A MATTER OF PATCH

Managing Vulnerabilities with Runtime Memory Analysis

Rezilion

According to analyst firm IDC, large-to-very large enterprise companies are spending 7-10% of their security budget on vulnerability management.[1]

SECURITY TEAMS ARE INUNDATED by vulnerability scanners that overload them with voluminous scan results. In order to cope, they prioritize vulnerability remediation using antiquated best practices and limited data, which provably do not reduce their risk or attack surface. In fact, a recent RAND Corporation analysis found no notable reduction of breaches in organizations with mature vulnerability management programs.[2] Firms with good security posture are equally breached by known vulnerabilities as those with poor security posture.

Vulnerability management firms often point at the multiplicative growth of Common Vulnerabilities and Exposures (CVE) as a proofpoint for increased investment in their tools, but there is no correlation between CVE sprawl and breaches due to vulnerability exploitation. In 2019, an analysis of the IBM X-Fore Vulnerability database[3] revealed that though there 16,000 new vulnerabilities discovered, only less than 5% led to exploitation. In 2021, 20,130 software vulnerabilities[4] were reported, but only 4% of them posed a high risk to organizations. This paints a murky picture of the enterprise attack surface.

Heartbleed (CVE-2014- 0160), one of the most devastatingly exploited vulnerabilities in the history of the internet, received a CVSS2 score of only 5.0/10.

Vulnerability Prioritization?

A popular prioritization methodology is the Common Vulnerability Scoring System (CVSS) — an open framework for communicating the characteristics and severity of software vulnerabilities.

CVSS consists of three metric groups: Base, Temporal, and Environmental. The Base metrics produce a score ranging from 0 to 10, which can then be modified by scoring the Temporal and Environmental metrics.
However, a vulnerability is only as dangerous as the threat exploiting it, and 95% of vulnerabilities with "high severity" CVSS scores have never been seen in the wild, nor linked to breaches.[3] This means that assigning a global critical/high/medium/low rating to any vulnerability is flawed because:

- Attackers don't care what threat score a vulnerability has and regularly exploit lower-ranked vulnerabilities if they're the easiest successful attack vectors.
- The known quantity and explosive growth of identified vulnerabilities makes it impossible to remediate them all.
- Not all vulnerabilities have patches or can be patched.

The CVE process is inhibitive, even when there is a known vulnerability in a software package the vulnerable components are typically not executed to memory, and are therefore, not exploitable. Capsul8 recently made some headway along this thinking in their analysis of Linux hardening when they concluded: **"Even in the presence of a CVE, it is not directly obvious if a vulnerability is exploitable."**[5]

[1]*Worldwide Security Spending Guide, IDC*

[2]*Improving Vulnerability Remediation Through Better Exploit Prediction, RAND Corporation*

[3]*Source: https://www.kennasecurity.com/news/kenna-research-shows-likelihood-of-organizational-exploitation/*

[4] *State of Vulnerability Risk Management Report, NopSec*

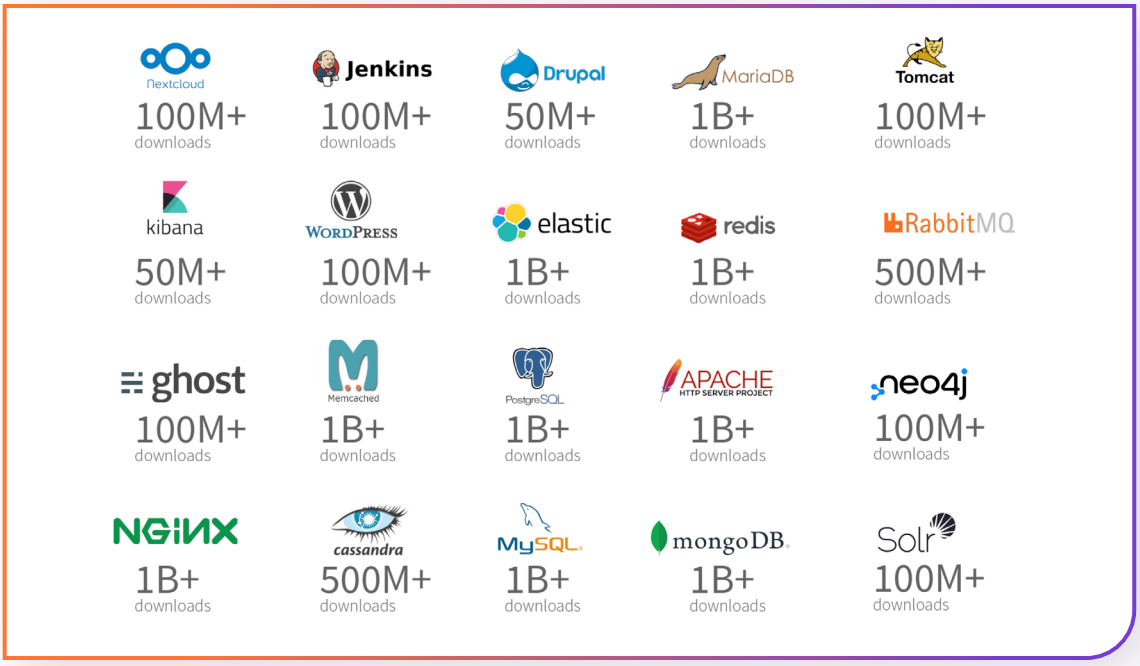
[5] *Linux hardening in the wild, Theofilos Petsios, Capsul8*

Prioritizing Vulnerabilities – Industry Wide

Developing a risk appetite formula requires an accounting for likelihood of exploitation, in other words, a mechanism to identify which vulnerabilities pose actual threat. Traditional approaches to vulnerability management have also exacerbated the culture of friction that exists between security and DevOps teams who often have competing deliverables: security vs uptime.

With cloud workloads, DevOps are deploying code and infrastructure more rapidly than ever, and with this rate of change, it's critical for security teams to develop a model for managing risk across the entire attack surface. We at Rezilion wanted to demonstrate the fact that most vulnerabilities do not pose an actual risk as they are associated with packages that are not loaded to memory.

For that purpose, we examined 20 of the most popular container images on DockerHub6 that have collectively been downloaded and deployed billions of times, as well as several base OS images from leading cloud providers (AWS, GCP, and Azure) to assess how many vulnerabilities aren't applicable and which vulnerabilities pose an actual risk.



Container Analysis

Analysis of these 20 container images found over 4,347 known vulnerabilities. Patching all these vulnerabilities at once would be tedious and practically impossible. However, during the course of testing — on average — only approximately 15% of found CVEs were ever loaded into memory, thus not posing any threat.

Container Name	Total CVEs	Loaded Vulnerabilities	Not Loaded	% Not Loaded
jenkins/jenkins	44	1	43	97.73%
kibana	349	71	278	79.66%
mariadb	81	1	80	98.77%
memcached	90	18	72	80.00%
mongo	365	20	345	94.52%
mysql	45	2	43	95.56%
mariadb	31	0	31	100.00%
memcached	71	13	58	81.69%
mongo	37	23	14	37.84%
mysql	167	27	140	83.83%
nextcloud	497	107	390	78.47%
nginx	118	16	102	86.44%
postgres	110	31	79	71.82%
rabbitmq	15	0	15	100.00%
redis	71	13	58	81.69%
tomcat	152	14	138	90.79%
wordpress	467	106	361	77.30%
elasticsearch	63	5	58	92.06%
neo4j	75	12	63	84.00%
solr	106	15	91	85.85%
	4,347	**683**	**3,664**	

PERCENTAGE LOADED

15.71%

AVERAGE OF VULNERABILITIES NOT LOADED TO MEMORY: 85%

- All container images and base OS images were executed using their recommended/default settings.
- Each image was scanned for vulnerabilities using the open source Trivy vulnerability scanner.
- A list of vulnerabilities and their respective packages was assembled.
- Memory analysis was conducted and each file loaded in memory was mapped to the package it originated from.
- Lastly, the list of packages that had at least one file loaded to memory was compared to the list of vulnerable packages initially enumerated, and the delta between vulnerabilities found and vulnerabilities running in each container environment was assessed.

Host Analysis

Rezilion analyzed 12 different base OS images of major cloud providers (AWS, Azure, and GCP). In total, these images contained 5,167 known vulnerabilities. Out of which, approximately 20% were ever loaded to memory.

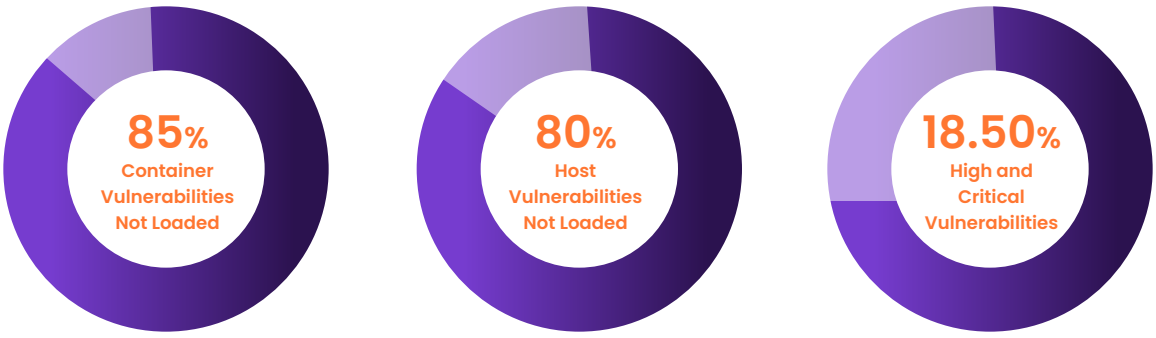
Container Name	Total CVEs	Loaded Vulnerabilities	Not Loaded	% Not Loaded
GCP Ubuntu 20	64	19	45	70.31%
GCP Ubuntu 18	72	10	62	86.11%
GCP Debian 10	5	1	4	80.00%
GCP CentOS 7	851	167	684	80.38%
AWS Ubuntu 20.04.3	142	45	97	68.31%
AWS Ubuntu 18.04.6	202	81	121	59.90%
Azure Ubuntu 20	114	22	92	80.70%
Azure Ubuntu 18	143	31	112	78.32%
Azure Ubuntu 16	331	132	199	60.12%
Azure Debian 10	1,403	280	1,123	80.04%
Azure CentOS 7	818	99	719	87.90%
Azure CentOS 8	1,022	69	953	93.25%
	5,167	**956**	**4,211**	

PERCENTAGE LOADED

18.50%

AVERAGE OF VULNERABILITIES NOT LOADED TO MEMORY: 80%

During the course of testing and analysis:



It is clear from the analysis that up to 85% of all the discovered vulnerabilities in containers and hosts were never loaded to memory and therefore, were not exploitable. If traditional vulnerability management approaches were used, one would spend upward of 85% of patching time and efforts on vulnerabilities that posed no actual risk to the environment.

Detection of installed, vulnerable components does not edify which code parts utilize them—akin to testing for operating system dependencies that inform which vulnerable libraries are installed, but cannot tell which apps are actually using these libraries. Monitoring which libraries are actually loaded in runtime is the right approach to successful vulnerability prioritization.[7]

Vulnerability Prioritization Helps Guide Mitigation Efforts

Implementing a risk-based approach to prioritizing vulnerability remediation focuses efforts on vulnerabilities, for which imminent threats prevail. This reduces risk, friction, and—by channeling remediation efforts at vulnerabilities that represent true risk—also reduces overhead and vulnerability backlog. This approach will result in a reduced software attack surface and will provide "breathing room" for additional patch installation. Adopting this approach to vulnerability management enables security teams to:

- Use more context and runtime visibility for continuous, adaptive risk-based decision making, rather than static "score"-based or manual policy driven binary "allow or block" security decisions.
- Allow development and product security teams to filter out the noise and focus on vulnerabilities that are actually exploitable so that they can release secure products quickly.
- Enable their risk management teams to move beyond risk management and compliance checklists to real-time contextualized security control decisions.
- Reduce their patching efforts by up to 85%.
- Provide continuous risk visibility feedback to DevOps and business units to adjust acceptable risk levels and controls as necessary.
- Save time and reduce risk by automating remediation and shortening the attack window.

[7] *Note that source code scanning is not a viable approach to solving this problem in that the method can't know what components will end up being used once in production, and recipes often pull in a library not specified in a manifest file or integrate external third party components as part of the build.*

[8] *Implement a Risk-Based Approach to Vulnerability Management, Gartner*

ABOUT REZILION

Rezilion's platform automatically secures the software you deliver to customers. Rezilion's continuous runtime analysis detects vulnerable software components on any layer of the software stack and determines their exploitability, filtering out up to 95% of identified vulnerabilities. Rezilion then automatically mitigates exploitable vulnerabilities across the SDLC, reducing vulnerability backlogs and remediation timelines from months to hours, while giving DevOps teams time back to build.

Learn more about Rezilion's software attack surface management platform at www.rezilion.com and get your 30 day free trial www.rezilion.com.