

## RESEARCH REPORT

# Q3-2022 API ThreatStats™ Report

- Where are we most likely to be attacked?
- What is the most common attack vector?
- How long do we have to patch API vulnerabilities?



## Executive Summary

In Q3-2022, the Wallarm Research team culled 203 API-related vulnerabilities out of a total of 100,093 records examined. While we did not see the meteoric increase in API vulnerabilities seen last quarter, analysis did yield some key insights which could inform on your API vulnerability management program.

Initial analysis of this quarter's data show API vulnerabilities leveling off: the number of API vulnerabilities and impacted vendors – metrics that saw huge jumps in the [Q2 API Vulnerability Report](#) – were basically unchanged during Q3, along with a virtually unchanged CVSS scores (both average and % in the critical or high range). However, upon further investigation we unearthed these key findings:

1. **Injects.** While the OWASP Top-10 Injection categories ([A03:2021](#) for web apps and [API8:2019](#) for APIs) top the charts at over 33% of all CVEs analyzed, further inspection reveals many, many variations that undoubtedly will require extra effort to remediate.
2. **Infrastructure.** A vast majority of the most impactful vulnerabilities analyzed in Q3 impacted development tools and infrastructure – which clearly shifts your security focus.
3. **Exploits.** A surprising finding was that the average gap between CVE and exploit POC publication was zero days! This will greatly impact your mitigation timeline.

All these findings will have significant implications on your organization's API security program.

Our research so far this year shows that the volume of API vulnerabilities continues to increase, and the time-to-exploit continues to decrease. Read on to learn why you need to create data defensible remediation policies that engineers and executives will support – particularly as a vast majority of organizations' use of APIs skyrockets.

# Table Of Contents

Executive Summary .....	2
Introduction .....	4
Key Findings .....	5
Infrastructure .....	6
Injections .....	7
Exploit Timeline .....	8
Other Findings .....	8
CVSS Severity Ratings .....	8
OWASP Analysis .....	9
Most Dangerous CWEs Mapping .....	10
Known Exploits .....	10
Open-Source vs. Commercial .....	11
Vendors / Products .....	11
Triaging API Vulnerabilities .....	12
Vendor-Centric Approaches .....	12
EPSS Approach .....	13
Most Impactful API Vulnerabilities .....	15
What's Right for Your Organization? .....	17
Conclusion .....	17
Learn More / Add'l Resources .....	18
About Wallarm .....	18
Appendix .....	19
Methodology .....	19
EPSS Efficiency and Coverage .....	20
Analysis Notes .....	20

# Introduction

In their Security & Risk Planning Guide 2023, Forrester recommends increasing or defending investment in API security because: *API-first is the de facto modern development approach, and APIs help organizations create new business models and methods of engagement with customers and partners.*<sup>1</sup> They go on to state that security breaches due to unprotected APIs and API endpoints are common.

The question is, of course, is this true? Does an API-first approach open you up to more risk?

To address this, the Wallarm Research team continually collects and analyzes published API vulnerabilities and exploits – including new issued security reports, CVEs, bug bounty reports, vendor bulletins, exploits galleries, and researchers' tweets. The focus is on public or private APIs regardless of protocol, and does not include binary APIs such as Android, iOS, ChromeOS or BAPI.

In this third quarterly report, the team found 203 API-related vulnerabilities out of a total of 100,093 records examined. Despite the apparent leveling-off, our assessment of the data suggests three key findings which will have significant implications on your API security effort. We will examine these and other findings in this paper.

To get here, we dissect the data to look for trends and insights from a variety of perspectives, including software type, vendor, CVSS scores, CWEs and both OWASP Top-10 (2021) for web apps and OWASP API Security Top-10 (2019). We also look at publicly disclosed exploit POCs to determine where the risk lies. The full methodology is described at the end of this paper.

We hope that DevOps and AppSec teams at organizations with an API-first approach will use this both to assess your exposure and to reduce the risk in your API portfolio.

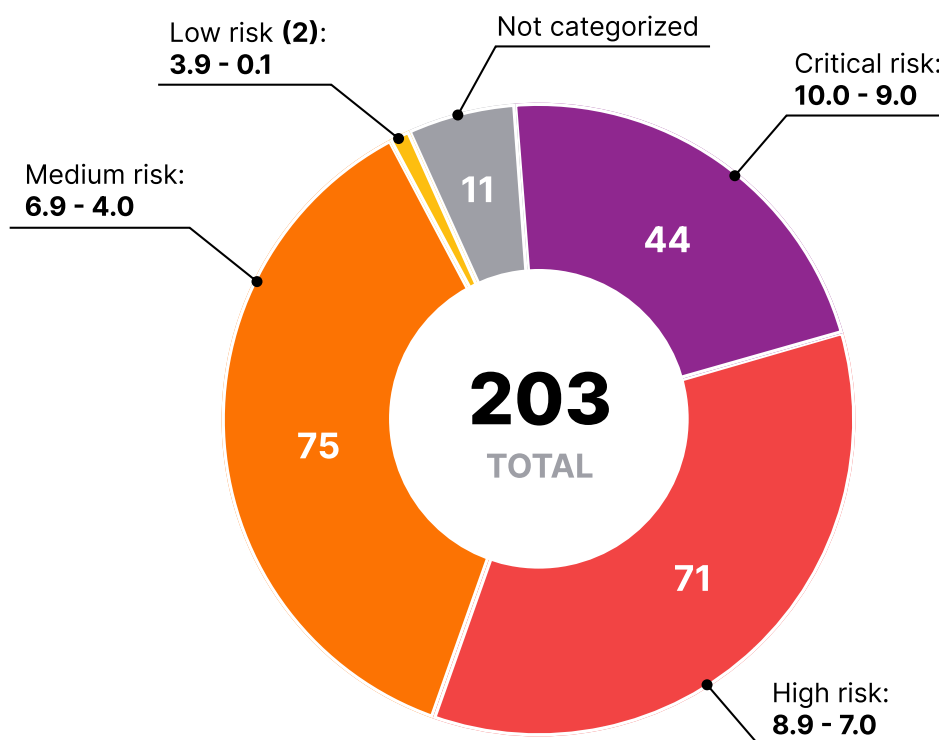
---

<sup>1</sup> See [Planning Guide 2023: Security & Risk](#) from Forrester.

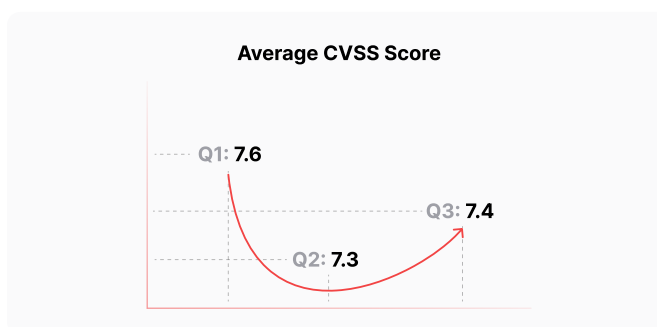


## Key Findings

Of the over 100,000 records analyzed, we found 203 API-specific vulnerabilities. This represents a 16% increase over Q2 in terms of vulnerabilities and vendors impacted. We view these results as basically unchanged, after the huge jump observed in the previous quarter.



The average CVSS score in Q3-2022 is 7.4 – compared to 7.3 in Q2. And 57% of all Q3 API vulnerabilities collected are rated Critical and High – unchanged from Q2.



All told, it looks like Q3 was not going to reveal anything terribly interesting or useful. However, a deeper analysis revealed **three key findings** that could have an immense impact on your organization's API security program.

## Infrastructure

The vulnerabilities analyzed in Q3 deemed the most impactful (see table below) are concentrated in **DevOps Tools** and **Infrastructure** categories, which should result in a shift of your organization's security focus – from externally-facing APIs to internally used APIs.

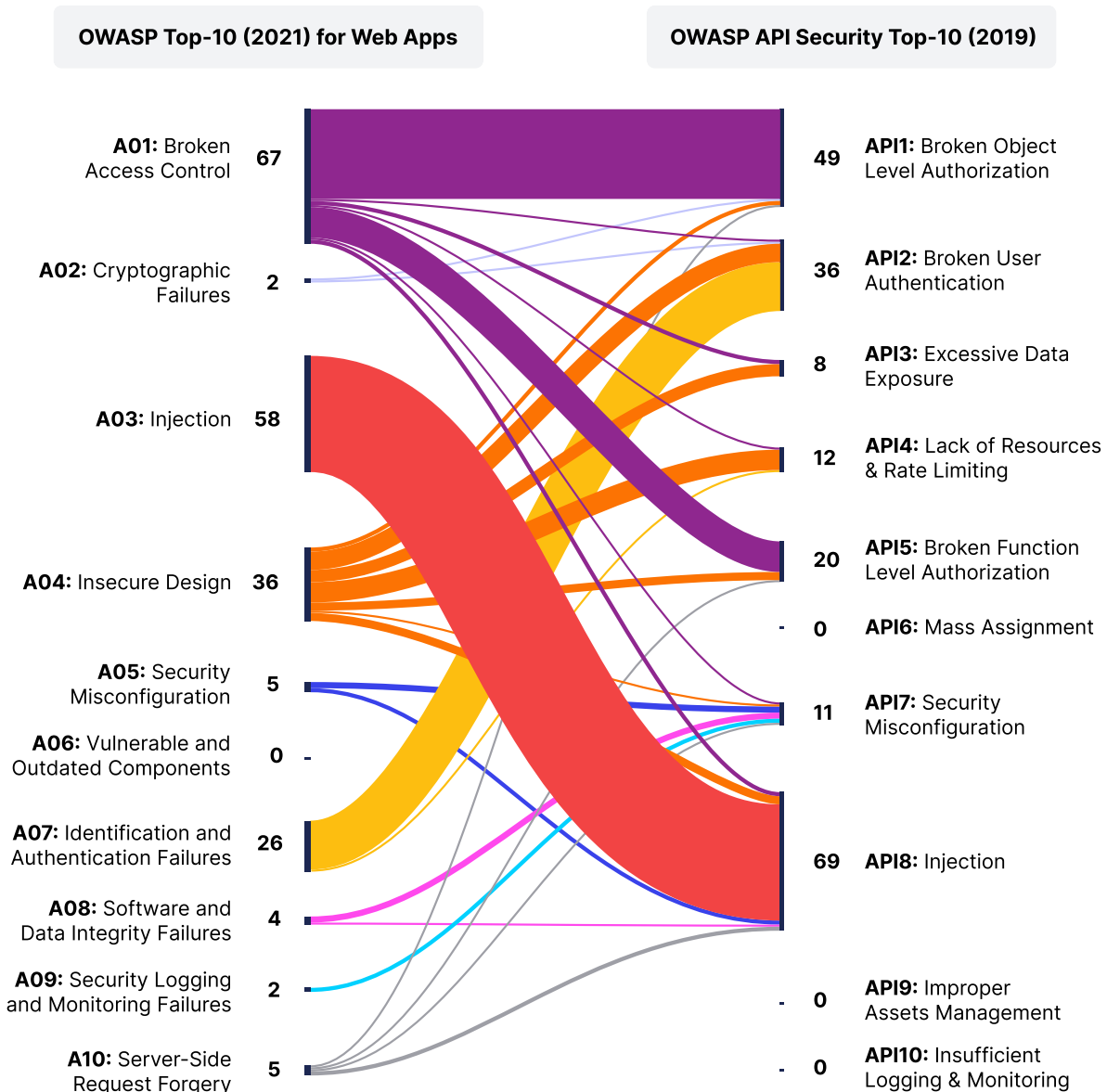
Category	Vendor	CVE	OWASP (2019)	CVSS	OSS
DevTools	GitLab	CVE-2022-2884	API8	<b>9.9</b>	Y
	Rancher	CVE-2021-36783	API1	<b>9.9</b>	Y
	Argo Project	CVE-2022-31105	API2	<b>9.6</b>	Y
	Casdoor	CVE-2022-38638	API8	<b>9.1</b>	Y
	Grafana	CVE-2022-31107	API1	<b>7.5</b>	Y
	HashiCorp	CVE-2021-41803	API8	<b>7.1</b>	Y
	GitLab	CVE-2022-1999	API1	<b>5.3</b>	Y
	Kubernetes	CVE-2022-3172	API1	<b>5.1</b>	Y
	JFrog	CVE-2021-46687	API1	<b>4.9</b>	Y
Enterprise HW / SW	Apache Software Foundation	CVE-2022-25168	API8	<b>9.8</b>	Y
	Fortinet	CVE-2022-29060	API2	<b>8.1</b>	N
	F5 Networks	CVE-2022-34851	API8	<b>6.5</b>	N

A closer look at these 12 CVEs shows that about 84% of them are open-source tools, with the remaining 16% commercial tools. We also find that 42% of them are related to Injection issues ([API8:2019](#)), another 42% are related to Broken Object Level Authorization (BOLA) issues ([API1:2019](#)), and the remaining 16% are associated with Broken User Authentication (BUA) issues ([API2:2019](#)).

While not all of these products have critical or even high CVSS scores (they range from 4.9 to 9.9), their ubiquity and importance to the SLDC makes them particularly impactful.

## Injections

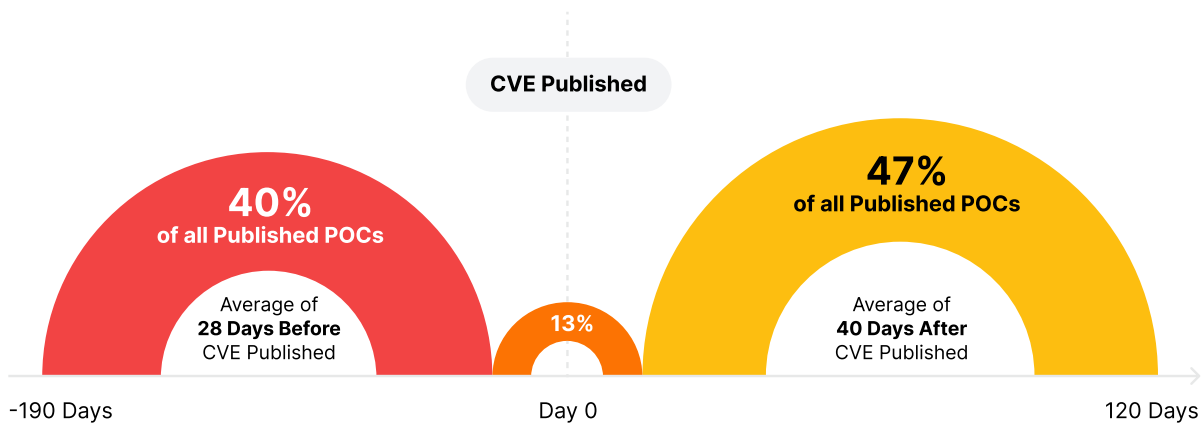
Because API risks remain high, both in terms of total CVEs and CVSS scores, there is the issue of how to triage them all. We found that the OWASP Top-10 Injection categories ([A03:2021](#) for web apps and [API8:2019](#) for APIs) topped the list at over 33% of all CVEs analyzed. After drilling into this further, we determined that nearly all of the [2022 Top 25 Most Dangerous Software Weaknesses](#) are included, and that many, many variations of the injections threat vector are covered.



This will undoubtedly require additional root-cause analysis and remediation effort.

## Exploit Timeline

Finally, we looked at published exploit POCs and found the number had dropped by half between Q2-2022 to Q3-2022 – from 61 to 30, or 33% and 15% of all CVEs analyzed, respectively – but with a **much faster median time-to-exploit**: the same day the CVE was published. And looking even more closely, we determined that over 53% of these exploit POCs had been released on or before the CVE publish date (aka 0-days), with an average time-to-exploit of about **28 days before** the CVE was published. The other 47% were published an average of 40 days after the CVE was published.



This somewhat surprising finding will doubtless add some urgency to your mitigation timeline.

## Other Findings

Much of the effort when analyzing these API vulnerability data goes into chasing down hypotheses to see if the data support them – so-called exploratory data analysis. It's said that if you torture the data long enough, it will confess to anything<sup>2</sup> – and while in the Q2 report we dug into these confessions with much gusto, in this quarter's report we'll just give you a taste, the high-level overview.

## CVSS Severity Ratings

As we have done all year, we look at the CVSS severity scores as one way to grasp the magnitude of the issues we're facing. Looking at the data over the past year, we see some movement in the averages but a rock-steady median; even the critical / high count does not move too much. All this suggests that, so far, we've found the bar and now need to be watching for significant changes which might impact our VM efforts. See table on next page.

---

<sup>2</sup> Attributed to Nobel Prize Economist [Ronald Coase](#)

	Q3-2022	Q2-2022	Q1-2022
Total count	203	184	50
Average CVSS	7.4	7.3	7.6
Median CVSS	7.5	7.5	7.5
Range	3.3 to 10.0	2.6 to 10.0	3.6 to 10.0
Critical / High count	115 (56.7%)	105 (57%)	30 (60%)

## OWASP Analysis

Another area that we take a magnifying glass to is the OWASP mapping: how do the API vulnerabilities look with respect to both [OWASP Top-10](#) (2021), which focuses on web applications and [OWASP API Security Top-10](#) (2019), which focuses on APIs. We like to slice & dice the data in several ways to look at how well the OWASP mapping of API vulnerabilities plays out in the real world.

Here we'll focus on the API Security list from OWASP:

	Q3-2022	Q2-2022	Q1-2022
Top-3 by count	API8 (33.7%) API1 (23.9%) API2 (17.6%)	API8 (38.6%) API1 (17.4%) API2 (14.7%)	API8 (40.0%) API5 (26.0%) API1 / API3 (10%)
Top-3 by CVSS	API5 (7.97) API2 (7.68) API1 (6.95)	API10 (9.80) API1 (7.76) API4 (7.72)	API2 (8.37) API5 (7.81) API8 (7.55)
Top-3 by rank <sup>3</sup>	API8 API1 API2	API8 API1 API2	API8 API5 API1

These data are in keeping with one of our Q3-2022 key findings: Injections (API8) are the biggest risk factor in your API portfolio. So one way of looking at it is that any of these popular approaches to triaging vulnerabilities is probably going to serve you well – but let's keep digging to see if the data show any other useful approaches.

<sup>3</sup> See [https://cwe.mitre.org/top25/archive/2022/2022\\_cwe\\_top25\\_supplemental.html#methodDetails](https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25_supplemental.html#methodDetails) or our Q2 report for more on our methodology

## Most Dangerous CWEs Mapping

In late-June, MITRE and CISA published their 2022 “Most Dangerous Software Weaknesses” list<sup>4</sup>. As we did in previous quarters, we compared our API vulnerability reports to this list. The results were similar: some movement in the percentage impacted, but nothing that appeared statistically significant.

Instances of a top-25 CWE found, by quarter:

	Q3-2022	Q2-2022	Q1-2022
Count	109	94	30
Percentage	54%	51%	60%

That is not to say that paying attention to CWEs (and indeed the Most Dangerous list) is unimportant. As we noted last quarter, it provides deeper insight and nuance into understanding what particular threat vectors are in play – for instance, as noted above, it gives us insight into the breadth of injection vulnerabilities we have to deal with. So, to reiterate from last quarter, this provides security and DevOps teams with another arrow in their vulnerability management quiver.

## Known Exploits

Having already explored how long it takes for known exploit POCs to get published against API vulnerabilities in Q3 (see Key Findings [here](#)), we now take broader look at these threats – who do they impact, and what’s it mean to your VM program?

	Q3-2022			Q2-2022		
	Total count	Exploit count	Percent	Total count	Exploit count	Percent
Vulnerabilities	203	30	14.8%	184	61	33.2%
Companies	129	27	20.9%	111	39	35.1%
Product Categories	9	7	77.8%	18	11	61.1%
CWEs	59	20	33.9%	58	25	43.1%
OWASP Top-10 (2021) for web apps	10	5	50.0%	10	5	50.0%
OWASP API Security Top-10 (2019)	10	5	50.0%	10	8	80.0%

<sup>4</sup> See [https://cwe.mitre.org/top25/archive/2022/2022\\_cwe\\_top25.html](https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html)

These data<sup>5</sup> are not overly informative; even looking at the average CVSS scores (7.7 for exploited vulnerabilities vs. 7.4 for unexploited vulnerabilities) is not too enlightening. None of these data points should be used to attempt any trend analysis at this time; instead, they merely suggest at this point that we should monitor the situation.

In fact, perhaps the only data we should be actively watching for is the median time-to-exploit which, as mentioned above, was zero (0) days in Q3, versus about 2-½ weeks in Q2.

## Open-Source vs. Commercial

One of several questions we always examine is: Are Open-Source Software (OSS) APIs more secure than Commercial APIs? And the answer in this quarter is not much different from last quarter: no, not really.

	Q3-2022	Q2-2022	Q1-2022
OSS products	62.4%	60.3%	54.0%
Commercial products	37.6%	39.7%	46.0%

While it's way too early to ascribe causation, the trend seems clear: you ignore open-source API products at your peril.

## Vendors / Products

Another issue we always dig into is: who are the vendors behind these vulnerabilities, and do we see any particularly high concentrations of issues with specific vendors.

Vulnerability count / vendor	Q3-2022	Q2-2022	Q1-2022
1	72.9%	71.2%	81.8%
2	16.3%	11.7%	6.1%
3	4.7%	9.0%	6.1%
4+	6.2%	8.1%	6.1%

Here again, we're not seeing any signs indicating abnormal concentrations of vulnerabilities with any particular vendors; there does appear to be a trend in the percent of vendors with two (2) vulnerabilities, but this is probably more because of the sample size. Bottom line, these data probably only become interesting if we see one of these groupings spike or drop suddenly.

---

<sup>5</sup> We did not collect this information in Q1

## Triaging API Vulnerabilities

We have presented several criteria by which security teams can assess and prioritize API and web app vulnerabilities for remediation. Here we look at the results of some of the more commonly used approaches to prioritize mitigation efforts.

### Vendor-Centric Approaches

We start by asking the question: are any particular vendors over-represented in the data? That is, are we more likely to suffer a successful attack because we have vendor X in our environment? There are a couple of ways to cut the data to look for clues on this point – by frequency (how often do we see a vendor's products in the data), by severity (on average, how bad are a vendor's vulnerabilities), and/or by ranking (a synthesis of frequency & severity). Let's take a look.

**Top-5 based on Frequency** – vendors with products which garnered the most vulnerabilities; in Q3 this includes seven (7) vendors.

Rank	Vendor	Count	Avg CVSS	Ranking
1	Red Hat	10	7.8	62.50
2	Jenkins	8	6.1	25.00
3	Cisco	6	7.6	32.41
3	Zyxel	6	5.7	13.56
5	Tabit Technologies	5	7.4	24.29
5	Harbor	5	6.6	18.57
5	GitLab	5	5.7	11.27

**Top-5 based on Severity** – vendors with products which amassed the highest average CVSS scores; in Q3 this includes 21 vendors, all with an average CVSS score in the 9.9 to 9.8 (critical) range.

SUSE	miniOrange	CNCF	Eric Cornelissen	laverdet	Polly	Transtek
AEB-labs	Acrontum	dotCMS	Jeecg	MiCODUS	Six Apart	unknown
Carlo Gavazzi	Alfasado	dotnetcore	KubeVela	Peisheng Information	some-natalie	Wavlink



**Top-5 based on Ranking** – vendors with products which accumulate the highest calculated impact<sup>6</sup> ranking.

Rank	Vendor	Count	Avg CVSS	Ranking
1	Red Hat	10	7.8	62.50
2	Cisco	6	7.6	32.41
3	Jenkins	8	6.1	25.00
4	Tabit Technologies	5	7.4	24.29
5	Harbor	5	6.6	18.57

Unsurprisingly, we don't see a lot of overlap with Q2 results. In fact, only two (2) vendors found in these lists were also included in the Q2 results: GitLab and Zyxel. This could be an indication of some sort of systemic issue (bad) or simply because of the popularity and/or breadth of offerings (more likely). While this bears watching over the long haul, and may serve as a signal to take a somewhat closer look, it's important not to succumb to the [clustering illusion](#).

In addition, while interesting and possibly germane, these approaches to triaging vulnerabilities for mitigation are highly dependent on a number of factors, such as:

- Are these vulnerabilities against a single product or multiple products?
- Are multiple CVE variations submitted against a single product, or are they aggregated?
- Are products from these vendors in your environment, and are they exposed?
- What are the probability and impact of a breach involving one of these vulnerabilities?

Bottom line: the approach you take should depend on the specifics of your API portfolio and your unique risk tolerance and assessment. To further this point, we'll now present two additional approaches for your consideration.

## EPSS Approach

The Exploit Prediction Scoring System (EPSS) Model is a community-driven effort at FIRST (Forum of Incident Response and Security Teams) to combine descriptive information about vulnerabilities (CVEs) with evidence of actual exploitation in-the-wild to estimate the likelihood that a vulnerability will be exploited.<sup>7</sup>

---

<sup>6</sup> As discussed in the Q2 Vulnerability & Exploit report, we apply the methodology used by MITRE to assess CWEs based on {normalized frequency} x {normalized CVSS avg}; for more on this approach, see [https://cwe.mitre.org/top25/archive/2022/2022\\_cwe\\_top25\\_supplemental.html#methodDetails](https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25_supplemental.html#methodDetails)

<sup>7</sup> For more on the EPSS Model, see <https://www.first.org/epss/model>

We explored how to adapt this approach to our data to assess how effective a CVSS-focused remediation strategy works. Here, we want to assess the efficiency and coverage resulting from a “focus on all API CVEs with CVSS scores of 7 and above” approach. We use the data to find how many CVEs would be correctly prioritized (i.e., true positives), incorrectly prioritized (i.e., false positives), correctly delayed (i.e., true negatives) and incorrectly delayed (i.e., false positives) using this approach.

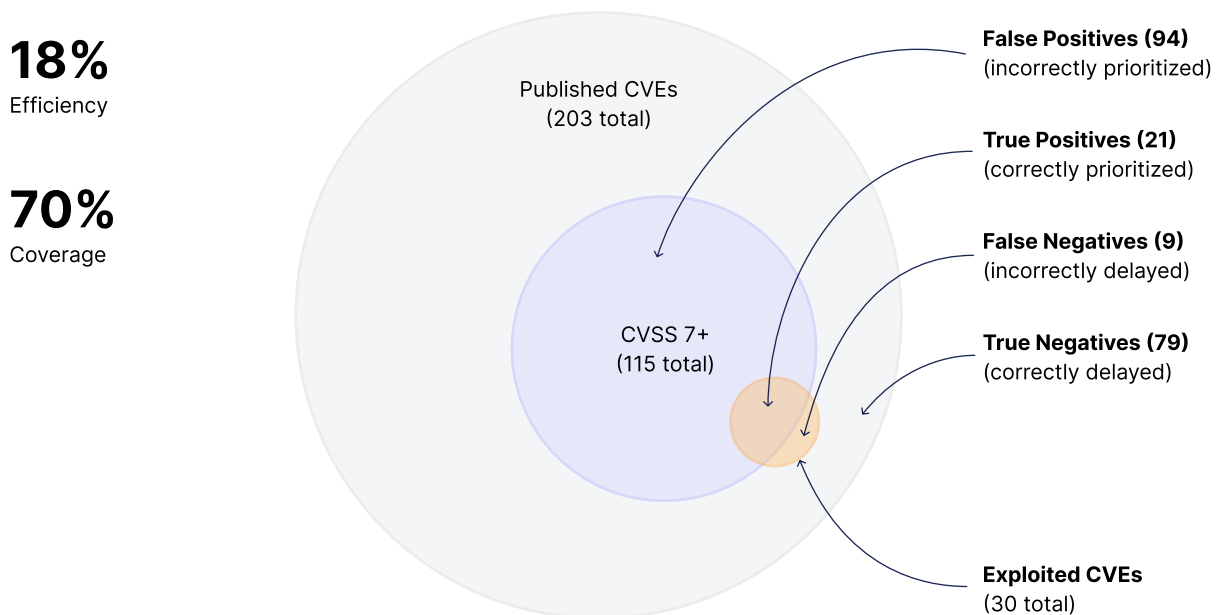
Starting with the 203 CVEs published in Q3-2022 and map known exploits across the four categories.<sup>8</sup>

CVEs with CVSS scores of 7+		CVEs with CVSS scores <7	
Exploited CVEs (TPs)	Unexploited CVEs (FPs)	Exploited CVEs (FNs)	Unexploited CVEs (TNs)
21	94	9	79
10.3%	46.3%	4.4%	38.9%

From here, we can determine the Efficiency (E) and Coverage (C) of this approach for Q3:

- $E = TP / (TP + FP) = 21 / (21 + 94) = 18\%$
- $C = TP / (TP + FN) = 21 / (21 + 9) = 70\%$

This can be visualized as follows:



**Notes:**

- For graphing purposes, let's assume CVSS = n/a means <7
- Based on actual known exploits, not hypothesized future exploits

<sup>8</sup> For purposes of this exercise, we've pooled CVEs with CVSS = n/a into the “less than 7” group






This compares to E = 37% and C = 64% in Q2, or E = 27% and C = 66% for the last two quarters combined. This suggests that while the “focus on all API CVEs with Critical or High CVSS ratings” approach might result in good coverage, it is not terribly efficient – if you prioritized all 115 Critical and High CVEs, over 80% of your effort would have been for naught (or at least incorrectly prioritized).





Of course, there are many caveats to this assessment – for instance: unknown exploits; what’s in your environment; what’s exposed, knowingly and unknowingly, in your environment; *et cetera* – which need to be considered. In addition, it would be useful if this assessment could be made predictive (the true intent of the EPSS model) – work continues on that front. In the meantime, it’s another reminder that your API vulnerability management program must be tailored to your situation instead of blindly following some age-old adage.

## Most Impactful API Vulnerabilities




Another approach is to rely on the expertise of long-time API security providers (such as, of course, Wallarm). In going through the Q3-2022 data, we assess these 12 CVEs to be the most impactful API vulnerabilities in Q3-2022, primarily due to the impact on development and delivery infrastructure.

### Category: Dev Tools

 <b>GitLab</b> <b>CVE-2022-2884</b> GitLab Remote Command Execution <b>CWE-77:</b> Improper Neutralization of Special Elements used in a Command ('Command Injection')	CVSSv3: 9.9	<b>A03</b> OWASP API Top-10 <b>API8</b> OWASP API Security Top-10
 <b>RANCHER</b> <b>CVE-2022-36783</b> Rancher - Failure to properly sanitize credentials in cluster template answers <b>CWE-312:</b> Cleartext Storage of Sensitive Information	CVSSv3: 9.9	<b>A01</b> OWASP API Top-10 <b>API1</b> OWASP API Security Top-10
 <b>argo</b> <b>CVE-2022-31105</b> Argo CD Improper Certificate Validation <b>CWE-295</b> Improper Certificate Validation	CVSSv3: 9.6	<b>A07</b> OWASP API Top-10 <b>API2</b> OWASP API Security Top-10
 <b>Casdoor</b> <b>CVE-2022-38638</b> Casdoor Arbitrary file write/overwrite Vulnerability <b>CWE-862:</b> Missing Authorization	CVSSv3: 9.1	<b>A04</b> OWASP API Top-10 <b>API8</b> OWASP API Security Top-10
 <b>Grafana</b> <b>CVE-2022-31107</b> Grafana Account Takeover Via OAuth Vulnerability <b>CWE-863</b> Incorrect Authorization	CVSSv3: 7.5	<b>A01</b> OWASP API Top-10 <b>API1</b> OWASP API Security Top-10

 <p><b>CVE-2022-41803</b> HashiCorp Consul Auto-Config JWT Authorization Missing Input Validation  <b>CWE-862</b> Missing Authorization</p>	CVSSv3: 7.1	<p><b>A03</b> OWASP API Top-10</p> <p><b>API8</b> OWASP API Security Top-10</p>
 <p><b>CVE-2022-1999</b> GitLab CE/EE Improper privilege Management  <b>CWE-269</b>: Improper Privilege Management</p>	CVSSv3: 5.3	<p><b>A04</b> OWASP API Top-10</p> <p><b>API1</b> OWASP API Security Top-10</p>
 <p><b>CVE-2022-3172</b> Kubernetes Aggregated API server can cause clients to be redirected  <b>CWE-918</b>: Server-Side Request Forgery (SSRF)</p>	CVSSv3: 5.1	<p><b>A10</b> OWASP API Top-10</p> <p><b>API1</b> OWASP API Security Top-10</p>
 <p><b>CVE-2022-4668</b> JFrog Artifactory Sensitive Data Exposure  <b>CWE-359</b>: JFrog Artifactory Sensitive Data Exposure</p>	CVSSv3: 4.9	<p><b>A01</b> OWASP API Top-10</p> <p><b>API1</b> OWASP API Security Top-10</p>

### Category: Enterprise HW / SW

 <p><b>CVE-2022-25168</b> Apache Hadoop Arbitrary Commands Injection  <b>CWE-88</b>: Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')</p>	CVSSv3: 9.8	<p><b>A03</b> OWASP API Top-10</p> <p><b>API8</b> OWASP API Security Top-10</p>
 <p><b>CVE-2022-29060</b> FortiDDoS - Use of hardcoded key for the JWT token  <b>CWE-321</b>: Use of Hard-coded Cryptographic Key</p>	CVSSv3: 8.1	<p><b>A02</b> OWASP API Top-10</p> <p><b>API2</b> OWASP API Security Top-10</p>
 <p><b>CVE-2022-34851</b> Argo CD Improper Certificate Validation  <b>CWE-20</b> Improper Input Validation</p>	CVSSv3: 6.5	<p><b>A03</b> OWASP API Top-10</p> <p><b>API8</b> OWASP API Security Top-10</p>

We've already discussed the potential impact these have on your internal processes, so the key point here is again to take time to assess vulnerabilities which – while perhaps not the most severe, not only customer-facing, or even not with a known exploit – might have a much wider blast radius should they be exploited.

## What's Right for Your Organization?

Expanding your vulnerability management program to cover APIs will require visibility across your entire API portfolio, assessing and triaging API vulnerabilities based on your environment, your actual traffic, and ensuring mitigations are implemented – both in the code and at run-time. Refer to our [API Security Tutorial](#) for more information.

## Conclusion

An important part of any vulnerability management program is to reduce the number of in-depth assessments required, to limit the impact on your security and DevOps teams and to surface the truly important vulnerabilities in your environment which must be addressed. With this effort, Wallarm has not only reduced the over 100,000 reports down to 203 API-specific issues, but we've also provided for your consideration various methods to help triage and mitigate those that are important in your environment.

This is our third API vulnerability and exploits report issued this year, and the trends we're seeing in terms of the number, severity, type, and focus of API vulnerabilities should be taken seriously. In short, the top 10 API issues affecting core DevOps and PaaS products, such as Kubernetes, Rancher, GitLab, HashiCorp, and several more, are concerning. In addition, the prevalence and variety of injection vulnerabilities is troubling. And finally, the almost simultaneous release of exploits, leaving little time to react & patch, is also worrying.

This impacts everyone working in the API ecosystem – whether developing APIs for customer use, leveraging them to support critical business purposes, or something else – and it's important to work with a partner who can explain what these trends really mean and how to mitigate the associated risks and impacts.

## Learn More / Add'l Resources

Want to learn more about API vulnerabilities and exploits? Here are some resources:



Join your peers following our new [API ThreatStats](#) LinkedIn page to keep up-to-date on all the latest [#apisecurity](#) exploits and updates.



For more on the Q3-2022 API ThreatStats™ Report, read the [blog](#), download the [infographic](#) and/or watch the [webinar](#).



Subscribe to our newsletter at [lab.wallarm.com](https://lab.wallarm.com)



Contact us via the web at <https://www.wallarm.com/request-demo> or via email at [request@wallarm.com](mailto:request@wallarm.com) to set up a personal demo with one of our security experts.

## About Wallarm

Wallarm delivers security software and services providing robust protection for APIs, web applications, microservices, and serverless workloads running in cloud-native environments. Hundreds of Security and DevOps teams choose Wallarm to discover all APIs and web applications running in their environment, to get full visibility into malicious traffic, to protect their entire public, private and partner API portfolio, and to automate incident response for their cybersecurity programs. We support modern tech stacks and API protocols, and offer deployment options for SaaS, multi- and hybrid-cloud, Kubernetes and more. Wallarm is headquartered in San Francisco, California, and is backed by Toba Capital, Y Combinator, Partech, and other investors.



# Appendix

## Methodology

This research is based only on security bulletins and bug bounty reports which became publicly available in Q3-2022. Some of the issues covered in this report are associated with CVE identifiers early, but full details only disclosed in Q3-2022. Hence, readers might note a few CVEs from 2021 or earlier in this report.

Every report disclosed or published in the Jul-01 to Sep-30, 2022 timeframe was analyzed by the Wallarm Research Team and included to this report if it met the following criteria:

- Affects any public or private API, including REST, gRPC, GraphQL, WebSocket, or other API protocols;
- **OR** is related to core API and/or service mesh components, including API Gateways and API Management solutions; and is
- **NOT** related to binary APIs, including Android, iOS, ChromeOS or other BAPI.

Every issue that satisfied these criteria was then categorized by OWASP Top-10 (2021) for web apps and OWASP API Security Top-10 (2019) groups in parallel for cross-referencing. We also enriched the data with software type (OSS vs commercial) and exploit information.

In addition, all vulnerable products were categorized into the following groups:

1. **Development Frameworks.** Libraries and frameworks used to build software, including Spring Framework, Django, and others.
2. **DevOps and Cloud-Native Software.** Everything that DevOps teams rely on, from load balancers to monitoring systems and Kubernetes components & tools.
3. **SaaS.** Any public APIs and Software-as-a-Service products.
4. **Enterprise Software.** B2B and enterprise software, including security products, communication tools, and other solutions.
5. **Enterprise Hardware.** Any business devices, including security & firewall appliances, VoIP and other hardware.

Some companies, such as Cisco and Ping Identity, have different products in different categories and therefore they might be mentioned more than once. The same holds for certain companies, such as Google and VMware, which show up in both OSS and commercial product groupings.

We will make the data analyzed for this report – including the full list of API security issues along with CVSSv3 scores and other attributes – available to accredited researchers for further assessment; please contact us at [research@wallarm.com](mailto:research@wallarm.com) to learn more.

## EPSS Efficiency and Coverage

Here is how FIRST defines efficiency and coverage<sup>9</sup>, and why they view it as important:

Using these four categories (TP, FP, FN, TN) we can derive two more meaningful metrics, what we've termed as **efficiency** and **coverage** (or what information theory calls [precision and recall](#) respectively).

**Efficiency** considers how efficiently resources were spent by measuring the percent of remediated vulnerabilities that were exploited. In the above diagram, efficiency is the amount of the blue circle covered by the red circle. Remediating mostly exploited vulnerabilities would be a high efficiency rating (resources were allocated efficiently), while remediating perhaps random or mostly non-exploited vulnerabilities would result in a low efficiency rating. Efficiency is calculated as the number of exploited vulnerabilities prioritized (TP) divided by the total number of prioritized vulnerabilities (TP+FP).

**Coverage** is the percent of exploited vulnerabilities that were remediated, and is calculated as the number of exploited vulnerabilities prioritized (TP) divided by the total number of exploited vulnerabilities (TP + FN). In the above diagram, coverage is the amount of the red circle covered by the blue circle. Having low coverage indicates that not many of the exploited vulnerabilities were remediated with the given strategy.

Measuring and understanding both efficiency and coverage allows different firms to approach vulnerability remediation differently based on their risk tolerance and resource constraints. Firms that do not have many resources may wish to emphasize efficiency over coverage, attempting to get the best impact from the limited resources available. But for firms where resources are less constrained and security is critical for success, the emphasis can be on getting high coverage of the highest risk vulnerabilities.

## Analysis Notes

A couple of notes about the analyses found in this report:

1. **CVSS Severity Ratings.** The CVSS scores were initially collected “on the fly” as CVEs were published, and our initial analysis & reporting was based on these. We updated that snapshot of CVSS scores in mid-October for this report. Since CVSS ratings are on an as-needed basis, there may be differences between this and any previous releases of this data, in some cases significantly.
2. **Thanks.** Thanks to Goran, Dinko & Alex for the data and research, Anastasia & Mike for the graphics and design, and Torry & William for their reviews and comments.

---

<sup>9</sup> See <https://www.first.org/epss/model> for the complete discussion





Book a [Wallarm demo](#)  
or start your [free trial](#) now

(415) 940-7077  
188 King St. Unit 508, San Francisco, CA 94107  
[www.wallarm.com](http://www.wallarm.com)