48%) had lags of 3 days or more**

Unfortunately the picture remains bleak even as we narrow our focus to the weaponized vulnerabilities. The only good news is that weaponized vulnerabilities had a lower percentage of CVEs with a lag of a year or more. For the 89 weaponized CVEs we found:

- **The average lag time was 46 days**
- **2 CVEs (2.2%) had lags of a year or more**
- **26 CVEs (27%) had lags of 30 days or more**
- **66 CVEs (74%) had lags of 3 days or more**

# 4. NVD Latency Analysis (Continued)

Likewise, the lag metrics did not seem to improve based on CVSS score or severity. We analyzed the dataset based on CVSS v2, CVSS v3, and RiskSense's own Vulnerability Risk Rating as seen in Figure 4(a). **Across all scoring systems, the highest severities also had the longest lag times.** Note that not all CVEs were assigned CVSS v3 scores, which is why the totals differ from CVSS v2 and RiskSense.

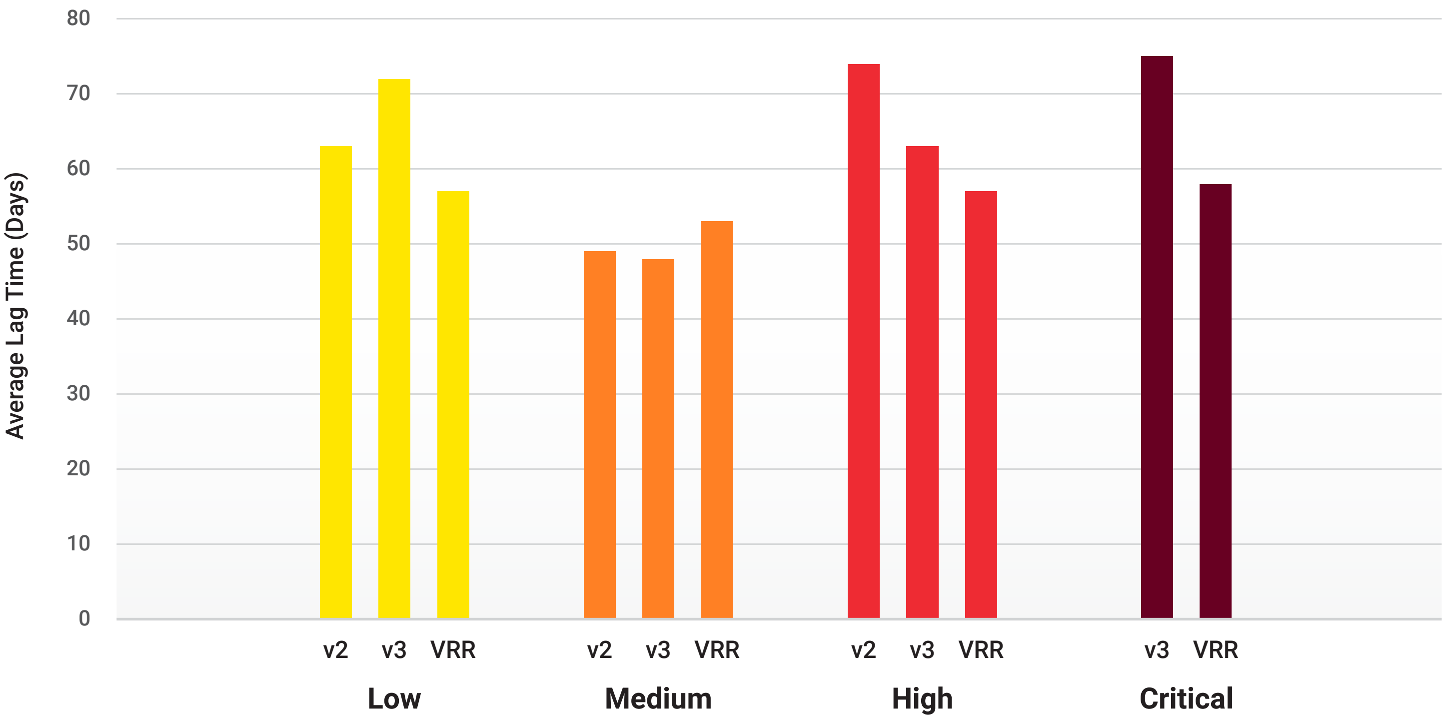



Figure 4(a): NVD Lag by CVE Severity

# 4. NVD Latency Analysis (Continued)

## NVD Lag Times by Open Source Project

While lag metrics were high overall, there was also a wide variance between the individual OSS projects. Figure 4(b) shows the average NVD lag times of all projects in the study ranked from highest average lag to lowest.

PostgreSQL had the longest average lag of 246 days across 47 CVEs. The Postgres numbers were heavily skewed by 6 CVEs with lag times longer than 1,000 days. Those CVEs include CVE-2015-0241, CVE-2015-0242, CVE-2015-0243, CVE-2015-0244, CVE-2015-3167, and CVE-2015-3166. Notably, CVE-2015-0244 and CVE-2015-3166 had CVSS v3 severities of Critical, while the others were classified as High. Some Postgres vulnerabilities had low lags, and some even were published on the same day the CVE was released. However, lags were a consistent issue with 23 out of the 47 total CVEs having a lag of 10 days or more.

Unfortunately, many of the projects with the most CVEs and most weaponized CVEs also had long lag times. For example Jenkins, which had the most total CVEs (646) and tied for the most weaponized (15), and an average lag time of 33 days. However, others fared considerably worse. JBoss averaged 154 days of lag across 88 CVEs and 8 weaponized CVEs. Likewise OpenStack averaged 130 days across 165 CVEs and 7 weaponized.

On the other hand, some projects fared much better. Vagrant, which had 6 of its 9 CVEs weaponized, had a lag time of 2 days. Cloud Foundry fared the best with less than a day of lag across 42 vulnerabilities.

| Product | Average NVD Lag (Days) |
|---|---|
| PostgreSQL | 246 |
| MongoDB | 240 |
| Logstash | 173 |
| JBoss | 154 |
| Ansible | 151 |
| OpenStack | 130 |
| Puppet | 126 |
| OpenShift | 120 |
| Tomcat | 114 |
| Elasticsearch | 98 |
| Kaltura | 77 |
| GitLab | 76 |
| Kibana | 70 |
| Kubernetes | 66 |
| Hive | 58 |
| Selenium | 57 |
| Redis | 52 |
| Mapbox | 48 |
| Gradle | 38 |
| Jenkins | 33 |
| Hadoop | 30 |
| Docker | 27 |
| Chef | 27 |
| Open vSwitch | 26 |
| Hbase | 20 |
| MySQL | 19 |
| Git | 15 |
| Spark | 12 |
| NPM | 12 |
| SVN | 10 |
| Magento | 10 |
| Nginx | 7 |
| Canvas | 4 |
| Cassandra | 3 |
| Vagrant | 2 |
| Vault | 2 |
| Artifactory | 1 |
| Consul | 1 |
| LifeRay Portal | 1 |
| Snort | 1 |
| Odoo | 0 |
| Kafka | 0 |
| Cloud Foundry | 0 |
| Alfresco | 0 |
| Appium | 0 |
| Coffescript | 0 |
| Heroku | 0 |
| Intellij | 0 |
| Maven | 0 |
| Nomad | 0 |
| Pentaho | 0 |
| Sentinel | 0 |
| Tensorflow | 0 |

Figure 4(b): Average NVD Lag

# 4. NVD Latency Analysis (Continued)

## Lag Examples

The overarching problem with NVD lag times is that it can make organizations blind to significant risks, especially if they rely solely on the NVD as the source of truth for CVE data. The following examples from our OSS dataset highlight this impact to an organization.

CVE Release Date · **NVD** NVD Release Date · Weaponized · Patch Release Date

### CVE-2019-1003000:

a sandbox bypass vulnerability that allows attackers to execute arbitrary code in the Jenkins JVM.

| CVSS v2 | CVSS v3 | RiskSense VRR | OSS Project |
|---|---|---|---|
| 6.5 | 8.8 | 9.84 | Jenkins |


**NVD Lag:** 15 days
**Time to Weaponize (TTW):** -169 days
**Time to Develop Patch (TTP):** 0 days

### CVE-2016-9587:

an input validation vulnerability that can enable remote code execution by an attacker. The vulnerability is assigned to Ansible but also affects OpenStack.

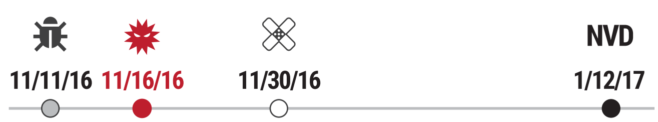
| CVSS v2 | CVSS v3 | RiskSense VRR | OSS Project |
|---|---|---|---|
| 9.3 | 8.1 | 10 | Ansible<br>OpenStack |


**NVD Lag:** 471 days
**Time to Weaponize (TTW):** 0 days
**Time to Develop Patch (TTP):** 17 days

### CVE-2016-9299:

an LDAP Injection vulnerability that can enable remote code execution.

| CVSS v2 | CVSS v3 | RiskSense VRR | OSS Project |
|---|---|---|---|
| 7.5 | 9.8 | 10 | Jenkins |




**NVD Lag:** 63 days
**Time to Weaponize (TTW):** 5 days
**Time to Develop Patch (TTP):** 19 days

# 4. NVD Latency Analysis (Continued)

CVE Release Date    NVD Release Date    Weaponized    Patch Release Date

## CVE-2016-6816:

an input validation vulnerability that can enable remote code execution.

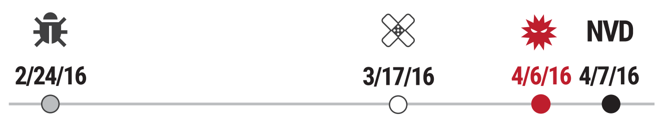
| CVSS v2 | CVSS v3 | RiskSense VRR | OSS Project |
|---|---|---|---|
| 6.8 | 7.1 | 8.68 | Apache Tomcat |


**NVD Lag:** 133 days
**Time to Weaponize (TTW):** 14 days
**Time to Develop Patch (TTP):** -6 days

## CVE-2016-0792:

is a deserialization vulnerability that can enable remote code execution.

| CVSS v2 | CVSS v3 | RiskSense VRR | OSS Project |
|---|---|---|---|
| 9 | 8.8 | 9.97 | Jenkins OpenShift |




**NVD Lag:** 44 days
**Time to Weaponize (TTW):** 43 days
**Time to Develop Patch (TTP):** 23 days

# 5. Vulnerabilities by Weakness

To better understand the vulnerabilities in the dataset, we analyzed them in terms of their underlying weaknesses using the Common Weakness Enumeration (CWE) classification. We then further highlighted the vulnerabilities that map to an OWASP Top 10 category. It should be noted that not all CVEs had corresponding CWE data, so the following analysis is limited to 2,209 CVEs.

Even with the reduction, the OSS dataset showed remarkable diversity, with a total of 106 CWE classifications represented. However only 32 of those CWE categories were weaponized. Figure 5(a) provides a ranked list of the most common CWEs that had at least one weaponized vulnerability.

From this list we can see that CWE-200 Information Exposure, CWE-79 Cross-Site Scripting (XSS), and CWE-20 Improper Input Validation were the most common weaknesses overall. It is no surprise to see high amounts of XSS, as they have consistently been some of the more common types of weaknesses and also some of the most handsomely rewarded by bug bounty programs. Likewise, CWE-20 Improper Input Validation is a common category as it covers a wide variety of potential attack patterns.

| CWE ID | CWE Name | Total CVEs | Weaponized |
|---|---|---|---|
| 200 | Information Exposure | 289 | 4 |
| 79 | Cross-Site Scripting | 236 | 11 |
| 20 | Improper Input Validation | 184 | 9 |
| 284 | Improper Access Control | 126 | 2 |
| 352 | Cross-Site Request Forgery (CSRF) | 104 | 4 |
| 264 | Improper Access Controls | 69 | 7 |
| 22 | Path Traversal | 61 | 2 |
| 119 | Improper Restriction within a Memory Buffer | 53 | 4 |
| 254 | Security Features | 47 | 6 |
| 732 | Incorrect Permission Assignment for Critical Resource | 44 | 1 |
| 287 | Improper Authentication | 37 | 1 |
| 918 | Server-Side Request Forgery (SSRF) | 36 | 1 |
| 502 | Deserialization of Untrusted Data | 33 | 5 |
| 310 | Cryptographic Issues | 30 | 1 |
| 276 | Incorrect Default Permissions | 27 | 1 |
| 89 | SQL Injection | 22 | 3 |
| 94 | Code Injection | 17 | 1 |
| 78 | OS Command Injection | 16 | 2 |
| 77 | Command Injection | 16 | 1 |
| 362 | Race Condition | 15 | 4 |
| 59 | Link Following | 15 | 2 |
| 74 | Injection | 14 | 1 |
| 434 | Unrestricted Upload of File with Dangerous Type | 11 | 3 |
| 190 | Integer Overflow or Wraparound | 11 | 1 |
| 835 | Infinite Loop | 9 | 1 |
| 798 | Use of Hard-coded Credentials | 6 | 1 |
| 426 | Untrusted Search Path | 5 | 1 |
| 427 | Uncontrolled Search Path Element | 2 | 2 |
| 388 | Error Handling | 2 | 1 |
| 704 | Incorrect Type Conversion or Cast | 2 | 1 |
| 90 | LDAP Injection | 1 | 1 |
| 216 | Containment Errors (Container Errors) | 1 | 1 |

Figure 5(a): Most Common CWEs

# 5. Vulnerabilities by Weakness (Continued)

However, we can start to see some interesting findings as we focus on the weaknesses that were both weaponized and trending in the wild. Figure 5(b) shows the top 10 CWEs in terms of weaponization. Most noticeably, Cross-Site Scripting takes over the top spot followed by Input Validation.

Other CWEs made noticeable jumps when ranked in terms of weaponization. For example, Deserialization was only the 13th most common CVE overall yet was 5th in terms of weaponization. Likewise Race Condition weaknesses were tied for the 6th most common weaponized weakness despite only having 15 total CVEs.

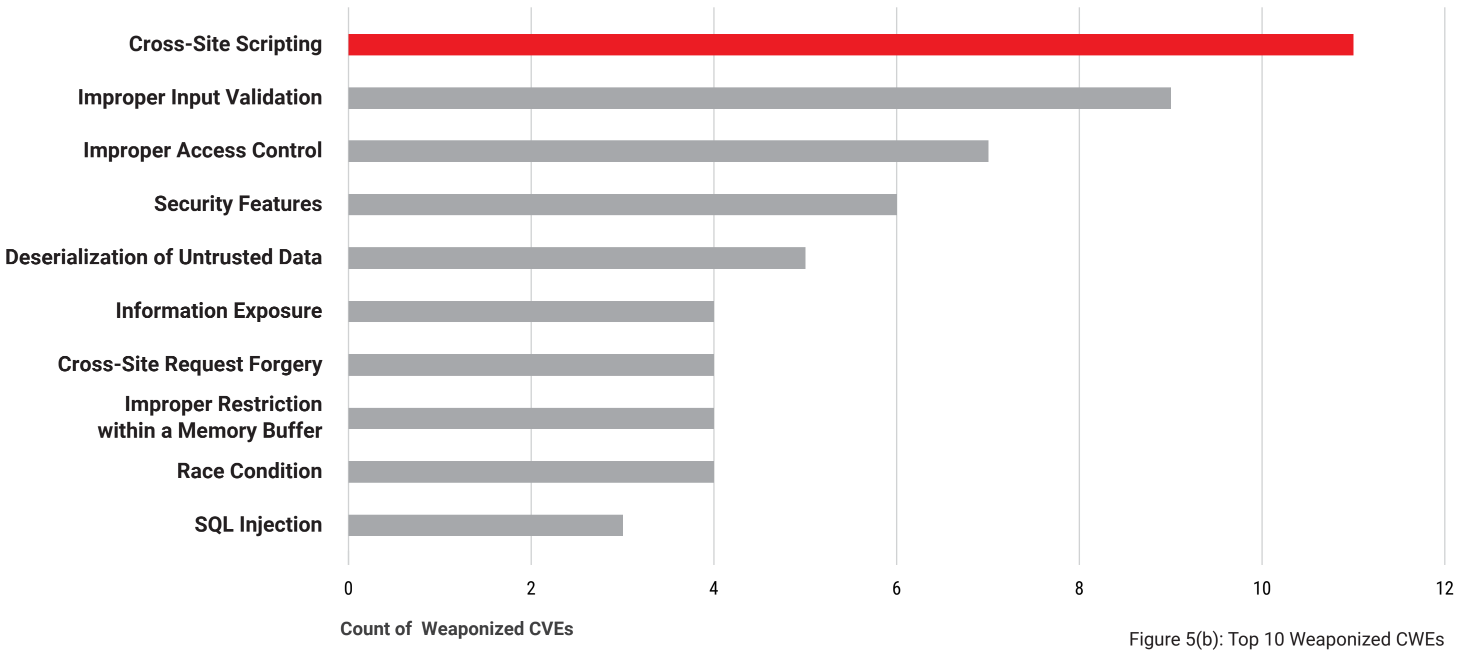



Figure 5(b): Top 10 Weaponized CWEs

Next, we further honed in the weaknesses that were specifically trending in the wild, shown in Figure 5(c). These vulnerabilities pose the most immediate risk to organizations due to their use in widespread attack campaigns. Most notably three of these weaknesses were not featured prominently in either of our previous lists above. Code Injection only had 17 CVEs total, with only 1 weaponized. Error Handling only mapped to 2 CVEs, one of which was both weaponized and trending, and the lone Container Error weakness in the dataset was likewise both weaponized and trending.

| CWE ID | CWE Name | CVE | Project |
|---|---|---|---|
| 20 | Improper Input Validation | CVE-2020-1938 | Apache Tomcat |
| 264 | Improper Access Control | CVE-2010-0738 | JBoss |
| 502 | Deserialization of Untrusted Data | CVE-2017-12149 | JBoss |
| 388 | Error Handling | CVE-2018-1002105 | Kubernetes, OpenShift |
| 216 | Containment Errors (Container Errors) | CVE-2019-5736 | Docker, OpenShift, Kubernetes |
| 94 | Code Injection | CVE-2019-7932 | Magento |

Figure 5(c): Weaknesses Trending in the Wild

# 5. Vulnerabilities by Weakness (Continued)

## Weaknesses by OWASP Top 10

Next we mapped CWE codes to their respective categories within the OWASP Top 10. In total 822 CVEs mapped to OWASP categories, of which 29 were weaponized and 2 were trending. A7: Cross-Site Scripting remained the most common weakness overall, followed by A5: Broken Access Control and A3: Sensitive Data Exposure.

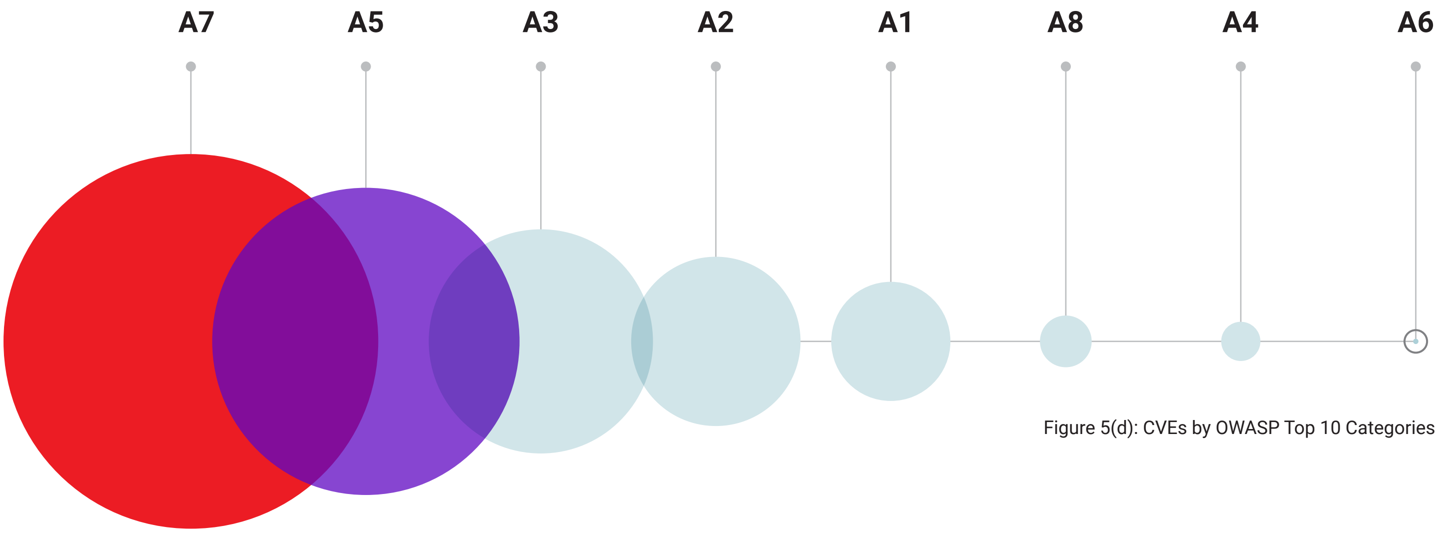



Figure 5(d): CVEs by OWASP Top 10 Categories

| OWASP Top 10 Category | Total CVEs | Weaponized | Trending |
|---|---|---|---|
| A1: 2017-Injection | 76 | 7 | 1 |
| A2: 2017-Broken Authentication | 108 | 1 | 0 |
| A3: 2017-Sensitive Data Exposure | 143 | 1 | 0 |
| A4: 2017-XML External Entities (XXE) | 25 | 0 | 0 |
| A5: 2017-Broken Access Control | 196 | 4 | 0 |
| A6: 2017-Security Misconfiguration | 2 | 0 | 0 |
| A7: 2017-Cross-Site Scripting (XSS) | 239 | 11 | 0 |
| A8: 2017-Insecure Deserialization | 33 | 5 | 1 |
| Total | 822 | 29 | 2 |

Figure 5(e): OWASP Top 10

# Conclusion

Open source software is an increasingly major part of an organization's attack surface. And while open source has many benefits, managing this new attack surface can be a particular challenge. As with more traditional software, open source projects are generating new vulnerabilities at a historically rapid pace, and traditional scoring systems (e.g., CVSS) on their own don't always do a good job in prioritizing which vulnerabilities carry the greatest real-world risk. Additionally many open source projects lag significantly behind more traditional software in terms of how CVEs are reported via standard resources like the National Vulnerability Database.

Unfortunately, these challenges including the number of overall vulnerabilities, weaponization rates, and NVD reporting lag times tend to vary considerably from project to project. This makes it all the more important to incorporate real-world vulnerability context into a risk-based approach to vulnerability management for open source software. We hope that the data in this report provides useful insights that organizations can put to use in their development, IT, and security practices. To learn more about the data in this report or about RiskSense products and services, please contact us at info@risksense.com.

# About RiskSense

RiskSense®, Inc. provides vulnerability management and prioritization to measure and control cybersecurity risk. The cloud-based RiskSense platform uses a foundation of risk-based scoring, analytics, and technology-accelerated pen testing to identify critical security weaknesses with corresponding remediation action plans, dramatically improving security and IT team efficiency and effectiveness. For more information, visit www.risksense.com or follow us on Twitter at @RiskSense.

RiskSense – the industry's only full spectrum risk-based vulnerability management and prioritization platform.

**Contact us today to learn more about RiskSense**

RiskSense, Inc. | +1 844.234.RISK | +1 505.217.9422 | risksense.com

CONTACT US | SCHEDULE A DEMO | READ OUR BLOG