NETRISE

Supply Chain Visibility & Risk Study

Edition 2: Containers; Q4 2024

Executive Summary: Overview and Key Findings

All companies rely on software to power their business. However, software development pressures and the corresponding enterprise software sprawl is driving up software supply chain risks. These risks are much greater than most security professionals understand. In fact, the software risk data most rely on today is only the tip of the iceberg and misses many of what should be considered the highest priority software risks that exist in the enterprise.

Containers are driving a growth of cloud-native technologies, fundamentally changing how many modern applications are designed, deployed, and managed. In fact, containers are the fastest growing - and weakest cybersecurity link - in software supply chains. And while they are considered lightweight and simple, containerized software / images are much more complex than commonly understood and they are introducing numerous risks and security challenges that require new solutions and skills to mitigate efficiently.

Key Findings and Recommendations

1 | Containers May Be Considered 'Lightweight' - But They Are Much More Complex Than Most Think

- Using a compiled and interpreted code analysis approach, we analyzed 70 randomly selected container images from 250 of the most commonly downloaded images on Docker Hub and generated detailed SBOMs. On average, within each container image we found 389 software components.
- Understanding containerized software and associated risks starts with the foundational step of getting visibility into the software itself. Containerized software, no matter how reputable the repository, is quite complex and poses risks. It's critical that those who build, buy, use, and maintain the container software can inventory and understand the scope and scale of their container images.

netrise.io | sales@netrise.io

2 | You Can't Rely on Container Manifests - New Visibility Methods are Needed

- We find that 12.4% (or 1 in 8) of the 27,261 components have no software manifest - they are "manifestless". This means these components lack the formal metadata typically found in manifests and that they don't include details about dependencies, version numbers, or the source of the package.

- The presence of these manifestless components poses a significant challenge for software visibility, security, and compliance. This also means that common / traditional container scanning tools that rely on manifests for analysis, will have big visibility gaps when reporting results unbeknownst to the user. This suggests a need for new processes and tooling required for tracking and auditing containerized software and to properly mitigate the associated risks.

3 | Container Risks Are Much Higher Than Commonly Understood

- We find that the average container has 604 known vulnerabilities in the underlying software components, with over 45% of them being more than 2 years old and some even more than 10 years old. Further, of the 16,557 identified CVEs that had a Critical or High CVSS Severity ranking, 691 (or 4.2%) were found to be weaponized vulnerabilities per NetRise's threat intelligence. "Weaponized vulnerabilities" as a category include vulnerabilities present in the CISA KEV catalog, those known to be used by botnets, to spread ransomware, used by threat actors, or used in known attacks.

- In addition, another cybersecurity concern are some of the non-CVE risks. We found 4.8 misconfigurations per container including 146 "world writable and readable directories outside tmp". Lastly, the containers had overly permissive identity controls with, on average, 19.5 usernames per container.

Introduction and Purpose

Introduction - Containers are the Weakest Link

Containers are the fastest growing - and weakest cybersecurity link - in software supply chains.

All companies rely on software to power their business, to connect with customers and partners, to automate back-office processes, and to build market presence. Today's world is built on software – 3rd party software, open source software, in-house developed software, operating system software, applications, containers, and device firmware to name a few.

Today, much of that application software exists in the form of containers. Containers have revolutionized the way we develop, deploy, and manage applications. In less than a decade, containers have grown from a lightweight virtualization technology, to the standard for software distribution, to a powerful underlying platform for complex and distributed applications. Containers have driven the growth of cloud-native technologies, fundamentally changing how many modern applications are designed, deployed, and managed.

In fact, a recent VentureBeat article states that "Gartner predicts that by 2029, more than 95% of enterprises will be running containerized applications in production, a major jump from less than 50% last year. In five years, 35% of all enterprise applications will run in containers, and more than 80% of commercial off-the-shelf (COTS) vendors will offer their software in container formats, up from less than 30% last year."[1]

The article goes on to say that containers and their orchestration platforms are dominating DevOps and DevSecOps across enterprises creating cloud apps, and it's going to accelerate.

However, this increasing reliance on containerized applications comes with 2 key cybersecurity challenges:

1. The need to maintain **visibility** of the detailed software components in containers and their provenance.

2. The need to identify and prioritize **vulnerabilities and risks** within the containers' components.

Cyber adversaries know there is a growing container software supply chain visibility and risk challenge. In fact, software supply chain attacks have seen triple-digit increases, but far too few organizations have taken steps to evaluate the risks of these complex attacks in their software supply chain.

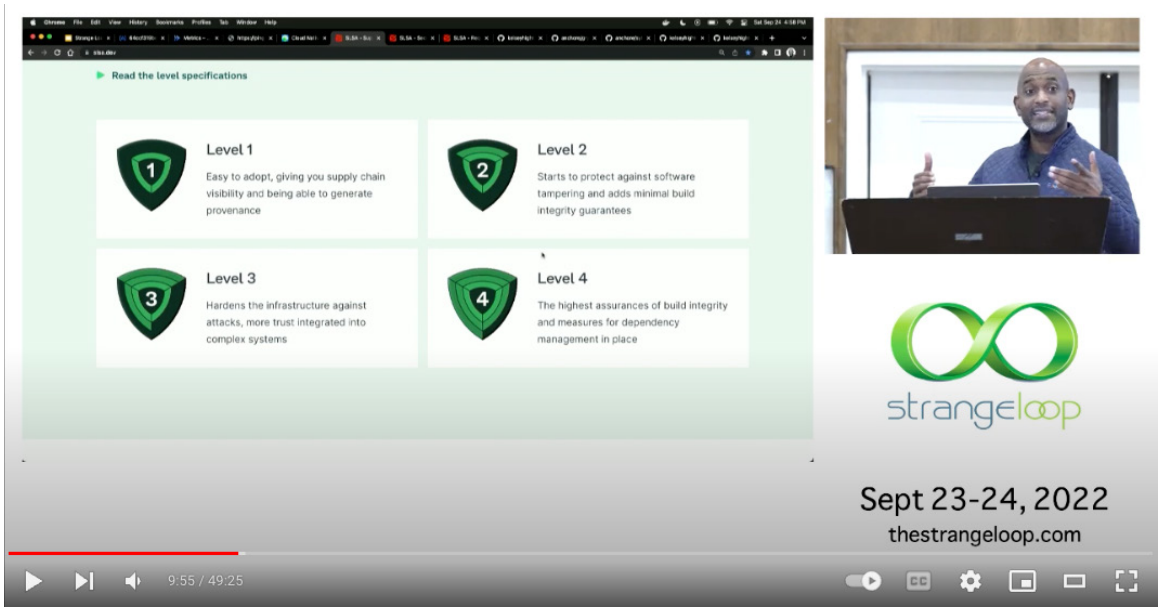
> *According to Capterra's "2023 Software Supply Chain Survey", 61% of companies have been impacted by a software supply chain cyber attack in the 12 months prior to the survey.*[2]

Many companies assume that the containerized software they acquire, catalog, build, deploy, and run is secure and free from vulnerabilities and risks, but recent high-profile software supply chain breaches have proven otherwise. The reality is that every piece of software, no matter how trusted the source or the repository poses risks. This is where the principle of "trust but verify" becomes crucial. Blind trust in software can lead to devastating consequences, from data breaches to operational disruptions.

> *Comprehensive visibility into all containerized software components and dependencies is an essential starting point for software supply chain detection and response.*

In fact, Kelsey Hightower from Google presents a fantastic comparison that illustrates the point - he compares today's practice of pulling code from GitHub repositories to the unimaginable act of plugging a random USB key discovered at a coffee shop into your laptop.[3]

His point - visibility matters.



"A lot of you are like, he is crazy; he is picking up a random USB key, sticking it in his laptop and sending it to production. Y'all act like y'all don't have GitHub accounts! [Laughter] Y'all are literally doing this every day! And it's actually becoming a critical problem."

Containerized applications are one of the weakest links in software supply chains. Companies are struggling to get container security right. Issues from misconfigured clouds, containers, and networks; to uncertainty over container security throughout the software's lifecycle persists. And cyber attackers, insiders, and nation-state actors are capitalizing on the disconnects by exploiting growing vulnerabilities in container images, runtimes, API interfaces and container registries. Unsecured containers with light identity security, if any at all, are a goldmine for insider attackers, too.

Purpose

The purpose of this NetRise "Software Supply Chain Research Study" is to get beyond the marketing reports and state of the market reports, and look at actual software compositions, vulnerability risks, and non-CVE risks that exist in different asset classes that are in every business's software supply chain.

The objective is to educate and inspire CISOs, security professionals, and procurement teams to understand the scope and scale of software and it's risks that likely exist within their software supply chains and to take proactive steps to securing their supply chains.

Scope and Methodology

Research Scope

Containers are the fastest growing - and weakest cybersecurity link - in software supply chains. With that backdrop, this NetRise “Software Supply Chain Risk Study” looks at 70 randomly selected container images from of the 250 most commonly downloaded images on Docker Hub.

And these enterprise class container images represent a vital component of enterprise software supply chains whether developed and used internally or acquired. And vulnerabilities within these container images can have far-reaching consequences. The scope of this research includes a comprehensive analysis of the software embedded in these containers leveraging the leading compiled code analysis capabilities of the NetRise Platform.

By analyzing the software and reporting on the vulnerabilities and risks associated with common container images, this report hopes to underscore the urgent need to prioritize software supply chain security.

While containers have revolutionized the way we develop, deploy, and manage software applications, when these container images aren’t secure, attackers can quickly move beyond the initial intrusion and breach entire networks and infrastructures.

And according to the Achore Software Supply Chain Security Report, the scale of the problem continues to grow as 88% of enterprises plan to continue expanding their container adoption over the next 24 months with 31% planning to increase use significantly. [4]

Research Methodology

The research methodology employed for this report is designed to provide a detailed and holistic understanding of the software components and software risks associated with each containerized application. The following steps outline the research process:

1. **Software Bill of Materials (SBOM) Analysis:**
   Objective: Gain complete visibility into the components that constitute the software running within each container.

Process: Use the NetRise Platform to generate detailed SBOMs for each container. This involves identifying all software components, including third-party libraries and dependencies (both direct and indirect), to understand the complete software stack.

2. **Vulnerability and Non-CVE Risk Assessment:**
   Objective: Evaluate the risk state of each container, considering both known vulnerabilities (CVEs) and non-CVE risks.
   Process: Use the NetRise Platform to identify vulnerabilities listed in the CVE database, and non-CVE risks, such as misconfigurations, outdated components, and potential security flaws that are not yet publicly disclosed.

3. **Evaluate the Priority of Identified Vulnerabilities:**
   Objective: Stratify vulnerabilities based on CVSS scores, weaponization, and network accessibility.
   Process: Use the NetRise Platform to identify weaponized vulnerabilities that are actively being exploited in the wild, and those that are also network accessible to narrow the list of priority vulnerabilities.

Current State of the Market

Market Research and Statistics

Security teams struggle to respond to vulnerabilities, especially where that vulnerability is included within embedded software dependencies. Because software components have not been traditionally disclosed, their content is often opaque to teams trying to ascertain whether they are affected. This requires extraordinary work to identify affected software and implement risk mitigations.

According to a recent Ponemon study, only 29% of organizations are conducting post-build software dependency/artifact analysis to prevent malicious packages from impacting the software they build, buy, or use.[5] And only 38% of respondents say budget and staffing dedicated to securing the software supply chain is sufficient or very sufficient.[5]

These software supply chain vulnerabilities and non-CVE risks extend to containerized software as well. In the “2024 Kubernetes Benchmark Report: The Latest Analysis of Kubernetes Workloads” published by the Cloud Native Computing Foundation, 28% of organizations have more than 90% of workloads running in insecure Kubernetes configurations. The majority of workloads, more than 71%, are running with root access, increasing the probability of system compromises and sensitive data being exposed.[6]

The lack of transparency and trust within the global software supply chain has emerged as a critical issue for organizations - and containerized software is no different. It's well understood that modern application development increasingly relies on open-source and, sometimes, commercially licensed third-party libraries. Thus, transparency into the contents of containers is essential to properly evaluate and vet the contents of the software against organizational standards for supply chain and operational risks. A failure to do so leaves organizations open to:

- **The presence of unknown software risks** in the form of unremediated and/or unmitigated vulnerabilities within the organization's software supply chain.
- **Potential legal risks** resulting from onerous or unattractive licensing terms and conditions associated with dependencies that may be inherited by the ultimate end user of the application.
- **Operational and supply chain risks** including factors such as the presence of significant technical debt in licensed software or software lacking appropriate security controls and checks.

The transparency required to effectively address and avoid such issues starts with the SBOM. In its most basic form, and like other bills of materials, SBOMs list the individual software components — open source, commercial, and (in some cases) proprietary — utilized in the creation of a containerized application. But while SBOMs are considered a best practice and critical to having a secure software supply chain, only 35% of respondents say their organizations produce or generate SBOMs.[5]

And for containerized software, only 39% of mature container users are currently creating SBOMs for the software they build and only 30% are using SBOMs for open source software they use according to a survey conducted by Anchore.[4]

Lastly, a detailed understanding of the software within an organization, can be critical to timely cyber attack investigation, response, and remediation. But only 38% of organizations say they are very or highly effective in detecting and responding to an attack on a software vulnerability. And almost half (47%) say it takes at least a month to more than 6 months to respond to a critical software vulnerability.[5]

Market Trends

The industry is however, beginning to make progress in software supply chain security and risk management including software that is containerized. This progress is being driven by several factors, including:

Containers are Higher Risk Than Traditional Apps

Containers have massively changed software infrastructure for many accompanied by a major increase in overall complexity (although much of it is hidden from developers thanks to multiple layers of abstraction and convenient tools). The increase in complexity has introduced numerous new risks and security challenges and new skills are needed to mitigate them efficiently.

These factors show up in a survey conducted by Anchore in which intermediate and advanced container users see containers as having higher risk than traditional applications (31 percent vs 23 percent).[4] As users leverage containers more, they may be more likely to understand risks such as typosquatting and dependency hijacking that can impact containers.

Increasing Software Supply Chain Cyber Threats

Containerized software, much like other software and firmware have become prime targets for cyber attackers. As noted earlier, containers are ripe targets for cyber attackers, insiders, and nation-states alike. And attackers are exploiting vulnerabilities in the DevOps process at an alarming rate, often using them as entry points for broader compromises.

Regulatory and Compliance Pressures

Governments and regulatory bodies are implementing stricter regulations to ensure the security of software supply chains. Compliance with standards such as the White House issued Executive Order 14028 mandating the presentation of a SBOM, and the European Union's Cyber Resilience Act (CRA) is becoming mandatory for many organizations. There is also a growing emphasis on the use of SBOMs to enhance transparency and security.

Technological Advancements

Organizations are increasingly adopting advanced software supply chain analysis and security tools for advanced risk management programs. These security tools provide:

- Detailed SBOM development for all software including embedded firmware, operating systems, virtualization software, containers, and applications.
- Detection of vulnerabilities and non-CVE risks associated with all of the SBOM software components in use.
- Prioritization of all identified software supply chain risks.

Information from these tools can then enrich and feed asset discovery and management tools and intrusion detection tools used within security operations.

Software Bills of Materials Analysis for Containers

Below we look at a summary of the software analysis for the 70 Docker Hub container images analyzed. We look at the number of software components per container, the number of manifestless components, and the number of different versions of some of the most common components in use.

Average Number of Software Components per Container

For the 70 images analyzed, each container had on average 389 software components.

Most Commonly Occurring Software Components

For the 70 container images analyzed, there were a total of 27,261 software components.

Excluding the Linux kernel software, the most common components based on how many of the 70 containers they were found in, are listed in the table below. In addition, we show how many different versions of each were found.

Component Name	Number of Containers it's Found In	Percentage	Unique Component Versions
openssl	69	98.6%	7
zlib	68	97.1%	6
bash	51	72.9%	10
prc-tools-arm	47	67.1%	1
coreutils	42	60.0%	6
findutils	42	60.0%	4
gzip	42	60.0%	3
glibc	41	58.6%	1
openldap	41	58.6%	1
gnutls	38	54.3%	1

Distribution of Component Types

For the 70 images analyzed, there were 27,261 software components found or an average of 389 components per container image. And if we look at unique versions of the components, then we find that there are a total of 2,992 different component name/version combinations in the 70 container images. Below we break out these total components by type.

Components by Type	Total Component Count	Unique Component Count
Maven	11,026	1,845
OS	9,951	557
Manifestless	3,390	313
NPM	1,809	186
PYPI	918	64
Golang	147	25
Windows	20	2
Total	**27,261**	**2,992**
Average / Container	**389**	**43**

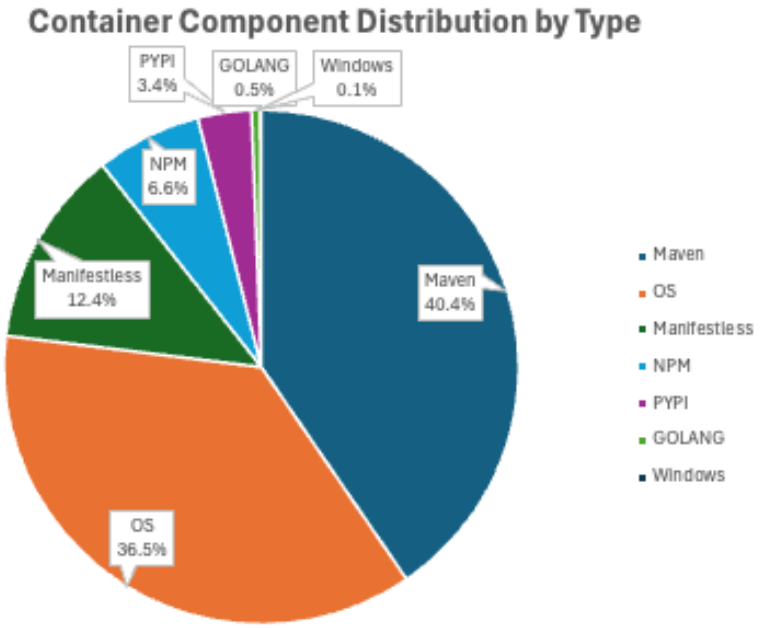
Manifestless Container Components

In looking at the total container components found by type, we see that 3,390 of the 27,261 components are manifestless, or 12.4%. This means 1 in 8 of all container components are lacking the formal metadata typically found in packages with manifests. This means they don't include details about dependencies, version numbers, or the source of the package.

What's worse, the vast majority of container scanning tools rely on the manifest information to provide visibility into what exists in the container. Without it, both the scanning tools and software users are blind to what's included.

The presence of such a large percentage of manifestless components poses a significant challenge

for visibility, security, and compliance. This suggests a need for enhanced tracking and auditing processes to mitigate associated risks, as well as possibly reconsidering the sources of these components to reduce reliance on components without manifests.



Average Component Dependencies

For the 70 images analyzed and the different types of components found above, we analyzed the direct and indirect dependencies for each component.

Direct dependencies are libraries or packages that a component directly relies on to function. This average provides insight into how complex each component type is in terms of its immediate dependencies.

Indirect dependencies are packages or libraries that a component relies on through other dependencies. Higher indirect dependencies can indicate complex dependency chains, which may increase the potential for security vulnerabilities or version conflicts.

Component by Type	Average Direct Dependencies	Average Indirect Dependencies
Maven	3.45	3.05
OS	2.67	93.15
Manifestless	7.00	15.49
NPM	2.06	11.46
PYPI	0.97	0.48
GOLANG	0.80	-
Windows	-	-

CVE-Based Vulnerability Risk Assessment Analysis

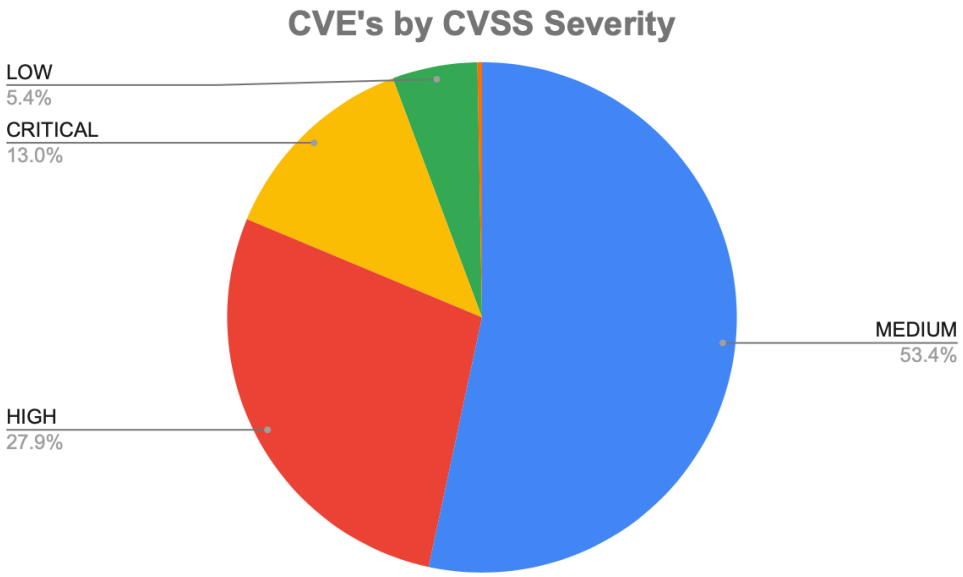
Below we look at a summary of the vulnerability analysis for the 70 container images analyzed. We look at the number of CVEs per container, the CVE CVSS scores, the number of weaponized vulnerabilities, and the age of different CVEs.

Average Number of CVEs per Container

For the 70 container images analyzed, each container had on average 604 CVEs. There were 42,247 total CVEs found in the 70 container images.

CVEs by CVSS Severity

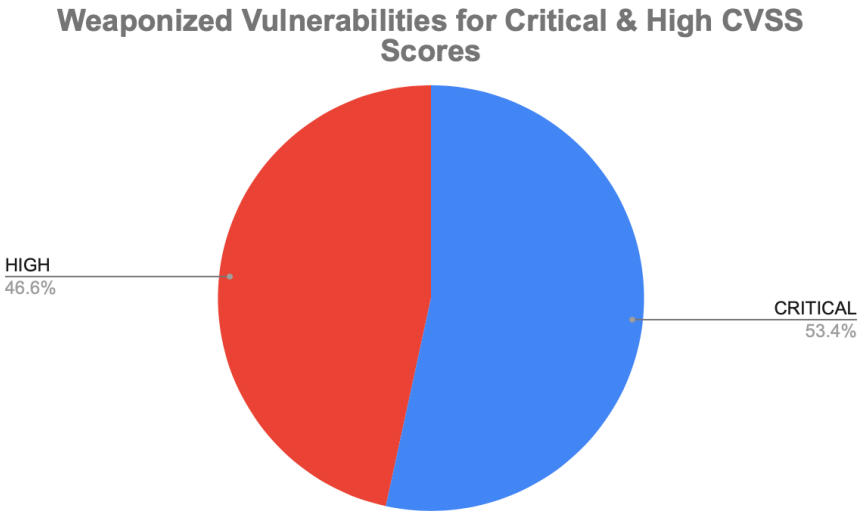
Of the 42,247 uniquely identified CVEs, 40.9% ranked Critical or High per the CVSS Severity scores.



Weaponized Vulnerabilities (Total and by CVSS Severity)

Of the 16,557 identified CVEs that had a Critical or High CVSS Severity ranking, 691 (or 4.2%) were found to be weaponized vulnerabilities per NetRise's threat intelligence.

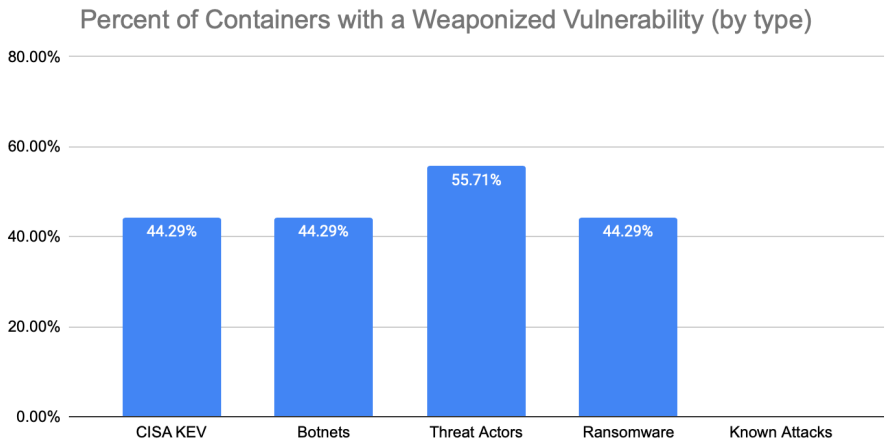
“Weaponized vulnerabilities” as a category include vulnerabilities present in the CISA KEV catalog, those known to be used by botnets, to spread ransomware, used by threat actors, or used in known attacks.



Containers with Weaponized Vulnerabilities

Forty four percent (44%) of the 70 containers analyzed had at least one vulnerability that is on the CISA KEV, had vulnerabilities used by botnets, or had vulnerabilities known to be used for spreading ransomware. Additionally, 55% had vulnerabilities used by known threat actors.

The weaponized vulnerability metric is an important threat source because it can be used to identify what known exploitable vulnerabilities exist within the container or enterprise environment which provides a focused list on where to prioritize remediation efforts.



Unique Vulnerabilities by EPSS Score

We next look at the EPSS Score for each vulnerability. The EPSS allows companies to prioritize the most pressing vulnerabilities with threat actor information and a probabilistic understanding of threats. The EPSS, developed by the Forum of Incident Response and Security Teams (FIRST) leverages current, real-world exploit and threat information from many sources to estimate the probability of a vulnerability being exploited in the next 30 days. [7]

Be aware that EPSS Scores are temporal meaning they can change over time. So the data below represents a snapshot of the data from the day the research was generated.

Below we categorize the CVEs by the EPSS Score for the 42,247 CVEs in the 70 container images.

EPSS Score	Unique Vulnerability Count
above .90	230
between .80 and .90	96
between .60 and .80	0
between .20 and .60	145
below .20	41776

Total Vulnerabilities Categorized by Age

If we look at the same 42,247 CVEs in the 70 container images, we find that 45.1% of the CVEs are over 2 years old and 7.9% over 5 years old. Below we categorize the CVEs by age.

Age	CVE Count
10+ years old	527
5-10 years old	2,792
2-5 years old	15,721
1-2 years old	18,370
6 months to 1 year old	4,827
3-6 months old	9

Non-CVE Based Risk Assessment Analysis

Below we look at a summary of the non-CVE risk analysis for the 70 container images analyzed.

Average Number of Misconfigurations per Container

For the 70 container images analyzed, each container image had on average 4.8 misconfigurations.

Most Common Misconfigurations

The most common misconfigurations based on type and how many occurrences were found in the 70 container images analyzed, are listed in the table below

Misconfiguration Type	Occurrences
World writable and readable directories outside tmp	146
Weak hash algorithms found	101
Insecure URL	54
One or more compilers exist	46
Services Without Configuration Files	43

Average Unique Usernames per Container

For the 70 container images analyzed, each container image had on average 19.5 unique usernames. This suggests a potentially high-risk environment with a broad attack surface and increased complexity in managing access and credentials.

Summary

Today's world is built on software – 3rd party software, open source software, in-house developed software, operating system software, applications, containers, and device firmware to name a few. Today, much of that application software exists in the form of containers which have revolutionized the way we develop, deploy, and manage applications. In this research we looked at a detailed analysis of the software in 70 of the most common containers from Docker Hub.

Software Bills of Materials Analysis: Using the detailed software analysis in the NetRise Platform, we find that the SBOM for the average container is more complex than many might believe and contains 389 software components.

Vulnerability Risks Analysis based on a Detailed Software Analysis Approach: Using the detailed software analysis in the NetRise Platform, we find that the average container has 604 known vulnerabilities with 40.9% of these ranked Critical or High CVSS Severity. Further, over 4% of these Critical and High CVEs are considered weaponized based on NetRise threat intelligence.

Analysis Methodology and Approach: First, every piece of software, no matter how reputable the source or the repository, is quite complex and poses risks. Second, it's critical that those that build, buy, use, and maintain the software can inventory and understand the scope and scale of their software, and the associated risks. We believe a deep analysis of the software using a compiled and interpreted code analysis is the only way to get to this information.

End Notes

1. "10 reasons why securing software supply chains needs to start with containers", VentureBeat, January 29, 2024.
2. Three in Five Businesses Affected by Software Supply Chain Attacks in Last 12 Months, Gartner/Capterra, May 11, 2023.
3. "The Secure Software Supply Chain", by Kelsey Hightower, Strange Loop 2022.
4. "2022 Software Supply Chain Security Report", Anchore, 2022.
5. The State of Software Supply Chain Security Risks, Prepared by Ponemon Institute, Sponsored by Synopsis, May 2024
6. "2024 Kubernetes Benchmark Report: The Latest Analysis of Kubernetes Workloads", Cloud Native Computing Foundation, January 26, 2024.
7. "Using EPSS to Modernize Vulnerability Prioritization", NetRise Blog.

Glossary of Terms

CISA KEV - The Cybersecurity and Infrastructure Security Agency's (CISA) Known Exploited Vulnerabilities (KEV) Catalog is a compilation of documented security vulnerabilities that have been successfully exploited, as well as vulnerabilities associated with ransomware campaigns.

CISO - Chief Information Security Officer.

CVE - The Common Vulnerabilities and Exposures (CVE) system provides a reference method for publicly known information-security vulnerabilities and exposures.

CVSS - The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities.

EPSS - The Exploit Prediction Scoring System (EPSS) is a data-driven effort estimating the likelihood (probability) that a software vulnerability will be exploited in the wild.

IoT - The Internet of things (IoT) describes devices with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. The Internet of things encompasses electronics, communication, and computer science engineering.

IT - Information Technology.

NVD - The National Vulnerability Database (NVD) is the U.S. government repository of standards-based vulnerability management data.

SBOM - A "software bill of materials" (SBOM) has emerged as a key building block in software security and software supply chain risk management. An SBOM is a nested inventory, a list of ingredients that make up software components.

XIoT - The extended internet of things (XIoT) is an umbrella term that includes all internet of things (IoT) or physical devices connected to the internet. It encompasses networking equipment, IoT, operational technology (OT), internet of medical things (IoMT), industrial IoT (IIoT), and supervisory control and data acquisition (SCADA).