Guide

Blast Radius Mitigation: Everything You Need to Know

uptycs

Table of Contents

Overview

Increasingly, savvy cloud security professionals are focused on minimizing the impact of potential breaches. This is where Blast Radius Mitigation comes into play—a comprehensive approach designed to contain and neutralize threats before they can escalate into a full-blown crisis.

An effective blast radius mitigation framework delivers a step-by-step strategy to ensure robust security across your infrastructure. The five key steps are:

1. **Threat Detection in Workloads:**
   Identify subtle anomalies and evolving threats in dynamic environments.

2. **Unified Risk Assessment and Attribution:**
   Understand the potential impact of each threat and prioritize responses based on risk.

3. **Real-Time Containment Through Policy Enforcement:**
   Stop threats in their tracks to minimize damage.

4. **Root Cause Analysis and Image Provenance:**
   Trace vulnerabilities back to their source to prevent recurrence.

5. **DevSecOps Guardrails and Risk Mitigation:**
   Embed proactive safeguards into CI/CD pipelines for long-term security.

In this guide, we'll walk you through everything you need to know about mitigating blast radius risks, breaking down complex threats, and empowering your teams to respond with confidence. Let's dive into the framework that is transforming cloud security, one layer at a time.

01 Detecting Anomalies to Strengthen Blast Radius Mitigation

The rise of cloud-native applications, containerized environments, and dynamic infrastructure has transformed how organizations deploy and manage workloads. While this evolution accelerates innovation, it also introduces new security challenges. Traditional signature-based detection methods, which rely on pre-defined behavioral patterns or known indicators of compromise (IoCs), often fail to keep pace with novel threats targeting modern systems.

Uptycs' anomaly detection capabilities address these challenges by leveraging eBPF-powered telemetry and machine learning (ML) to detect deviations in expected behavior across containers, hosts, and Kubernetes environments. This blog explores the limitations of traditional detection methods, highlights real-world attack scenarios, and shows how anomaly detection delivers actionable insights for securing dynamic workloads.

Why Signature-Based Detections Fall Short

Signature-based detection has been a cornerstone of cybersecurity, but its reliance on known patterns makes it ineffective against evolving threats. Today's adversaries exploit this limitation in environments that span containers, hosts, and Kubernetes orchestration layers. Here are some scenarios where behavioral signatures fall short:

1. Cryptojacking Campaigns in AI and GPU Workloads

- **Example:** Attackers infiltrate Kubernetes clusters, deploying containers that exploit GPU resources for cryptocurrency mining. These malicious workloads mimic legitimate AI/ML processes, leveraging GPUs for cryptomining while blending into baseline activity.
- **Challenge:** Traditional tools fail to differentiate between malicious and legitimate workloads, as both exhibit similar resource consumption patterns.

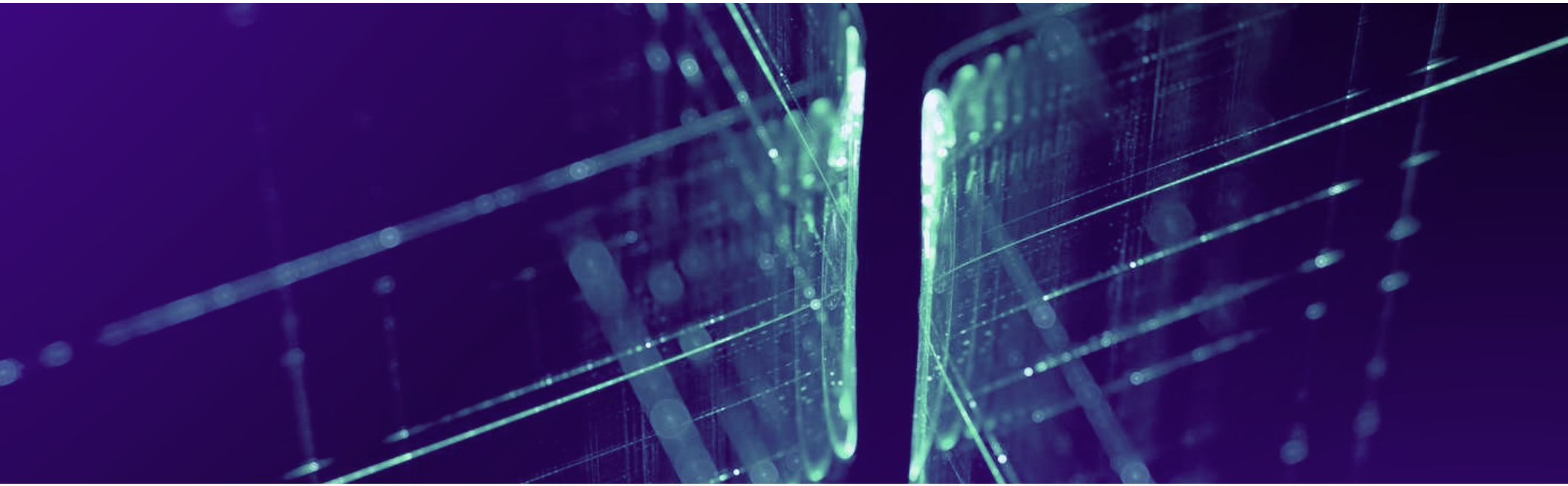
2. Supply Chain Attacks via Compromised Images

- **Example:** A public container image is compromised, embedding a payload that communicates with an external command-and-control (C2) server. The payload executes stealthy actions like downloading scripts or manipulating network traffic.
- **Challenge:** Signature-based detections miss these payloads if the attack leverages novel techniques not previously cataloged.

3. Fileless Malware on Hosts and Containers

- **Example:** Fileless malware executes on a host or container by abusing legitimate system tools like bash, curl, or PowerShell. These tools perform lateral movement, credential theft, or data exfiltration.
- **Challenge:** With no files to monitor or modify, signature-based methods cannot detect these attacks.

These examples highlight the need for adaptive defenses like anomaly detection that monitor deviations in behavior across systems, providing visibility into containerized workloads, host activity, and Kubernetes orchestration layers.



Anomaly Detection in Action

Anomaly detection focuses on identifying deviations from established baselines of normal behavior, making it ideal for dynamic and distributed environments. Whether it's detecting unusual process execution on a host or spotting anomalous resource consumption in a container, anomaly detection provides visibility into threats that evade traditional methods.

Examples Across Hosts and Containers

1. Unusual Process Execution

- **Scenario:** A container running a Node.js application suddenly spawns a curl process that attempts to download a binary from an unknown URL.
- **Detection:**
  - The application container typically doesn't execute curl.
  - The downloaded binary doesn't match any legitimate files previously seen in the environment.
- **Response:** Alerts the security team, blocks the download, and captures telemetry for investigation.

2. Anomalous Network Activity

- **Scenario:** A Kubernetes pod labeled billing-service initiates an outbound connection to an unknown IP address in an unfamiliar region.
- **Detection:**
  - The pod usually communicates only with internal APIs and payment gateways.
  - The unknown IP is outside the organization's trusted infrastructure.
- **Response:** Isolates the host, preventing lateral movement while IR teams investigate.

3. Host-Level Anomalies

- **Scenario:** A host typically running static workloads experiences a spike in CPU usage, with unusual processes scanning internal network ports.
- **Detection:**
  - eBPF telemetry identifies anomalous process activity and network scans originating from the host.
  - The ML model flags this behavior as a deviation from the host's baseline.

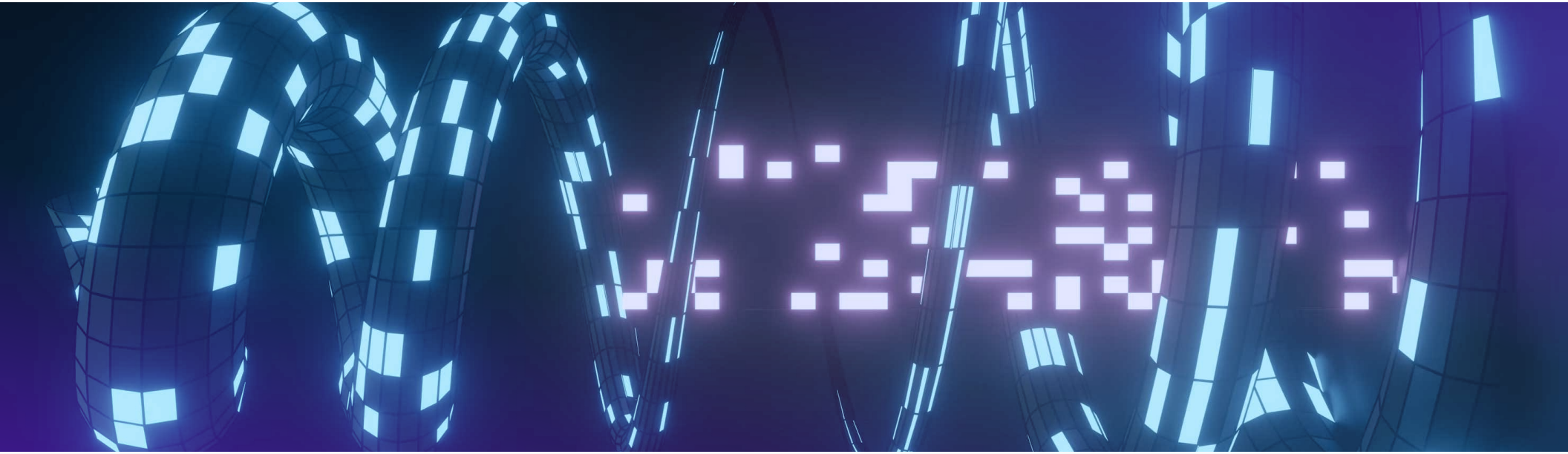
- **Response:** Isolates the host, preventing lateral movement while IR teams investigate.

4. Container Resource Abuse

- **Scenario:** A Kubernetes pod labeled billing-service initiates an outbound connection to an unknown IP address in an unfamiliar region.
- **Detection:**
  - The pod usually communicates only with internal APIs and payment gateways.
  - The unknown IP is outside the organization's trusted infrastructure.
- **Response:** Isolates the host, preventing lateral movement while IR teams investigate.

5. Unexpected File Activity

- **Scenario:** A container or host begins accessing sensitive configuration files in /etc, a behavior it has never exhibited before.
- **Detection:** Unauthorized file access is flagged as anomalous, with telemetry captured at the kernel level using eBPF.
- **Response:** Blocks access attempts, generates detailed telemetry, and alerts the security team.



How Uptycs Detects Anomalies Across Hosts and Containers

Uptycs leverages **eBPF** for telemetry collection and **ML models** for adaptive baselining, enabling granular insights into behavior. Here's how it works:

1. Deep Telemetry Collection

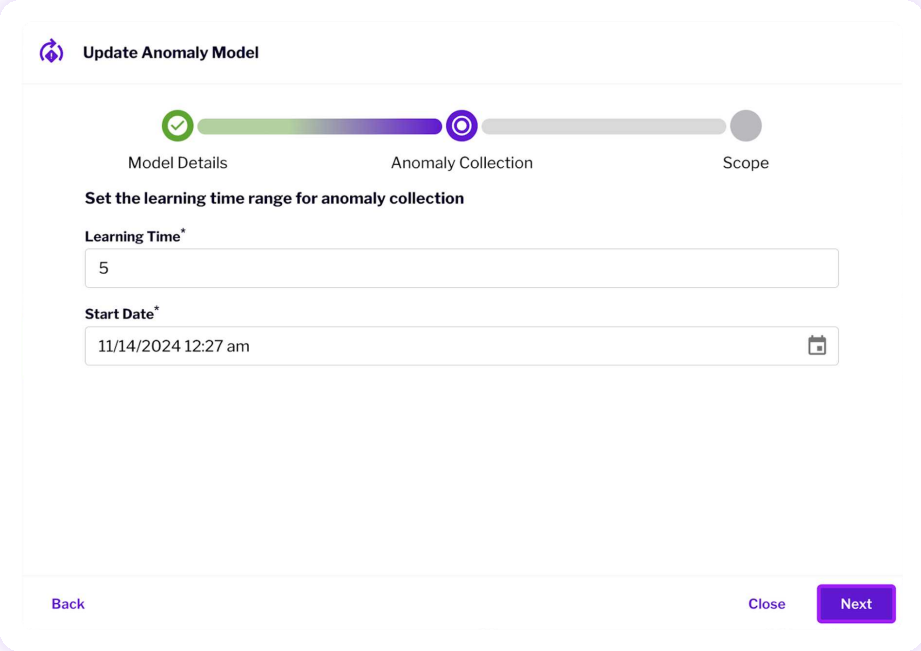
- **Processes:** Tracks process creation, execution, and relationships across containers and hosts.
- **Sockets:** Monitors network connections, identifying suspicious outbound or lateral movement.
- **File Activity:** Detects unauthorized file reads, writes, or deletions.
- **GPU Metrics:** Captures deviations in GPU usage to detect cryptojacking and resource abuse.

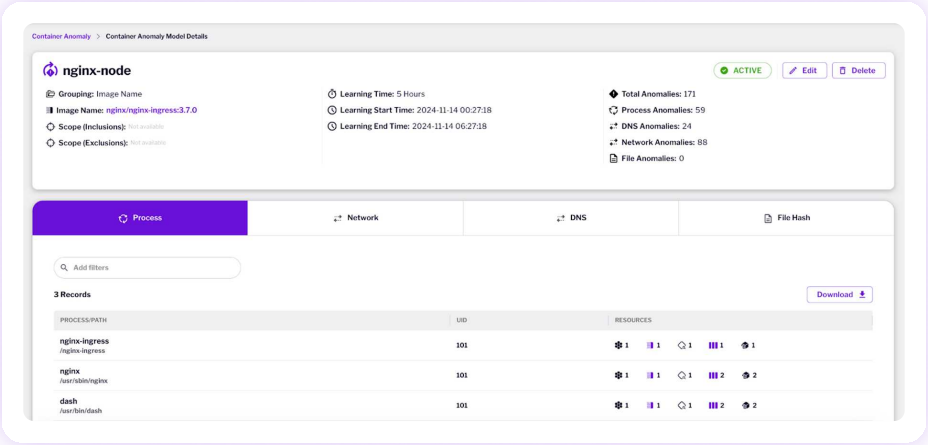
2. Context-Aware Baselines

- **Container Image and Digest:** Unique baselines for immutable images and digests.
- **Host-Level Grouping:** Adaptive models tailored to individual hosts, allowing flexibility for unique workloads.

3. Anomaly Scoring and Detection

- Uptycs' ML models continuously analyze telemetry to establish baselines and detect deviations.
- Alerts are enriched with context, allowing IR teams to prioritize critical threats.





How Anomaly Detection Empowers Incident Response

Anomaly detection goes beyond raising alerts—it provides actionable intelligence for rapid response. Uptycs enhances IR workflows by:

1. **Enriching Alerts:** Alerts include detailed telemetry, correlating anomalies with metadata like hostnames, container images, and Kubernetes labels.
2. **Automating Actions:** Automatically throttles or isolates suspicious processes, containers, or hosts while maintaining system uptime.
3. **Facilitating Forensics:** Provides detailed logs, including process trees, file access events, and network activity, for thorough post-incident analysis.

Broadening the Threat Landscape

Anomalies often reveal the early stages of an attack, whether in hosts or containers. Examples include:

- A new, unexpected process launching on a container designed for static workloads.
- Outbound connections from a host to unfamiliar IPs.
- Sudden resource spikes indicative of cryptomining or DDoS participation.
- Unauthorized file modifications or deletions across hosts or containers.

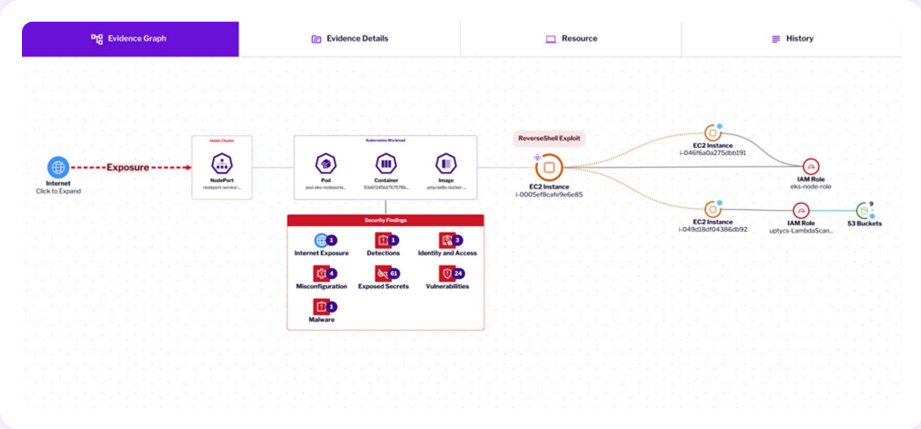
By detecting these deviations in real-time, anomaly detection helps security teams respond quickly, minimizing damage and preventing lateral movement.

02 Unified Kubernetes Risk Insights for Effective Blast Radius Mitigation

Let's talk about the next step, which builds on threat detection by providing critical context to understand each threat's potential impact.

Once a threat is identified, this second step adds essential insights around asset exposure, lateral movement potential, and access to sensitive data, helping teams assess the threat's broader implications. This comprehensive analysis enables security teams to prioritize their response actions, focusing on areas most vulnerable to escalation. In this blog, we dive into the importance of this second step, illustrating how it informs strategic, prioritized decision-making that strengthens organizational defenses.

Organizations have long taken a siloed approach to addressing Kubernetes security risks in their environment through static compliance and misconfiguration scanning or image vulnerability scanning. However, Uptycs intelligently combines insights and presents attack paths to crown jewel assets and presents a unified risk score for every container. This enables security teams to triage and prioritize the most critical risks based on runtime and Kubernetes context that assesses the likelihood of a breach and the potential severity of a breach for every container workload.



The move to Kubernetes across enterprises has been massive, with a recent CNCF survey showing over 95% of organizations using Kubernetes or planning to do so. Kubernetes has its superpowers when it comes to fault-tolerant orchestration and in-built controls inside the cluster for RBAC, governance, networking, and more.

Kubernetes security is crucial as the platform's complexity presents a wide attack surface, often exploited by recent breaches including cryptomining attacks and privilege escalation.

However, with greater control comes greater risk and the complexity of Kubernetes security presents a wide attack surface which has shown to be wildly exploited by recent breaches including the following:

- **Kinsing Malware:** This attack targeted exploitable vulnerabilities and exposed postgres containers for cryptocurrency mining. Once the attacker gained foothold in one part of the environment, they were able to move laterally in other parts of the cluster.

- **RBAC Buster:** This persistence based attack was due to excessive permissions that came from the kube-system namespace where then a new cluster role with admin privileges was created and tied to a service account. Next the attacker was able to do anything - in this case create daemon sets to hijack resources and steal data in the cloud account.

- **WireServing:** This privilege escalation based attack was due to not securing the credentials while node provisioning in AKS. A pod with command execution can access TLS bootstrap tokens of the node. Using these tokens an attacker can steal secrets stored in the cluster.

With these types of attacks, the following is clear: Your CNAPP needs to go a level deeper than traditional agentless/image scanning for Kubernetes security:

- **Runtime Insights:** Without data from runtime, it is extremely hard to be able to prioritize the most critical risks and vulnerabilities in your cluster. A vulnerability that is loaded into memory and internet facing likely deserves more attention than others. Similarly a container that is being exploited for a real-time threat such as a reverse shell definitely deserves attention.

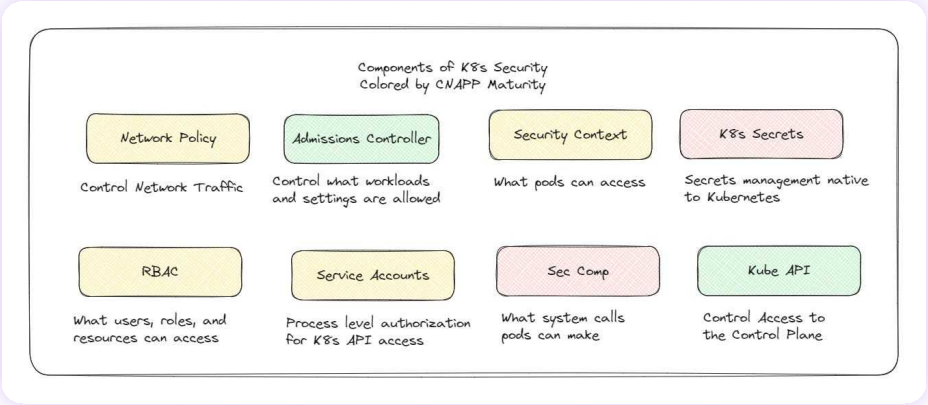
- **Kubernetes security with RBAC:** With identity being the initial attack vector into a cluster, it is important to not only be able to understand cloud permissions but understand the deeper context on what users, service accounts, and groups inside the cluster can access and what permissions they have that can lead to malicious attacks such as privilege escalations and container escapes.

- **Build To Runtime Image Provenance:** With attacks moving from exploiting vulnerabilities and malware to injecting malicious code, an understanding of what malicious code commits went into your runtime deployments and who is responsible is absolutely critical to root cause analysis.

- **Deeper understanding of Kubernetes networking:** The severity of an attack inside a cluster can be controlled via network policies. Similarly, many CNAPPs flag internet exposure without an understanding that the container may be protected because of network policies, which act as a final "firewall". This creates many false positives when it comes to risk prioritization.

- **Admission Controls:** Risk remediation comes down to understanding the profile of the types of insecure deployments that can be blocked through governance based policies.

Fellow analyst James Berhoty summarizes this nicely in terms of Kubernetes security maturity in the CNAPP Market:

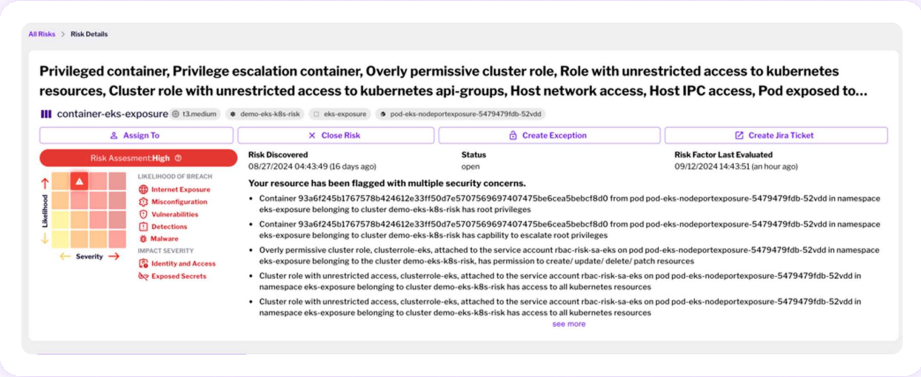


Source: The Kubernetes Gap in CNAPP

Uptycs Kubernetes Risk Prioritization

Risk Calculation

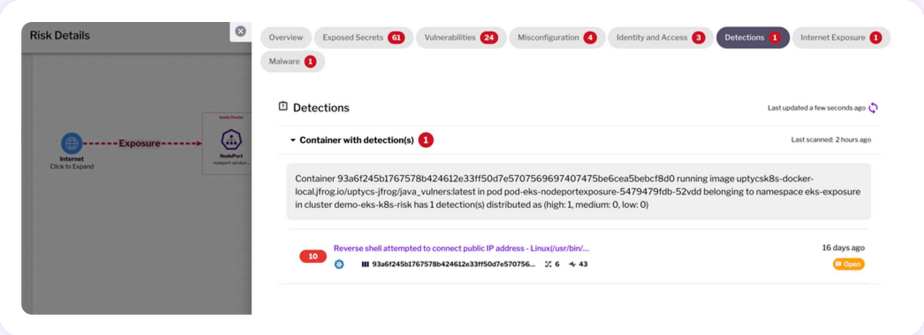
As seen above, factoring all these insights in isolation is extremely hard and creates a siloed approach where key attack paths and critical risks are missed. Uptycs takes the heavy lifting by correlating insights from runtime, Kubernetes control plane, audit logs, and cloud control plane. These are mapped to breach likelihood and severity and turned into a single risk assessment score, which significantly enhances Kubernetes security.



These include the following:

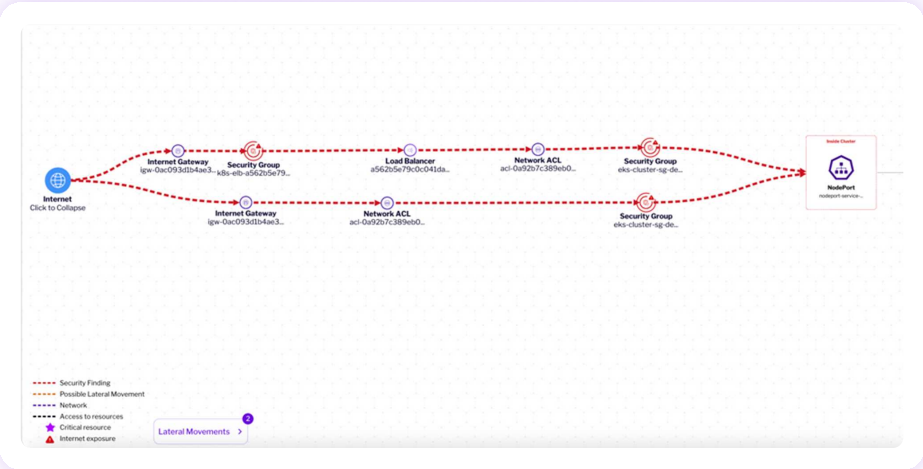
Breach Likelihood:

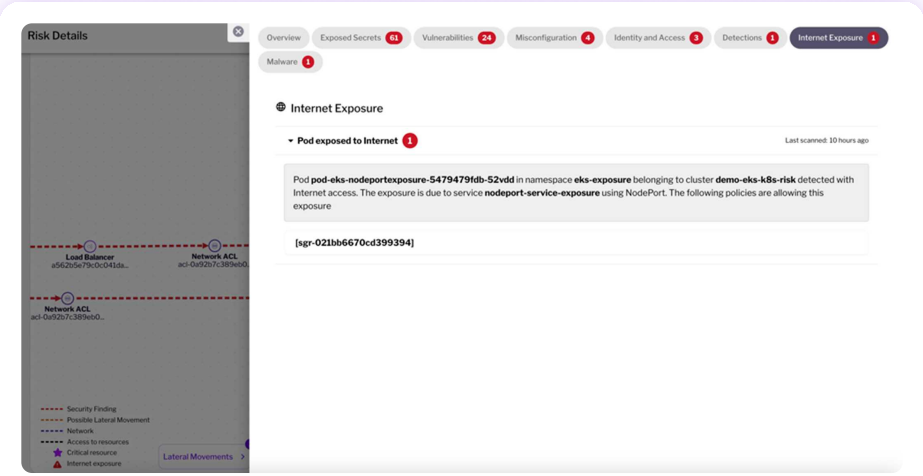
Runtime Threat Detections: such as reverse shells, cryptominers and more with correlation to potentially malicious Kubernetes control plane activity and code commits in the developer pipeline.



Internet Exposure: While many CNAPP solutions look at Kubernetes load balancers, node ports, and services as well as cloud security groups, NACLs, the missing ingredient is looking at Network Policies which act as means of defense against exposure and lateral movements. If your risk engine is not telling you if a network policy is active, you're missing key context.

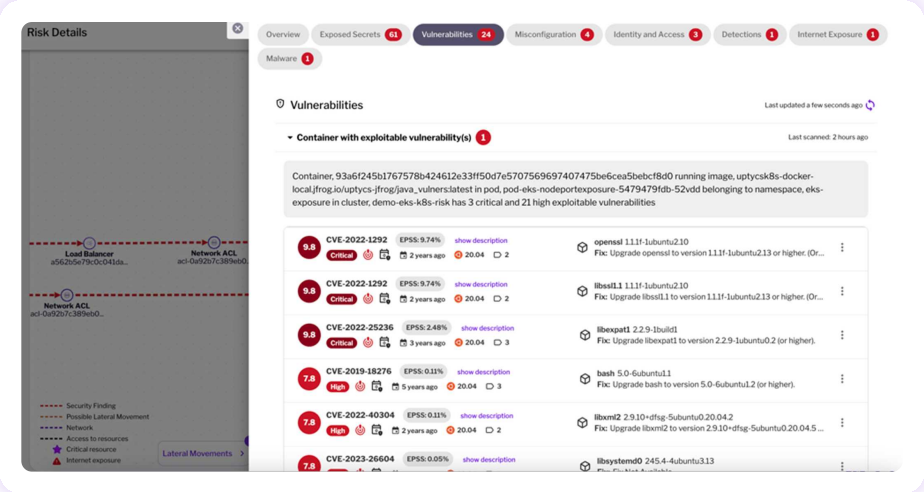
Uptycs not only provides visibility into the full internet exposure path and attack path to crown jewels in the cloud, but integrates with Cilium network policies to show you which network policy is active and whether it is causing internet exposure risk or not.





Pod Security Context/Misconfigurations: If a pod is running as root or has host network access, it has the ability to escalate into the cloud and get access to crown jewels such as S3 buckets with data.

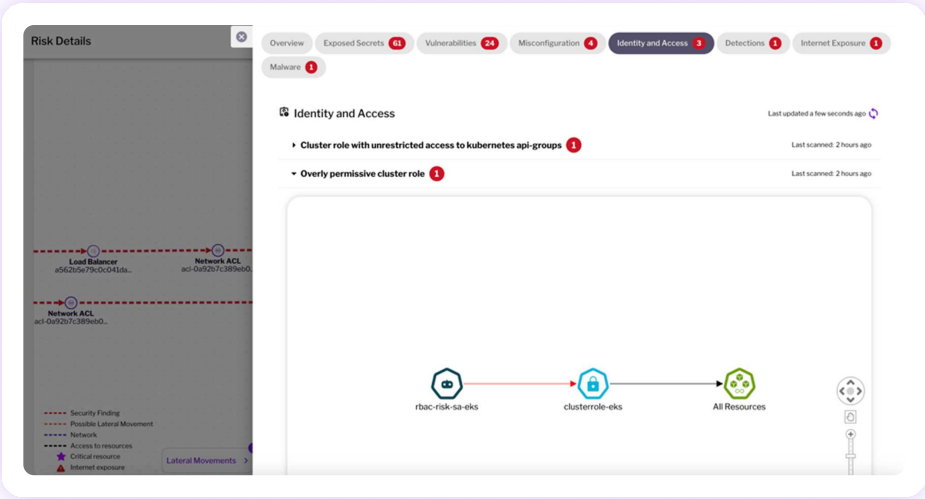
Vulnerabilities: Exploitable vulnerabilities with packages that are in-use by running processes.



Malware: Images detected with cryptominers and ransomware that are deployed.

Impact severity:

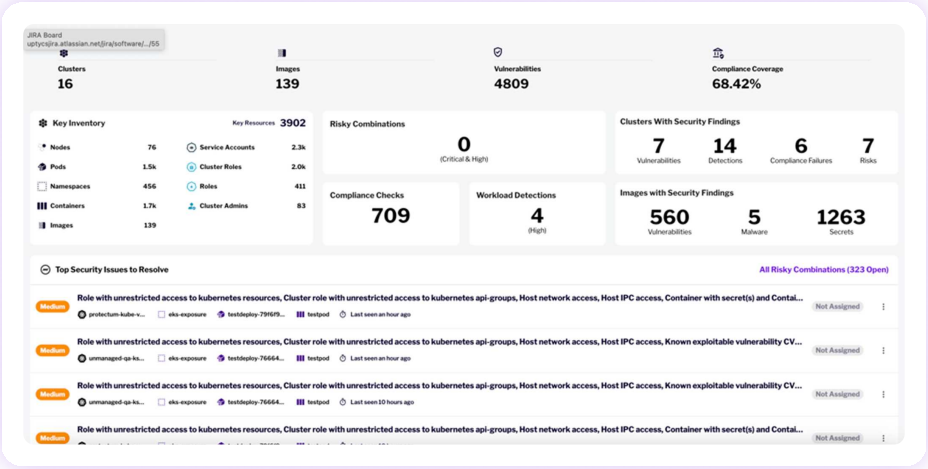
RBAC: With identity being a key attack vector for escalation of privileges and lateral movements, it is extremely important to understand excessive permissions in the cluster. Many clusters also come with default service accounts which are great for dev/test but in production are very risky and when exploited can lead to loss of data and open attack paths.



Exposed Secrets: Credentials that are exposed are more likely to be exploited to access sensitive information and data.

Risk Prioritization

With this unified risk calculation, security teams can prioritize the most critical risks on a daily basis across their container and Kubernetes inventory.



Risk Remediation and Prevention

While this blog focuses on risk prioritization, here is a glimpse of what Uptycs can enable for security teams to mitigate risks in real-time and prevent future threats and risks from happening:

- **Runtime Protection Engine:** Stop malicious threats such as cryptominers, reverse shells and ransomware in real-time before they happen.
- **Container Process Blocking and Remediation:** block malicious processes from spawning or kill a container process if detected against a malicious threat.
- **Kubernetes Deployment Hardening via Admission Controls:** Control what is allowed for deployment in your clusters for any resource and make sure they are compliant/have the right posture.
- **Image Hardening via Policies:** Only allow hardened images to be built and deployed. If images contain exploitable vulnerabilities, malware, secrets, or do not come from trusted sources, fail builds and deployments across the development pipeline to only allow hardened artifacts into runtime. Trace vulnerabilities back to image layers in a Dockerfile to enable developers to fix at the source.

Why Uptycs For Kubernetes Risk Prioritization

	Competition	Uptycs
Runtime Insights	Can't prioritize the most critical risks based on real-time threats or packages in use for vulnerabilities - especially with agentless scans.	Risk is powered by what is active and running in my environment now.
Network Exposure	Does not factor in network policies which acts as a firewall in Kubernetes clusters to prevent lateral movements and internet exposure.	Uptycs combines findings from inside cluster and outside cluster including Cilium Network Policies.
Root Cause Analysis	Limited to data from agentless and image scans, cannot trace back to how the artifact is deployed.	With image provenance, we can trace back malicious runtime risks to actual code commits.
Identity & Access	CSPM only Kubernetes risk engines look at IAM permissions but not risks inside the cluster.	Look at Kubernetes excessive permissions inside cluster as well as risky service accounts, users, groups, and more.

With Unified Risk Assessment and Attribution complete, the next step in the Uptycs Blast Radius Mitigation Framework focuses on rapid threat containment— continue reading to understand each stage of our end-to-end protection approach.

03 Blast Radius Containment through Policy Enforcement

In this section, we focus on Containment through Policy Enforcement, where Uptycs acts to quickly halt the spread of identified threats. With its Protect eBPF sensor, Uptycs enforces security policies at runtime, effectively stopping malicious activities like lateral movement, cryptomining, and ransomware in their tracks.

For effective enforcement at runtime, security systems can incorporate generic behaviors based on the premise that attacks tends to exhibit certain common behaviors:

- Suspicious file or network activity
- Abnormal memory usage
- Unauthorized privilege escalation attempts
- Code execution anomalies

By writing protection policies that focus on these generic behaviors, security systems can identify malware (any type) or exploit activity without needing to know the exact exploit in advance (zero day attacks). Figure (1) provides an example of eBPF based detection and policy based containment of a typical cloud workload.

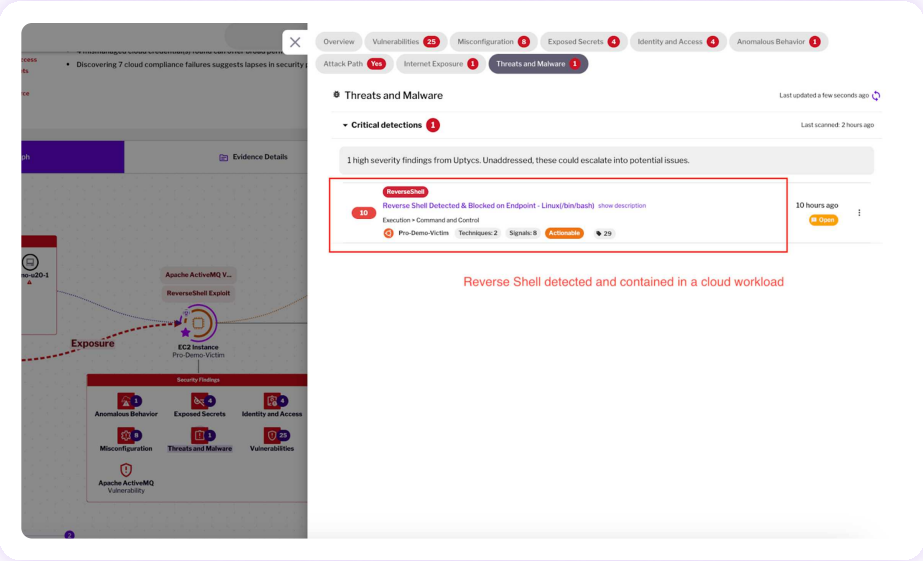


Figure (1) - Attack contained through policy enforcement

We'll detail how Uptycs' Blast Radius Mitigation Framework provides robust, immediate protection during such active threat incidents in your cloud workloads.

1. Malware and exploits

Malware and exploit activity pose continuous risks to cloud workloads. Uptycs' eBPF sensor efficiently detects and contains threats against these key attack types.

- **Exploit**: Exploits target software vulnerabilities in workloads (e.g., Apache servers). Uptycs uses behavioral analysis and signature-based detection to identify exploit attempts, preventing unauthorized access.
- **Privilege Escalation:** Attackers escalate privileges to gain higher access (e.g., from user to admin). Uptycs monitors user behavior and access patterns to detect and block these attempts.
- **Code Injection:** Uptycs detects suspicious code injection activities (e.g., shell injection) that may compromise cloud systems, which are common entry points for attackers.
- **Compromised Credentials:** Uptycs identifies anomalous login attempts (e.g., unusual locations or invalid credentials) to detect and prevent attackers from leveraging stolen credentials to escalate access.

2. Cryptocurrency Mining

Cryptocurrency mining attacks, where attackers hijack a cloud system's resources to mine cryptocurrencies, are on the rise. These attacks are often stealthy, using minimal resources until the system is significantly impacted.

Uptycs can detect mining activity, such as Coinminer processes that consume abnormal CPU and memory usage patterns. By continuously monitoring cloud workload resource usage, Uptycs identifies when processes attempt to mine cryptocurrency, enabling security teams to act and protect cloud resources from being siphoned for malicious purposes.

3. Network Scanning

Network scanning techniques such as IP Scan and Port Scan are commonly used by attackers to discover vulnerable systems, identify open ports, or facilitate lateral movement within a network.

- **IP Scan:** This technique is used by attackers to identify active IP addresses within a given range, enabling them to discover potentially vulnerable systems. Uptycs detects IP scanning activity by monitoring network traffic and identifying patterns indicative of scanning attempts.
- **Port Scan:** Attackers perform port scanning to locate open ports that can be exploited. Uptycs' network monitoring capabilities help detect unauthorized port scanning activities, blocking attackers before they can exploit weaknesses.

These scanning activities are often precursors to larger attacks, such as exploits or ransomware. By detecting behaviors early, Uptycs helps prevent attackers from discovering and exploiting vulnerable cloud systems.

4. Ransomware

Ransomware attacks are highly damaging, as they encrypt valuable files and demand payment for decryption. Uptycs detects ransomware activity by identifying processes that attempt to encrypt files on a system. File encryption is often rapid and widespread, making it a key indicator of ransomware. By monitoring file system changes and process behaviors, Uptycs can detect and block ransomware attacks before they cause irreversible damage to cloud data.

5. Information Stealers

Information stealer attacks, such as Keyloggers and Credential Stealers, are designed to capture sensitive information from compromised systems.

- **Keylogger:** This type of malware records keystrokes, potentially capturing credentials, financial information, or other sensitive data. Uptycs monitors for unusual processes associated with keylogging activities and alerts security teams to potential breaches.
- **Credential Stealer:** Attackers use credential stealers to harvest usernames, passwords, and other authentication tokens. Uptycs detects unusual processes that attempt to access authentication systems or network resources without proper authorization, preventing attackers from gaining further access to cloud services.

By detecting information stealers, Uptycs helps prevent data theft and breaches.

6. Web Shells and Reverse Shells

A Web Shell is a script that attackers use to control a compromised web server, often with a remote connection. Reverse Shells allow attackers to remotely execute commands on a compromised system. Uptycs detects both these activities by monitoring for suspicious web traffic and unexpected outbound connections that could indicate the presence of a web shell or reverse shell.

- **Web Shell:** Uptycs detects signs of web shell activity by analyzing web server logs, looking for suspicious HTTP requests or files that could indicate malicious scripts are running on the server.

- **Reverse Shell:** Uptycs detects reverse shell activity by monitoring for unexpected outgoing network connections, helping to block unauthorized remote access and limit attacker control over cloud workloads. Figure (2) gives an example of how reverse shell is contained and providing deeper insights into which command line and container image led to this detection

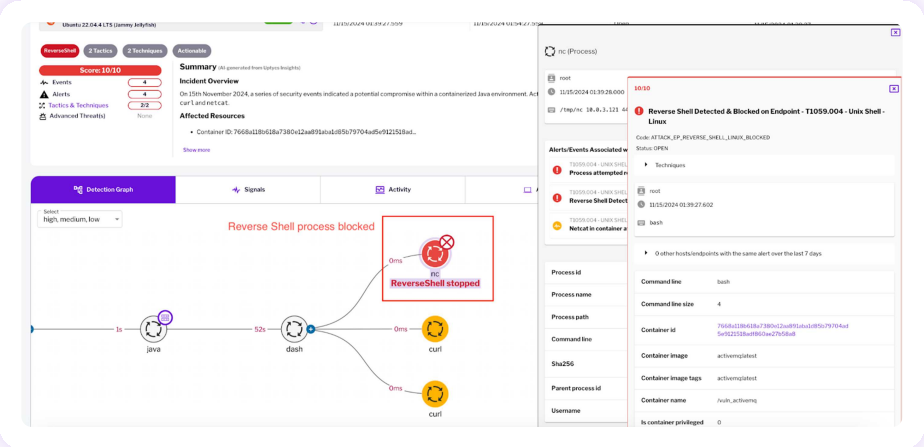


Figure (2) - Detection attribution to a specific command line and container

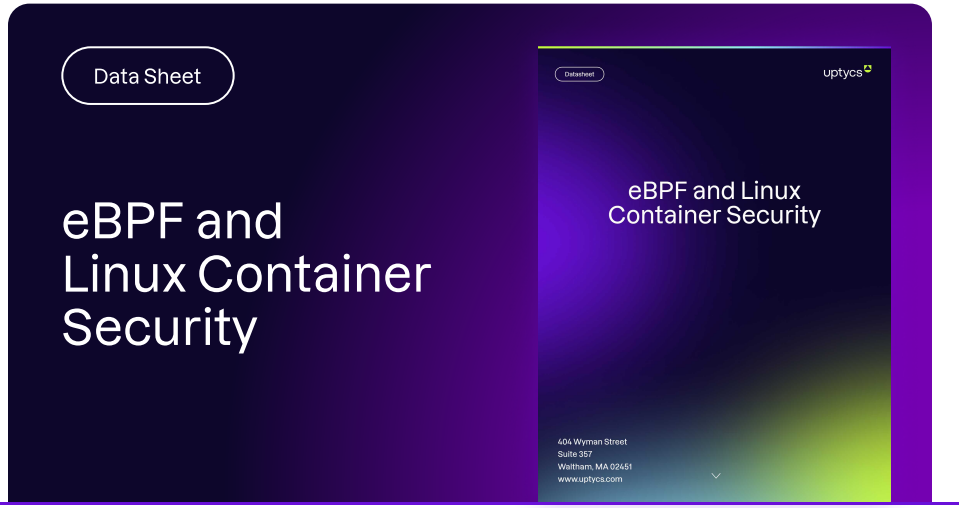
7. Fileless Malware

Fileless malware is a type of malware that runs directly in memory, making it harder to detect through traditional file-based antivirus solutions.

Uptycs is designed to detect Fileless Malware by monitoring system processes and behaviors rather than relying on file signatures. By analyzing runtime behavior, Uptycs can identify anomalies that suggest the presence of fileless malware, helping to prevent attacks that would otherwise bypass conventional security defenses.

8. Malicious Toolkits

Malicious Toolkits like malware and exploit kits, including those detected by Yara rules, are often used by attackers to launch a variety of malicious activities. Uptycs integrates signature-based detection (e.g., Yara rules) with behavioral analysis to identify malicious toolkits and prevent their execution within cloud workloads.



The Comprehensive Benefit

Having an effective unified enforcement strategy to contain attacks is beneficial for the following reasons. Figure (3) shows an example of how a unified enforcement strategy can help Security Operations drive better efficiency

1. **Shared Indicators of Compromise (IoCs):** All these attack types often share similar Indicators of Compromise (IoCs), such as unusual network traffic, unauthorized access attempts, or abnormal system processes. Grouping these attacks together allows security tools to more effectively recognize patterns across multiple vectors.

2. **Comprehensive detection and protection framework:** By combining these attack types, we can improve the efficacy of detection systems that look for signatures or anomalous activity indicative of exploitation.

3. **Faster Incident Response:** Treating these activities under a single policy driven enforcement model enables security teams to respond faster. Rather than spending time distinguishing between different types of attacks, teams can focus on identifying exploit activity, triaging it, and mitigating the threat before it escalates.

4. **Holistic Threat Mitigation:** For example, patch management, which is a common defense against exploit-based attacks, can be prioritized across all types of vulnerabilities—whether they affect a web server or an operating system.

Figure (1) - Attack contained through policy enforcement

After containing threats, the Uptycs Blast Radius Mitigation Framework moves into Root Cause Analysis for deep insights—keep reading to see how each step drives resilient, secure cloud environments.

04 Uncover Hidden Attacks and Perform Fast Root Cause Analysis Via Image Provenance

Now, let's cover **Step 4: Root Cause Analysis & Image Provenance**, which focuses on tracing threats back to their source, down to the code commit level.

This investigative step is essential for understanding how and where vulnerabilities and threats were introduced, allowing teams to prevent future occurrences with a full code commit to runtime provenance. This step aligns with CIS Software Supply Chain guidelines, aiding teams in compliance and long-term security hardening. This article will discuss how Uptycs' fourth step in the Blast Radius Mitigation Framework provides deep-rooted insights that support lasting resilience.

Even with the presence of various offline security scanners, as part of your software pipeline, malicious and vulnerable code makes it to production deployment. It is not to say offline security scanners (e.g. Agentless, CI time, etc) are not necessary. However, they are proving insufficient to catch more new waves of attacks that focus on injecting malicious code rather than exploiting known vulnerabilities.

Hence there is a need to run time protection and also there is a need to do a very quick RCA when these are caught during runtime.

Let's first dive deep into recent attacks that highlight the pain points and then discuss how Uptycs enables quick detection & root cause analysis of these types of attacks.

The Emerging Threat Landscape

The complexity and interconnectedness of modern software development environments have opened new avenues for cyberattacks. Recent incidents highlight the vulnerabilities across different stages of the software supply chain:

These examples underscore the importance of dynamic and comprehensive security measures throughout the software development lifecycle to safeguard against sophisticated and evolving threats.

Recent breaches in the software supply chain reveal how attackers exploit vulnerabilities at different stages of software development, emphasizing the need for robust security measures across the entire lifecycle:

1. **SolarWinds Attack:** Attackers inserted malicious code into the Orion software updates, compromising thousands of organizations, including U.S. government agencies. This breach led to widespread espionage and data theft, demonstrating the risks of supply chain vulnerabilities.

2. **Dependency Confusion:** Attackers uploaded malicious packages to public repositories, tricking organizations into incorporating these compromised packages into their software builds. This resulted in unauthorized access and data theft, highlighting the dangers of relying on public dependencies without stringent validation.

4. **Fake Dependabot Commits:** Attackers mimicked Dependabot, an automated tool for managing dependencies, by creating fake commits. These fake updates could introduce malicious code into software projects, stressing the importance of verifying the authenticity of automated tools and their outputs.

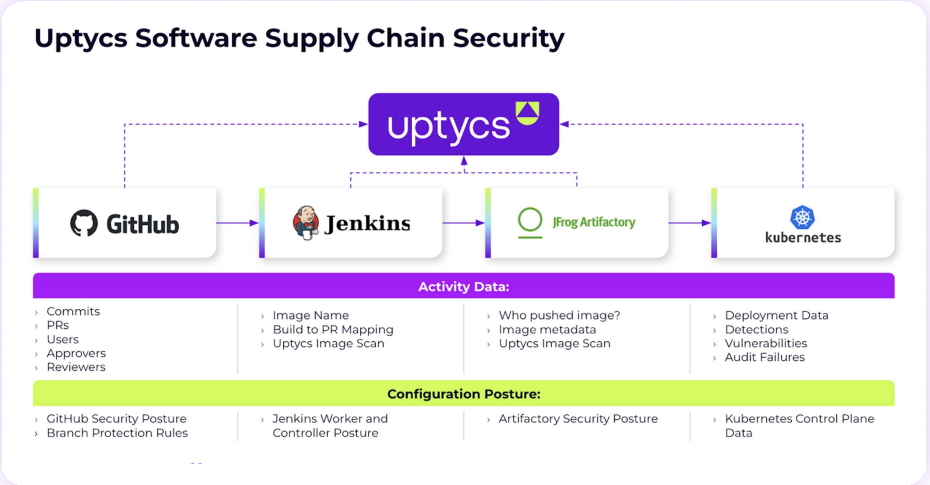
5. **NPM Reverse Shell and Data Mining:** In this type of attack, malicious NPM packages were used to establish reverse shells, giving attackers control over compromised systems.

The Solution

In order to solve for these new waves of threats, you need the following:

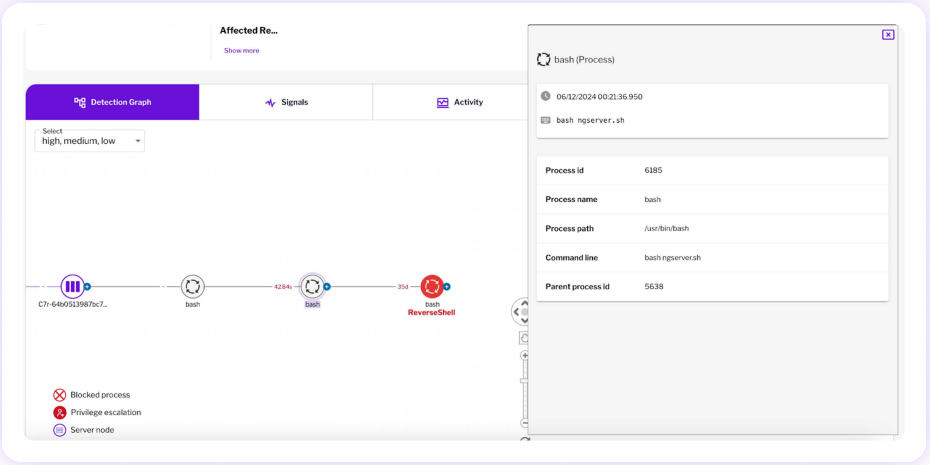
This evidences the need to solve for some key challenges:

- **Image Provenance -** Software install to commit (image provenance) understand entire software process flow that was involved in producing the deployed artifact.
- **Runtime Workload SCA -** an understanding of the software that went into the image including 3rd party dependencies, licensing, and packages and metadata of what is deployed and running.
- **Software Supply Chain Posture -** Your supply chain may be secure now but how secure was it during the time of the image build. Potential vulnerabilities may have been introduced due to software supply chain vulnerabilities.
- **Continuous Compliance and Hardening -** In order to prevent malicious commits from entering the pipeline and addressing software supply chain vulnerabilities in Code repositories, CI pipelines and Registries, it is important to continuously measure and harden your software supply chain.

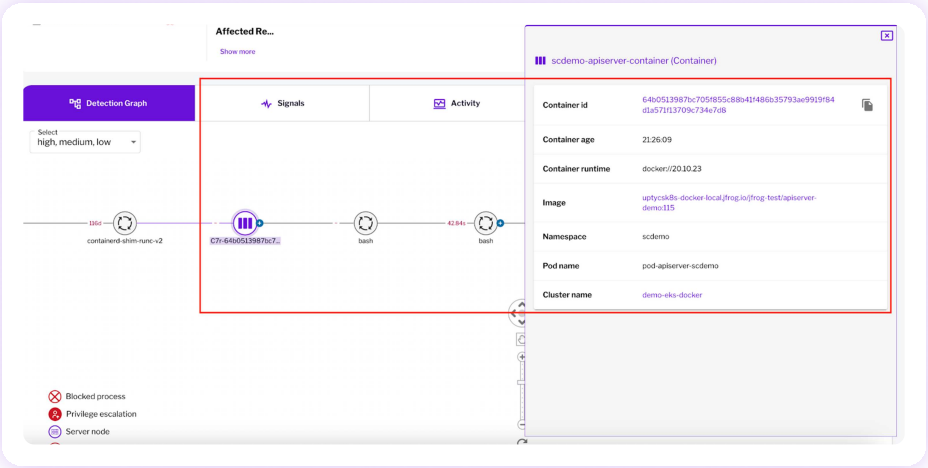


Example

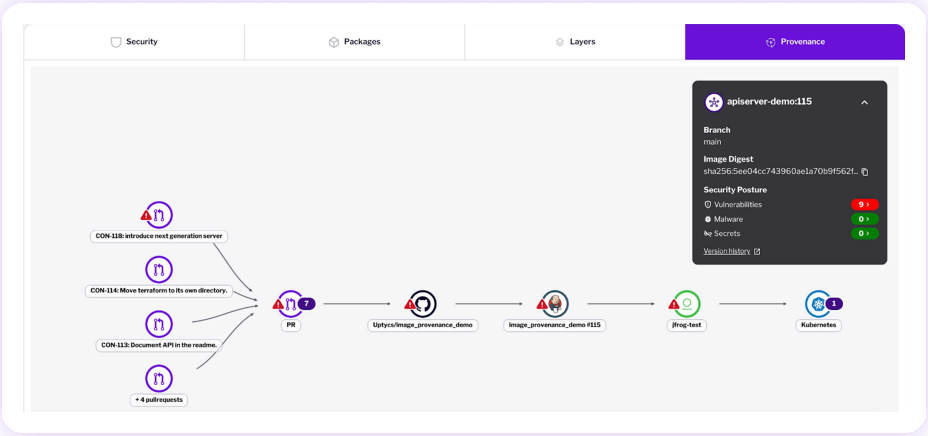
Detection: First we see a reverse shell via eBPF Detection. We see with deep attribution context that the reverse shell came from execution of ngserver.sh



This was spawned from a container instance deployed. We can see the container image - this can tell us about what was checked code wise and also if we caught any vulnerabilities during the build phase.

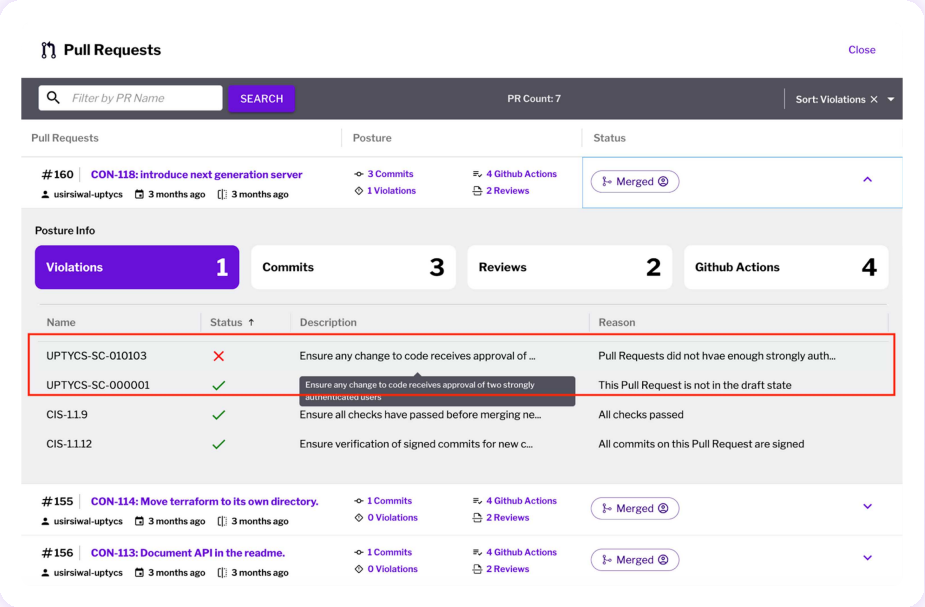


Root Cause Analysis: Next in order to understand the origin of the root cause we go to the container image to see a full code commit to runtime provenance.

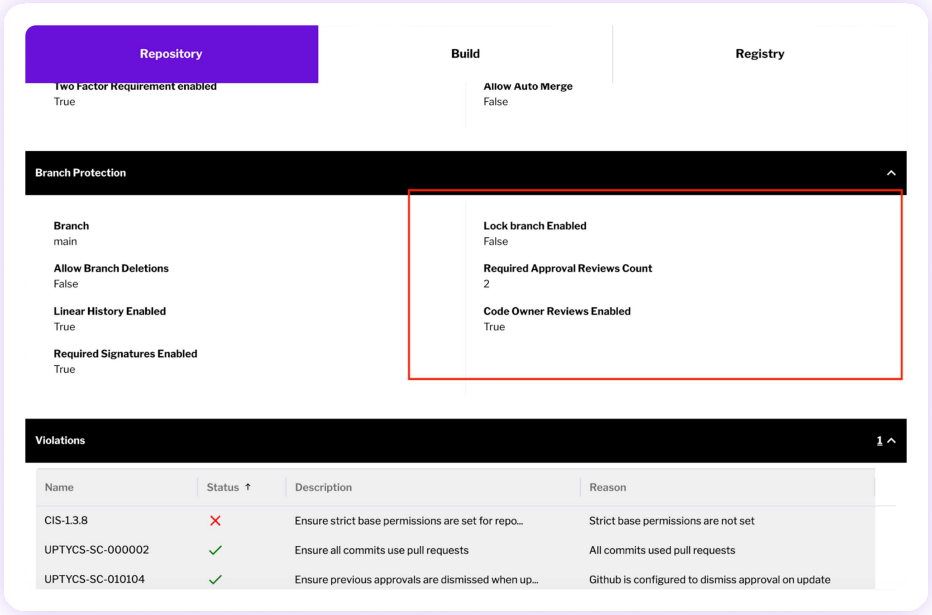


Here we can see that this image went through GitHub, Jenkins, Artifactory, and is now deployed in Kubernetes. We also see that each stage of the development process has a violation. We can work backwards to triage the root cause of the reverse shell detection by looking at the code commit. We see that one of the PRs has violations.

When we look at the PR we see that it only has 1 Approved Review while others have 2. Looking at the Repository Branch Protection Rules, it does say that every PR should have 2 Required Approval Reviews. This could be an indication that there has been a bypass and that some malicious code could have been committed.

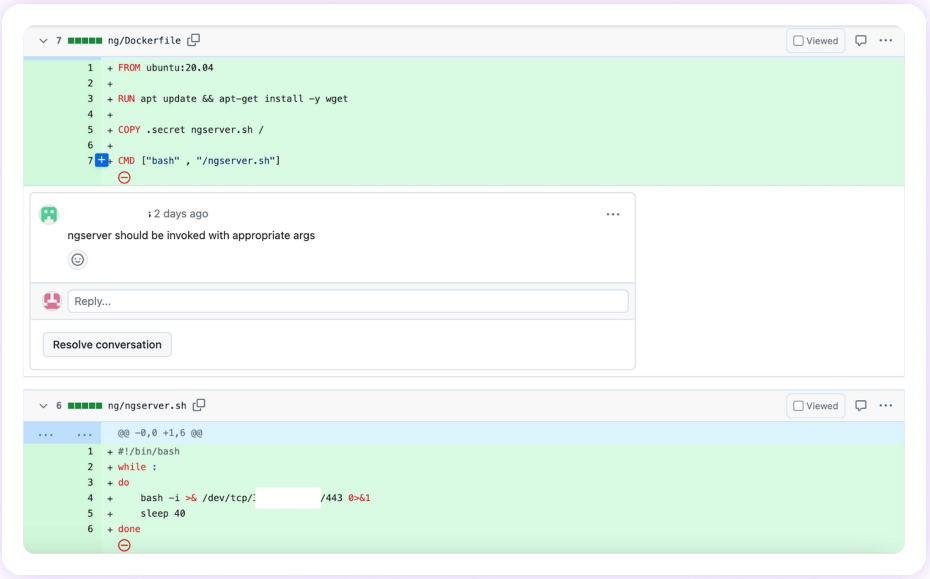


PR failing the check of having 2 Approved Reviews.



Branch Protection rules requires 2 approved reviews per PR

When we dig deeper into the code review, we can see that malicious code for the reverse shell - the same ngserver that we saw in the detection was checked in. This ngserver itself is leading to the reverse shell.



With provenance, organizations can save significant time in root causing vulnerabilities and instead use that time to assess risk and patch key risks and vulnerabilities in their environment!

With Root Cause Analysis completed, the final step in the Uptycs Blast Radius Mitigation Framework ensures proactive defenses—read on to discover how DevSecOps Guardrails round out this powerful security strategy.

05 DevSecOps Guardrails & Risk Mitigation

The five steps of the Uptycs Blast Radius Mitigation Framework are designed to ensure robust security throughout the cloud security lifecycle:

1. **Threat Detection in Workloads**
2. **Unified Risk Assessment and Attribution**
3. **Containment through Policy Enforcement**
4. **Root Cause Analysis & Image Provenance**
5. **DevSecOps Guardrails & Risk Mitigation**

This final section focuses on how Uptycs establishes proactive defenses to reinforce security across CI/CD pipelines. By embedding secure practices throughout development, Uptycs enables DevSecOps teams to reduce the potential attack surface continuously, providing automated guardrails to maintain secure production environments. This post explores how the fifth and final step in Uptycs' Blast Radius Mitigation Framework provides lasting protection and risk reduction across cloud applications.

Once you have attributed a vulnerability or threat to the specific code commit/source and have remediated it, it is ideal to establish a DevSecOps process that enables the following the following:

- Establish a security process that integrates into the development workflow and build pipelines
- Scanning container images and AMIs for vulnerabilities, malware, secrets, integrity and compliance and block builds that contain key security issues before they are deployed into production
- Build Trusted gateways for deployment and only allow deployments from trusted sources

While shift-left promised to address risk before they reach runtime environments, we've seen some key challenges in the market when it comes to actually making this a reality:

- **It is not possible to fix everything at once:** Not every critical vulnerability is fixable. Moreover, the context of the application being deployed and the risk of your runtime environment can help prioritize what vulnerabilities need fixing. Shift-left policies that don't have exception management to be able to handle key use cases fail to be scalable in enterprise organizations.

- **Developer Access/Integration To Security Tooling:** Developers may not have access to your CNAPP tooling but still need context into the vulnerabilities found in their image and how to fix them.

- **Image Deployments From Non-Trusted Repositories:** Many threats emerge from images that are deployed from non-trusted repositories or even public repositories which may have a lot of secure images, but where it is hard to identify the integrity/provenance of every image present. You need to be able to enforce controls pre-runtime deployment as well.

Building Trusted Development Pipelines Through Image Policy and Enforcement

In today's fast-paced development landscape, ensuring the security of your software is more crucial than ever. At Uptycs, we're excited to announce our new Image Security Policies feature, designed to fortify your CI/CD pipeline and streamline the way you manage image vulnerabilities. This enhancement allows you to create trusted gateways that not only enhance security but also empower your development teams.

What Are Image Security Policies?

Image Security Policies allow you to define and enforce security standards for your container images throughout the development lifecycle. By implementing these policies, you can:

1. Automatically scan images for vulnerabilities, malware, and policy violations.
2. Route non-compliant images to a separate repository for review and remediation.
3. Ensure that only trusted, policy-compliant images are deployed to your production environment.
4. Grant exceptions when developers need more time to address issues.

Image Security Policies provide a framework for managing and enforcing security standards throughout your development pipeline. This feature enables teams to identify and mitigate vulnerabilities in container images, ensuring that only trusted images are deployed to your environments.

How It Works

1. Policy-Based Scanning in CI/CD

As part of your CI/CD pipeline, our scanner examines each container image against your defined security policies. These policies can check for:

- Exploitable vulnerabilities
- Presence of malware and secrets
- Compliance issues such as usage of root uses in a dockerfile
- Integrity of the image - making sure that it is signed by the likes of a Cosign or Notary or making sure it comes from a trusted registry source

2. Intelligent Routing

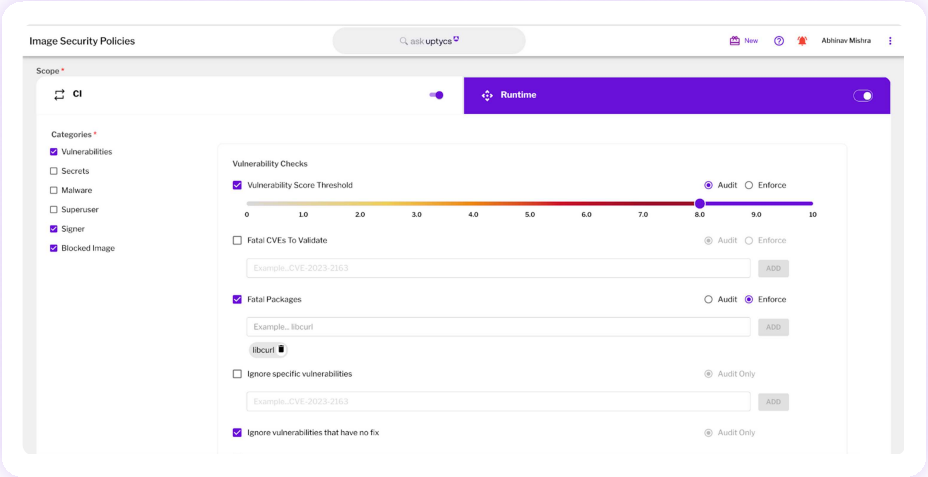
Based on the scan results:

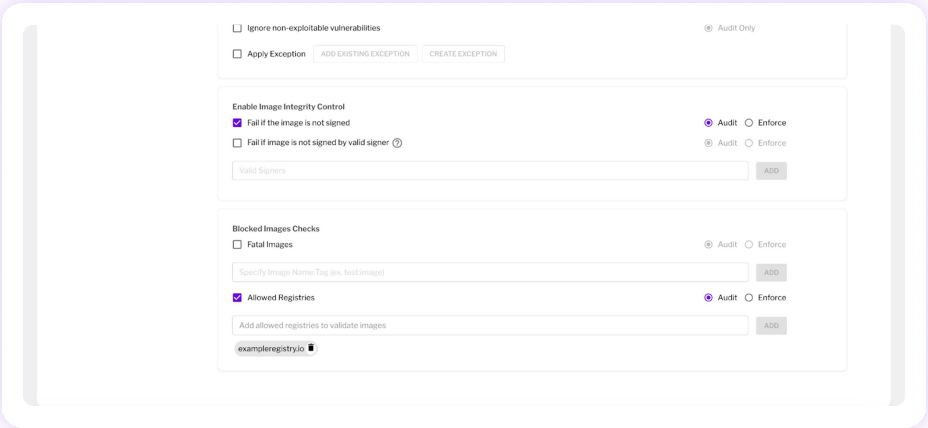
- Images that fail the policy checks can be sent to a designated repository for developer review and remediation.
- Images that pass all checks can be automatically routed to a trusted repository, ready for deployment.

3. Enforced Deployments via Admission Controls

To ensure the integrity of your production environment:

- Kubernetes deployments are configured to pull only from the trusted repository.
- Admission controls can be enabled to prevent any attempts to bypass the trusted gateways.





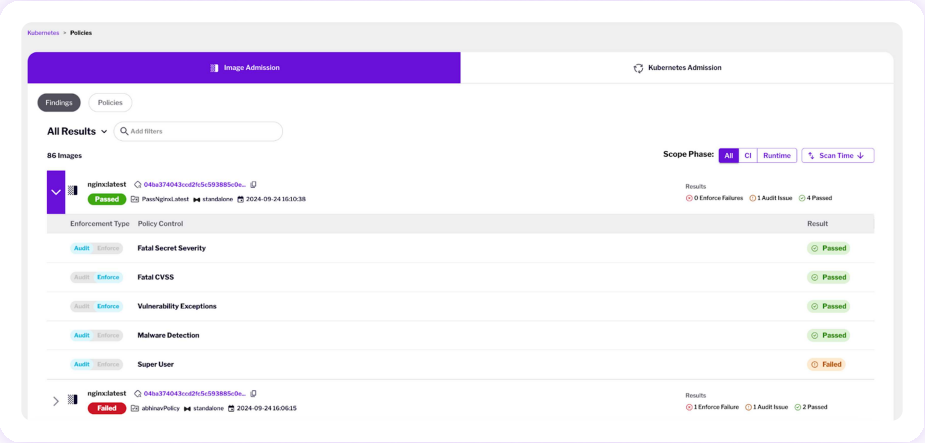
4. Local CLI Scanning

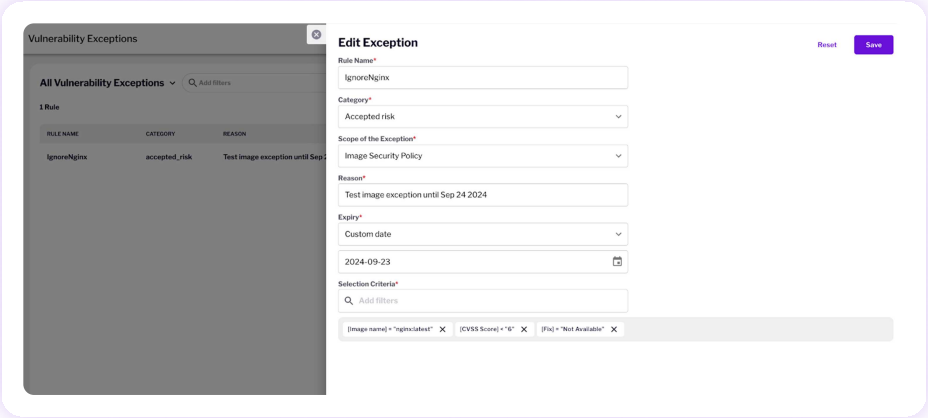
Developers can run our CI scanner locally in their command-line interface before pushing changes. This empowers them to catch and fix issues early, streamlining the development process.

5. Flexible Exception Handling

We understand that security and development velocity sometimes need to be balanced. That's why our Image Security Policies feature includes an exception mechanism:

- If a developer needs more time to address identified issues, an exception can be granted.
- This exception allows the image to proceed through the build or deployment process, even if it doesn't fully meet the defined security policies.
- Exceptions can be time-bound and require appropriate approvals, ensuring that security risks are acknowledged and managed.





Why Image Security Policies Matter

With the rapid adoption of containers and microservices, the risk of deploying vulnerable images has increased significantly. Image Security Policies empower organizations to take proactive steps in managing these risks, ensuring that only secure images are pushed to production. By integrating security directly into the development process, teams can enhance collaboration, reduce friction, and improve overall software quality.

Enhancing Collaboration Between Devs and SecOps

One of the standout benefits of our Image Security Policies is the way it fosters collaboration between development and security teams. With clear visibility into vulnerabilities and a structured process for remediation, developers can work closely with security teams to ensure that their applications are not only functional but also secure.

Conclusion

The introduction of Image Security Policies marks a significant step forward in Uptycs' commitment to enhancing the security of your development pipeline. By implementing these policies, you're not just protecting your applications; you're also enabling your teams to innovate with confidence.

Uptycs is dedicated to leading security innovations in hybrid cloud environments, ensuring robust protection and enabling our customers to innovate safely and efficiently. Included in the 2024 CNAPP Market Guide, Uptycs provides comprehensive security solutions that bridge the gap from code to cloud. Our platform excels in Cloud Workload Protection (CWPP), Vulnerability Management, Cloud Security Posture Management (CSPM), Detection & Response, Software Pipeline Security, XDR, and Risk & Compliance. Trusted by leading enterprises like PayPal and Comcast, Uptycs transforms potential vulnerabilities into fortified security, ensuring your digital environments are safeguarded from development through runtime.

Secure Everything from Dev to Runtime

Learn more at Uptycs.com