oxsecurity

OSC&R in the Wild

A New Look at the Most Common Software Supply Chain Exposures

Introduction

A recent survey from Enterprise Strategy Group (ESG)/TechTarget reported that 91% of organizations experienced at least one software supply chain security incident in 2023. It is clear that AppSec teams need help to better anticipate and defend against the malicious actions used by adversaries to gain access to application code through the software supply chain.

For ***OSC&R in the Wild: A New Look at the Most Common Software Supply Chain Exposures***, the first edition of a planned annual release, we've collected over one hundred million software supply chain security alerts from tens of thousands of repositories, cloud-deployed applications, and organizations. This data was normalized, contextually analyzed, and parsed against the Open Software Supply Chain Attack Reference (OSC&R) framework, the first and only industry-accepted cyber attack matrix focused solely on software supply chain attacks, to identify trends and patterns of malicious behavior across the stages of the software development lifecycle (SDLC).

Understanding how adversaries view and target the attack surface of a software supply chain lays a foundation that enables AppSec, DevOps, and Product Security teams to recognize, prioritize, and remediate weaknesses in their software development environments more effectively and efficiently. By basing our research on such a large real-world dataset, we shed a unique light on the extent to which modern software ecosystems are exposed to security risks.

Table of Contents

Executive Summary

Increased reliance on open-source software and cloud-native technologies has also increased the challenge of securing software supply chains. To better understand the supply chain vulnerabilities plaguing AppSec teams, the first OSC&R Snapshot report demonstrates how supply chain security issues and vulnerabilities can be mapped to the phases of attack. The results are both enlightening and concerning.

Our researchers analyzed nearly one hundred and forty thousand enterprise applications over a nine-month period and correlated the data to the Open Software Supply Chain Attack Reference (OSC&R) threat framework. With OSC&R's kill chain and TTPs specific to software supply chains, we show how an adversary views a software supply chain environment: as a prize to be taken, likely ripe with vulnerabilities if they look hard enough.

While AppSec programs and practices continued to mature in 2023, our analysis indicates there is much more work needed if we are to manage the risks effectively. Detecting and remediating legitimate security risks from within the mountain of benign code-hygiene alerts issued by traditional AppSec technologies continues to be problematic, and serious vulnerabilities are passing to production code with concerning regularity.

Our researchers found nearly half of the applications analyzed had high, severe, or apocalyptic risk scores based on the vulnerabilities they contained, and many were found to contain vulnerabilities spanning multiple stages of the kill-chain, leaving them even more vulnerable to a successful attack. In addition, a surprising number of vulnerabilities were still very commonly seen despite having been documented for years.

Without a fundamental paradigm shift in the way enterprises approach supply chain security, we can expect the problem to increase, a conclusion consistent with macro-trends identified in other recent security research.

However, the status quo is not a foregone conclusion — AppSec teams are not destined for "alert overload." Application security technologies are rapidly evolving to provide AppSec and DevSecOps teams with the tools and processes needed to gain greater visibility and faster response. And we are seeing more investments in proactive application security measures within enterprise markets. Are these changes happening quickly enough? Time will tell.

For example, by integrating OSC&R with existing tools and third-party solutions, and specifically connecting ADR to ASPM, users can create a dynamic and resilient security posture that continuously evolves to meet emerging threats. This approach not only enhances threat detection and response capabilities but also fosters a culture of continuous improvement and adaptation within an organization's security practices.

What is the OSC&R Framework?

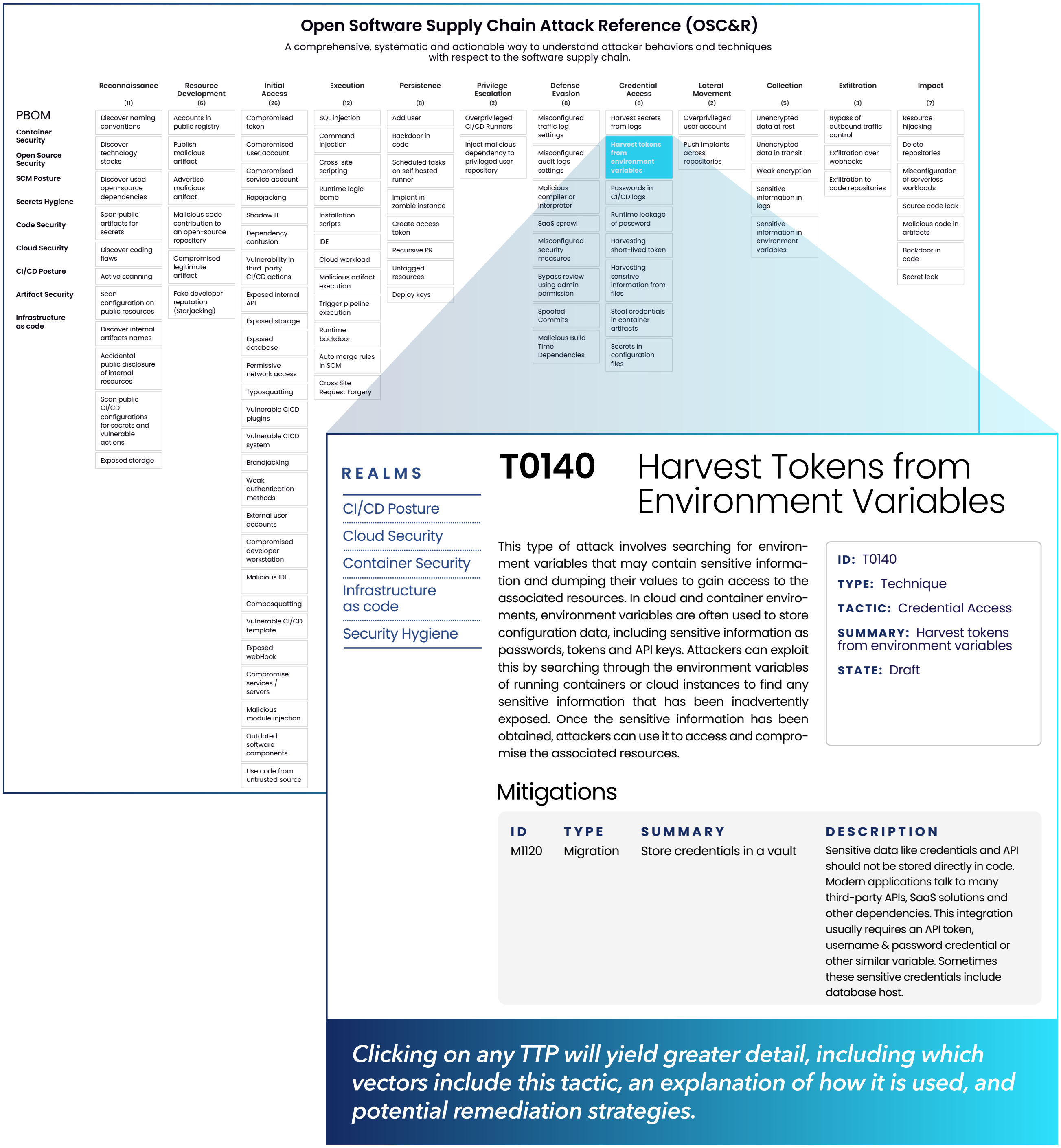
With the astounding growth in cyber attacks targeting enterprise software supply chains in the past five years, it became clear to the AppSec community that a MITRE ATT&CK-like framework was needed to provide a common language and structure for understanding and analyzing the various adversarial behaviors commonly used in these attacks. The result was the 2023 introduction of **OSC&R**, the first and only industry-accepted cyber attack matrix focused solely on software supply chain attacks.

Developed collaboratively by cyber security veterans from OX Security, Microsoft, Oracle, GitLab, Fortinet, FICO, and others, OSC&R provides a systematic, structured mapping of TTPs used in cyber attacks on the software supply chain. From reconnaissance to exfiltration, each TTP is meticulously dissected, providing security and development teams with unparalleled insights into the threats that can potentially compromise their organization's software supply chain. With this, the AppSec community now has a single point of reference for proactively assessing strategies to secure their software supply chain and evaluate the effectiveness of their AppSec programs.

*For a full explanation of the research process, please refer to the Methodology section at the end of this report.

As shown in the OSC&R matrix illustrated in Figure 1, by selecting an attack vector from the vertical list on the left, the user is presented with a complete listing of TTPs for each stage of an attack on that vector.

FIGURE 1: OSC&R matrix & TTP detail



Open Software Supply Chain Attack Reference (OSC&R)

A comprehensive, systematic and actionable way to understand attacker behaviors and techniques with respect to the software supply chain.

PBOM
Container Security
Open Source Security
SCM Posture
Secrets Hygiene
Code Security
Cloud Security
CI/CD Posture
Artifact Security
Infrastructure as code

Reconnaissance (11)	Resource Development (6)	Initial Access (26)	Execution (12)	Persistence (8)	Privilege Escalation (2)	Defense Evasion (8)	Credential Access (8)	Lateral Movement (2)	Collection (5)	Exfiltration (3)	Impact (7)
Discover naming conventions	Accounts in public registry	Compromised token	SQL injection	Add user	Overprivileged CI/CD Runners	Misconfigured traffic log settings	Harvest secrets from logs	Overprivileged user account	Unencrypted data at rest	Bypass of outbound traffic control	Resource hijacking
Discover technology stacks	Publish malicious artifact	Compromised user account	Command injection	Backdoor in code	Inject malicious dependency to privileged user repository	Misconfigured audit logs settings	**Harvest tokens from environment variables**	Push implants across repositories	Unencrypted data in transit	Exfiltration over webhooks	Delete repositories
Discover used open-source dependencies	Advertise malicious artifact	Compromised service account	Cross-site scripting	Scheduled tasks on self hosted runner		Malicious compiler or interpreter	Passwords in CI/CD logs		Weak encryption	Exfiltration to code repositories	Misconfiguration of serverless workloads
Scan public artifacts for secrets	Malicious code contribution to an open-source repository	Repojacking	Runtime logic bomb	Implant in zombie instance		SaaS sprawl	Runtime leakage of password		Sensitive information in logs		Source code leak
Discover coding flaws	Compromised legitimate artifact	Shadow IT	Installation scripts	Create access token		Misconfigured security measures	Harvesting short-lived token		Sensitive information in environment variables		Malicious code in artifacts
Active scanning	Fake developer reputation (Starjacking)	Dependency confusion	IDE	Recursive PR		Bypass review using admin permission	Harvesting sensitive information from files				Backdoor in code
Scan configuration on public resources		Vulnerability in third-party CI/CD actions	Cloud workload	Untagged resources		Spoofed Commits	Steal credentials in container artifacts				Secret leak
Discover internal artifacts names		Exposed internal API	Malicious artifact execution	Deploy keys		Malicious Build Time Dependencies	Secrets in configuration files				
Accidental public disclosure of internal resources		Exposed storage	Trigger pipeline execution								
Scan public CI/CD configurations for secrets and vulnerable actions		Exposed database	Runtime backdoor								
Exposed storage		Permissive network access	Auto merge rules in SCM								
		Typosquatting	Cross Site Request Forgery								
		Vulnerable CICD plugins									
		Vulnerable CICD system									
		Brandjacking									
		Weak authentication methods									
		External user accounts									
		Compromised developer workstation									
		Malicious IDE									
		Combosquatting									
		Vulnerable CI/CD template									
		Exposed webHook									
		Compromise services / servers									
		Malicious module injection									
		Outdated software components									
		Use code from untrusted source									

T0140 Harvest Tokens from Environment Variables

REALMS

CI/CD Posture
Cloud Security
Container Security
Infrastructure as code
Security Hygiene

This type of attack involves searching for environment variables that may contain sensitive information and dumping their values to gain access to the associated resources. In cloud and container environments, environment variables are often used to store configuration data, including sensitive information as passwords, tokens and API keys. Attackers can exploit this by searching through the environment variables of running containers or cloud instances to find any sensitive information that has been inadvertently exposed. Once the sensitive information has been obtained, attackers can use it to access and compromise the associated resources.

ID: T0140
TYPE: Technique
TACTIC: Credential Access
SUMMARY: Harvest tokens from environment variables
STATE: Draft

Mitigations

ID	TYPE	SUMMARY	DESCRIPTION
M1120	Migration	Store credentials in a vault	Sensitive data like credentials and API should not be stored directly in code. Modern applications talk to many third-party APIs, SaaS solutions and other dependencies. This integration usually requires an API token, username & password credential or other similar variable. Sometimes these sensitive credentials include database host.

Clicking on any TTP will yield greater detail, including which vectors include this tactic, an explanation of how it is used, and potential remediation strategies.

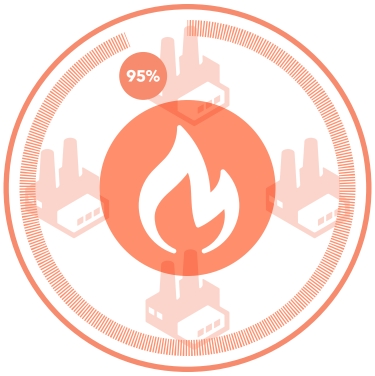
Key Findings

The following section provides highlights from our analysis of the applications and vulnerabilities in the sample set. Additional details for these findings are provided in the subsequent sections of this report.

AppSec Teams Face an Unmanageable Volume of Alerts

Organizations are overwhelmed by the sheer volume of vulnerability alerts generated across the software supply chain. Based on the active applications collected by our researchers over a nine-month period, security teams must monitor 129 applications and manage over 119,000 alerts on average.

119k
ALERTS
129
APPS



Nearly 95% of Organizations Face High Severity Risks

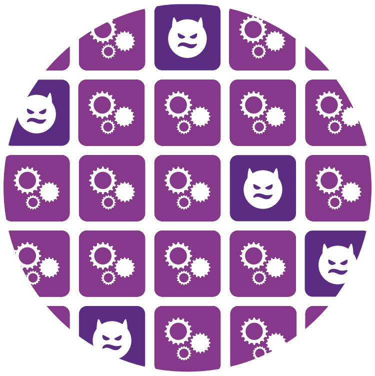
Ninety-five percent of organizations had at least one high, critical, or apocalyptic risk within their software supply chain, with the average organization having nine.

Automated Alert Analysis Helps Find the Signal in the Noise and Respond

Using the OX Security Active ASPM platform to deduplicate alerts and consolidate issues tied to a common root cause, our researchers reduced the massive volume of unscrubbed alerts by 97%. Using the OX platform to then apply contextual analysis based on reachability, exploitability, and business impact, the team isolated alerts representing a high, critical, or apocalyptic risk (our three highest rankings of severity), and further reduced the alert volume down to just one-half of one percent of the original volume.

One in Five Applications Contain Run-Time Exposure

Analysis against attack phases showed that 20% of all applications have high, critical, or apocalyptic issues during the Execution stage, where attackers aim to deploy malicious code. The most significant number of security issues are typically found in later attack phases, when the potential impact on business operations is higher.

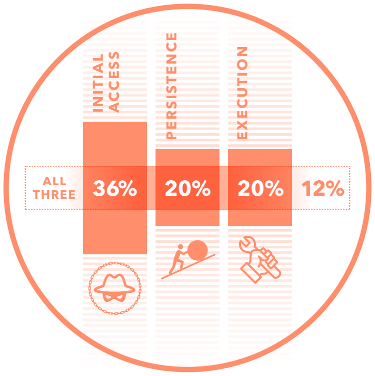


Older Vulnerabilities are Still the Most Common

While some newer tactics did appear, the three most frequently observed vulnerabilities: command injection (15.4% of applications), sensitive data in log files (12.4% of applications), and cross-site scripting (11.4% of applications) have all been around for many years.

Nearly Half of Applications Have High Severity Security Issues

Even with the substantial noise reduction, almost 50% of applications evaluated exhibited high, critical, or apocalyptic issues in at least one attack stage across the kill-chain.



Vulnerabilities Across Multiple Attack Stages Offer Fertile Ground for Attack

36% of applications were vulnerable to exploits in the Initial Access attack stage. 20% of those were also vulnerable to either Persistence or Execution exploits, and 12% were found to contain vulnerabilities tied to tactics in all three stages.

A Detailed Look at the Results

AppSec Teams are Drowning in a Sea of Useless Alerts

One of the primary complaints among AppSec teams is the volume of alerts they must manage, a challenge our researchers faced as well as they analyzed the applications from our expansive sample. Our dataset included nearly 106,000,000 alerts collected from nearly 900 organizations, resulting in an average of roughly 118,000 alerts per organization. Applying automated analysis with the OX Security Active ASPM platform to deduplicate alerts and consolidate issues tied to a common root cause decreased the alert volume to an average of 4,000 alerts per organization, a reduction of approximately 97%. Despite this significant reduction, this volume is still much too high for the average AppSec team to triage.

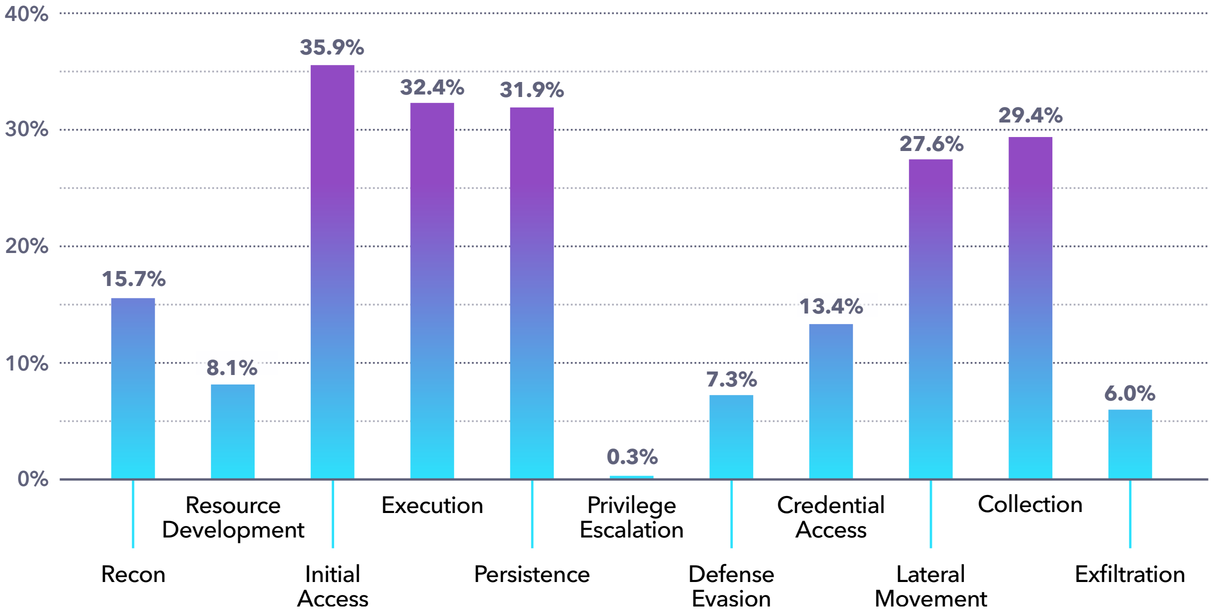
We applied contextual analysis with the OX platform to determine the reachability, exploitability, and business impact of the vulnerability, isolating alerts that represented a high, critical, or apocalyptic risk to the organization. This further reduced the alert volume to approximately 660 per organization or one-half of one percent of the original volume.

The reduction in alert volume based on these two processes was profound, but more importantly, it enabled our researchers to identify those vulnerabilities that pose a significant security risk, which could then be mapped to the OSC&R matrix.

Vulnerabilities Found Across the Kill Chain

Understanding where an application is most vulnerable to attack is foundational to an effective AppSec strategy. To map where vulnerabilities were most commonly located, , our researchers categorized all of the vulnerabilities according to the OSC&R attack stages . The distribution is shown in Figure 2.

FIGURE 2: % of Applications with at Least One Vulnerability



RECONNAISSANCE

The first step in any attack is Reconnaissance, whereby the attacker attempts to gather information about the target environment, including naming conventions and technology stacks, and scanning publicly available CI/CD configurations for potential vulnerabilities, misconfigurations, and secrets to plan the attack. Our research found that nearly 16% of applications contained vulnerabilities that could allow an adversary to gather meaningful information.

INITIAL ACCESS

Vulnerabilities that could be exploited during the Initial Access stage were the most common exposures within the OSC&R matrix, with 36% of scanned applications exhibiting vulnerabilities in this category. Attackers can capitalize on a wide range of techniques, including compromised tokens/accounts/workstations, exposed APIs or databases, and weak authentication, among many others, to gain initial access to the development environment.

EXECUTION AND PERSISTENCE

There was also a high incidence of weaknesses that could be exploited using tactics tied to the Execution and Persistence stages. Thirty-two percent of applications contained vulnerabilities that could be exploited in one or both of these stages. Tactics frequently employed in the Execution stage include SQL and command injections, runtime logic bombs and backdoors, and cross-site scripting. Many of these techniques appear in the **OWASP Top 10 list** of the most critical security risks to web applications, so vulnerabilities in these areas are particularly concerning.

Tactics to achieve persistence in the target environment, which enable the attacker to return to the compromised environment at a later time, include adding an unauthorized privileged user account, inserting a backdoor into the code, or creating a persistent access token.

LATERAL MOVEMENT

The path of an attack is rarely a straight line. The attacker will gain entry at the weakest point they can find, but then must traverse the environment to locate and gain access to their ultimate target. Vulnerabilities that could enable lateral movement were found in 28% of applications analyzed.

The rather straightforward process of accessing overprivileged user accounts to move across the environment is one example of a lateral movement tactic. More sophisticated techniques include gaining access to the target's code repository (using stolen credentials), implanting malicious code or a backdoor, and pushing the modified code back to the repository. The victim then unknowingly deploys the modified code, resulting in the compromise of their systems.

COLLECTION

Alerts tied to the Collection stage were identified in 29% of applications. As the name implies, Collection tactics focus on an attacker collecting data, code, or other digital assets. In most cases these tactics exploit weak encryption or usable data, such as personally identifiable information (PII), tokens, credentials, and internal system information, that is stored in operational logs or in the code itself.

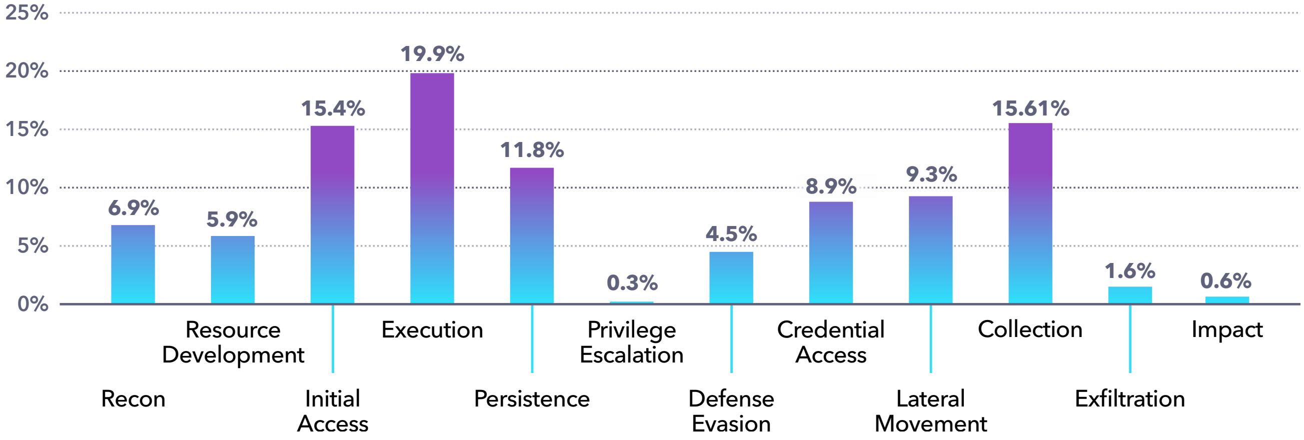
OTHERS

The Resource Development, Privilege Escalation, Defense Evasion, Credential Access, Exfiltration, and Impact stages of the OSC&R kill chain showed a smaller frequency of occurrence. Vulnerabilities tied to these stages were found in less than 15% of the applications in the sample.

Execution Phase More Pronounced Among High-Severity Issues

When we isolate the applications in which high, critical, or apocalyptic issues were found as shown below in Figure 3, the distribution across the OSC&R stages was largely the same as that of the whole dataset shown in Figure 2, except for issues tied to the Execution phase, which were more prevalent in the higher severity subset. This is likely because many exploits related to vulnerabilities in the Execution phase could enable the attacker to run malicious code directly. For this reason, they tend to have higher CVSS scores and therefore rate higher in our severity rankings.

FIGURE 3: % of High Impact Issues Mapped to OSC&R Tactics



Applications Vulnerable Across Multiple Attack Stages

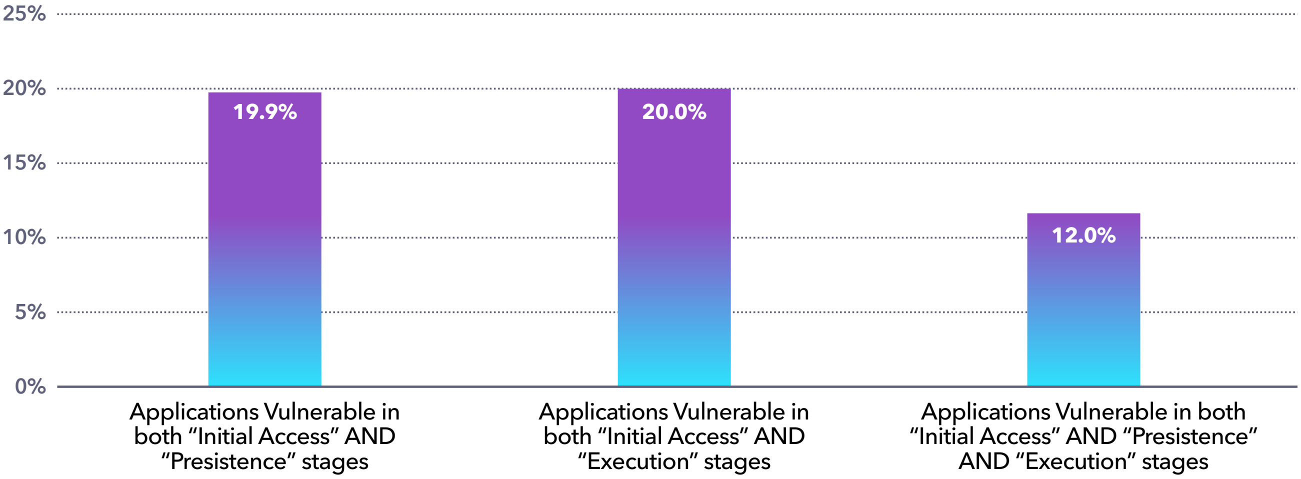
This analysis provides insights about not only vulnerabilities by individual attack stage, but also the frequency in which applications contained multiple vulnerabilities that spanned across stages.

Applications vulnerable to exploits across combinations of the Initial Access, Execution, and Persistence stages offer especially fertile ground for an attack. The convergence of these tactics creates a domino effect, amplifying the consequences of each vulnerability and deepening the potential damage to the software supply chain ecosystem. An adversary facing a target that is vulnerable to initial access and execution or persistence stands a much greater chance of executing a successful attack.

More applications had vulnerabilities tied to the Initial Access attack stage than any other stage with 36% of applications being vulnerable to Initial Access tactics.

As shown in Figure 4, further investigation into the frequency of cross-technique vulnerabilities showed approximately 20% of those applications vulnerable to Initial Access tactics were also vulnerable to Persistence exploits. Roughly the same percentage were vulnerable to both Initial Access and Execution exploits. And approximately 12% were found to contain vulnerabilities tied to tactics in all three stages.

FIGURE 4: Applications Exposed to Multiple Stages of Attack



Most Common TTPs by Attack Stage

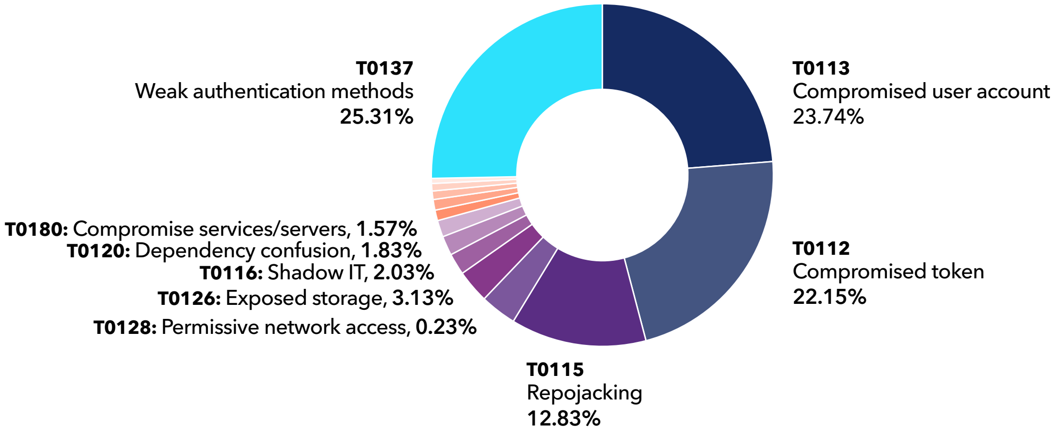
As shown in Figures 3 and 4, vulnerabilities identified in our application sample were most frequently tied to the Initial Access, Execution, Persistence, and Collection stages of OSC&R attack kill chain. We took that analysis deeper, to identify the specific TTPs that were used most frequently within each of those stages.

INITIAL ACCESS

The initial foothold in a software supply chain attack is often the first step toward a complete chain of compromise. Many techniques with which attackers attempt to establish their initial access are visible to common security controls, yet over 15% of applications evaluated had high, critical, or apocalyptic issues that could enable an attacker to gain initial access.

Figure 5 indicates that the majority of the high, critical, or apocalyptic vulnerabilities tied to the Initial Access stage were related to either weak authentication methods, compromised user accounts, compromised tokens, or repo jacking (an adversary taking over the admin account of a targeted code repository, or the admin user name became available to register after a name change).

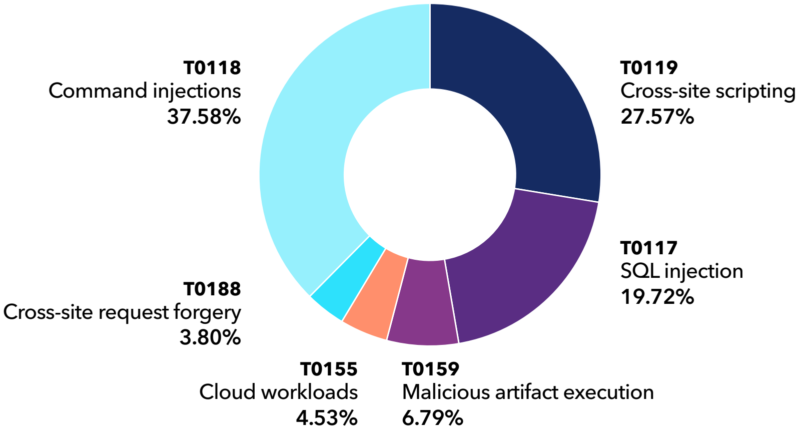
FIGURE 5



EXECUTION

Of the applications found to have high, critical, or apocalyptic vulnerabilities, 20% were exposed to Execution tactics. Figure 6 below shows three techniques—command injection, cross-site scripting, and SQL injection—comprised nearly 85% of all exposures tied to the Execution stage. These techniques allow adversaries to leverage web vulnerabilities and inject malicious code to exploit their foothold. Many of these issues can be mitigated by employing best practices around input validation and modern web frameworks.

FIGURE 6

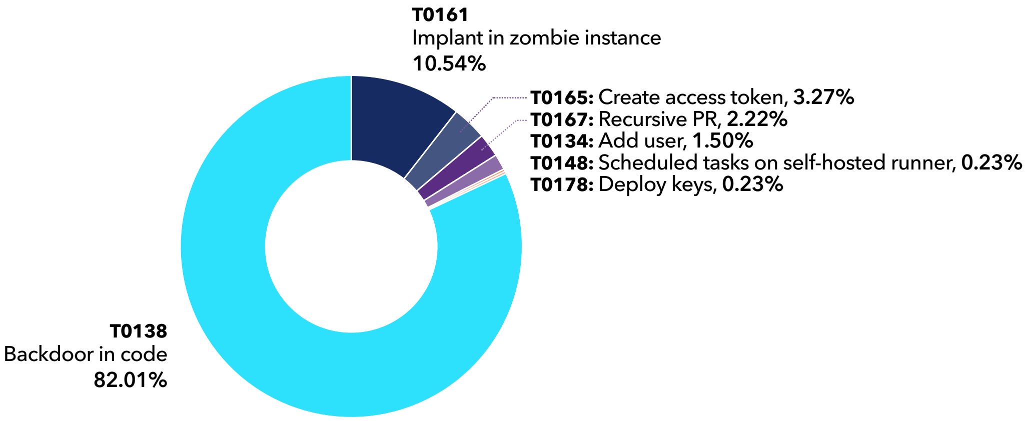


PERSISTENCE

Establishing persistence within a compromised environment enables an adversary to come back and continue an attack at a time of their choosing. During this phase, adversaries use any access, action, or configuration changes that enable them to maintain their foothold on systems, such as replacing or hijacking legitimate code or adding startup code.

Our analysis found that nearly 12% of applications had high, critical, or apocalyptic issues that expose them to persistence techniques. By far, the most prominent issue is vulnerability to placing a backdoor in the code, as shown in Figure 7. This is particularly concerning, as a backdoor can allow the attacker to maintain access and control over the system even after their initial access has been removed or blocked. Mitigating persistence issues requires a holistic approach that includes stringent access controls, regular security audits, and proactive monitoring of code repositories.

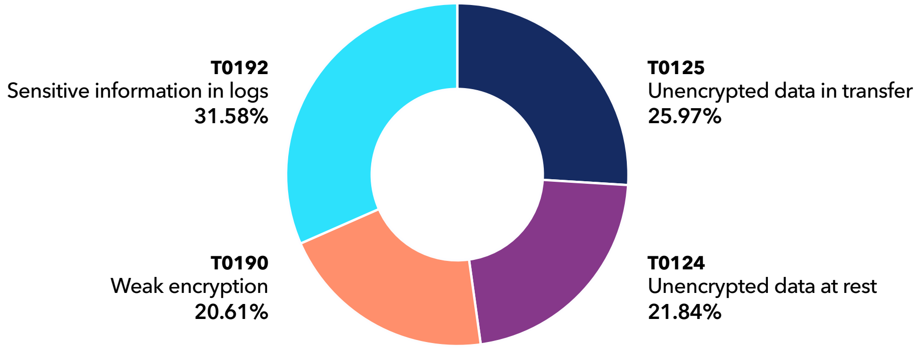
FIGURE 7



COLLECTION

Collecting sensitive information emerged as an exposure in over 15% of applications containing high, critical, or apocalyptic issues. As can be seen in Figure 8, the findings were concentrated around five tactics. All of these TTPs rely on poor protection of sensitive data. In addition to exposing the application to the risk of compromise and manipulation, poor data protection practices can also make organizations vulnerable to compliance violations, legal liability, and reputational damage, depending on the type of data that is compromised. Security teams should enforce encryption at rest and in motion, comprehensive data leak prevention strategies, and stringent data handling practices to mitigate these risks.

FIGURE 8



Top 10 Attacker Techniques Map to Age-Old Security Issues

During the analysis, our researchers sought to determine if threat trends were shifting by identifying the most prevalent attacker TTPs as defined by OSC&R.

OSC&R Technique		% apps
T0138	Backdoor in code	30.87%
T0131	Overprivileged user account	27.57%
T0118	Command injection	27.19%
T0192	Sensitive information in logs	22.02%
T0119	Cross-site scripting	21.66%
T0158	Vulnerable CI/CD template	18.72%
T0125	Unencrypted data in transit	18.41%
T0190	Weak encryption	13.68%
T0124	Unencrypted data at rest	13.67%
T0137	Weak authentication methods	13.12%

Four of the top ten techniques found among security issues—sensitive information logs, unencrypted data in transit,weak encryption, and unencrypted data at rest—all fall under the Collection phase of the kill-chain, and are all tied to data security issues.

The prevalence of **TO138: Backdoor in Code** vulnerabilities is particularly concerning. Attackers have been exploiting vulnerabilities to insert malicious backdoor code for more than 25 years. Yet the recently discovered **CVE-2024-3094** exploit, which targets XZ Utils, a widely used package present in major Linux distributions, makes it clear that adversaries are still successfully targeting this attack path. With the prevalence of these vulnerabilities in our code sample, it's not hard to understand why.

Our Take

One of the questions this analysis sought to answer was whether there was alignment between the vulnerabilities found in the wild and the focus of AppSec teams. The data suggest that these are not yet aligned. We found significant volumes of vulnerabilities at the beginning, middle, and end stages of the kill chain, indicating that companies are still vulnerable to high-impact vulnerabilities. The volume of vulnerabilities in the Initial Access, Execution, and Persistence stages are of particular concern, as successful exploits in these stages can have the greatest influence on a successful attack.

While AppSec teams struggle to scale against the volume of alerts being generated, security debt continues to accumulate. As the gap between vulnerability and exploitation continues to shrink, teams must find ways to work smarter and more efficiently if they are going to gain ground. Our analysis showed that alert volume can be greatly reduced by applying world-class automated consolidation, deduplication, and contextual analysis. In applying these processes, our researchers reduced the average volume per organization from 118,000 raw alerts down to 666 high-priority issues. This is still a significant number, but much more manageable.

To further enhance efficiency, integrating Application Detection and Response (ADR) and Application Security Posture Management (ASPM) capabilities against frameworks like OSC&R is crucial. ADR, particularly agentless ADR solutions, helps monitor application behavior continuously and detect anomalies without the need for additional software agents. While ASPM supports the prioritization. This integration ensures that threats are identified and addressed promptly, creating a dynamic and resilient security posture.

It is clear from this analysis that we still have a long way to go before the problem of vulnerable software supply chains is solved. Progress is being made, but the high volume of vulnerabilities that are passed through the supply chain into live applications, and the large percentage of organizations reporting supply chain security incidents, both indicate that greater focus could be put on prevention rather than on detection.

To learn more about the OSC&R framework, visit **pbom.dev**.

For more information on OX Security's Active ASPM platform, and OX's data consolidation and contextual analysis, **schedule a demo** with one of our Application Security experts.

Methodology

To investigate the exposure of software applications, repositories, and organizations to vulnerabilities outlined in the OSC&R attack matrix, we implemented a systematic, data-driven approach consisting of several stages of analysis. Between June 1, 2023 and March 1, 2024, our research team collected a diverse dataset of enterprise applications from over 138,000 repositories and organizations across multiple industries. This data was enriched to include details such as technology stacks, open-source dependencies, CI/CD configurations, naming conventions, and coding practices.

This dataset was consolidated to eliminate duplicates and combine individual issues linked by a common root cause. Vulnerabilities were mapped to the OSC&R attack matrix before and after consolidation to categorize them according to the stage of attack in which they were likely to be exploited.. In the vulnerability analysis phase, our researcher used automated analysis technologies to identify and quantify vulnerabilities in each category, prioritizing them based on their potential impact and likelihood of exploitation. Exposure profiling was then conducted to assess the extent and severity of exposure across different stages of the OSC&R attack, identifying common trends and patterns that increase susceptibility to particular vulnerabilities.

Throughout this process, ethical considerations were rigorously maintained, ensuring that all research data adhered to ethical guidelines and data privacy regulations, with sensitive information protected and anonymized, as necessary, to uphold confidentiality.

We automatically enrich and triage the data by employing unique and adaptive models based on proprietary research and threat intelligence. These models actively collect business, environmental, and attack contexts such as vulnerable code reachability, exploit applicability, application posture, and business priority.

About the OSC&R Community and Framework

The Open Software Supply Chain Attack Reference (OSC&R) community is a collaborative effort dedicated to enhancing the security of software supply chains. Launched in February 2023 and led by OX Security, the community includes cybersecurity veterans from OX Security, Microsoft, Oracle, GitLab, Fortinet, and FICO. These experts are responsible for creating the OSC&R framework. Modeled after MITRE ATT&CK, the OSC&R framework helps organizations assess their software supply chain security strategies, identify vulnerabilities, and compare solutions effectively. As an open-source framework, OSC&R provides actionable insights into the tactics, techniques, and procedures (TTPs) used by adversaries to compromise software supply chains.

Integrating the OSC&R framework with existing tools and third-party solutions, and specifically connecting Application Detection and Response (ADR) to Application Security Posture Management (ASPM), enables a seamless and efficient way to monitor application behavior without the need for additional software agents. This integration creates a continuous feedback loop, known as a Digital Learning Loop (DLL), enhancing the overall security posture by automating threat detection and response processes. By providing a standardized language and framework, OSC&R empowers the security community to proactively secure software supply chains and mitigate risks.

The OSC&R community is consistently looking for ways to enhance proactive security measures and shares insights at industry conferences like RSA, OWASP, and Black Hat. This active engagement fosters a culture of collaboration and continuous improvement within the community.

For more information, visit **pbom.dev** or join the conversation and contribute to our Slack community **here**.

PBOM
.DEV

Thank you to the Original Authors of the OSC&R Framework

Neatsun Ziv
Co-Founder & CEO
OX Security

Lior Arzi
Co-Founder & CPO
OX Security

Eyal Paz
Head of Research
OX Security

David Cross
former Microsoft and Google cloud security executive

Hiroki Suezawa
Senior Security Engineer
GitLab

Naor Penso
Head of Product Security
FICO

Shai Sivan
CISO, Kaltura

Dineshwar Sahni
Senior Cybersecurity leader

Maxim Kovalsky
Managing Director
Cybersecurity and Privacy,
Grant Thornton

Dr. Chenxi Wang
former OWASP Global
Board member

Roy Feintuch
former Cloud CTO at
Check Point Technologies

Hadas Harel Lavie
Senior Security Architect
at eToro

Ronen Atias
Security Architect at
OX Security

Gadi Evron
former Innovation Domain
Lead, AppSec at Citibank