HOWDY! Who likes an agenda?! 🚀🚀🚀🚀🚀

- Intros
- CVE noise and breaking it through
- Vulnerability prioritization demonstration
Breaking Through CVE Noise: Analyzing 5 Key Prioritization Inputs
Streamlining Vulnerability Resolution Process
Understanding CVE Noise

Managing thousands of CVEs poses significant challenges for security teams in prioritizing threats effectively.

With thousands of new CVEs reported annually, security teams face the daunting task of sifting through and prioritizing the most critical vulnerabilities.

Distinguishing between the severity levels of vulnerabilities is crucial to allocating resources efficiently and addressing the most impactful threats first.
CVSS Scores

The Common Vulnerability Scoring System (CVSS) provides a standardized method for assessing and prioritizing vulnerabilities based on their severity and impact on systems: https://nvd.nist.gov/vuln-metrics/cvss

EPSS Scores

The Exploit Prediction Scoring System (EPSS) helps security teams predict which vulnerabilities are likely to be exploited and need immediate attention, enabling proactive security measures.

CISA KEV Catalog

The CISA (Cybersecurity & Infrastructure Security Agency) Known Exploited Vulnerabilities (KEV) Catalog helps identify vulnerabilities actively exploited in the wild, guiding security teams in focusing on immediate threats with known exploit activity.
VEX Data

Vulnerabilities Exposure Factor (VEX) communicates whether or not a vulnerability is exploitable given its real world use case and its potential impact across the business organization.

Reachability Analysis

Reachability analysis infers if the vulnerability is reachable in your first-party code. Reachability analysis lists call paths from your first-party code to vulnerable functions associated with CVEs. This way, you can proactively remediate the issue by modifying your usage of dependency. This highly depends on your build environment.

Business Context

Prioritization of vulnerabilities comes with what you may know in your own business (e.g. the types of security tools that are available (or not available), the cybersecurity budget your organization you may have, the applications that are the most valuable to your org).
CVSS (Common Vulnerability Scoring System) has long been seen as the de facto mechanism for vulnerability prioritization, but industry focus in recent years has shifted to exploitation as an indicator for vulnerability risk.

Projects such as EPSS and CISA Known Exploited Vulnerabilities (KEV) have emerged, providing visibility into this data for the entire security community.
Example of KEV Data

https://www.cisa.gov/known-exploited-vulnerabilities-catalog

VMWARE | VCENTER SERVER

🚨 CVE-2022-22948

**VMware vCenter Server Incorrect Default File Permissions Vulnerability**: VMware vCenter Server contains an incorrect default file permissions vulnerability that allows a remote, privileged attacker to gain access to sensitive information.

Known To Be Used in Ransomware Campaigns? **Unknown**

**Action**: Apply mitigations per vendor instructions or discontinue use of the product if mitigations are unavailable.

- **Date Added**: 2024-07-17
- **Due Date**: 2024-08-07

Additional Notes +
Vulnerability Exploitability Exchange (VEX)

VEX (Vulnerability Exploitability eXchange) is a set of formats used to describe whether vulnerabilities that affect components of a software product affect the product itself.

Why is VEX important?

The vast majority of vulnerabilities actually aren’t exploitable in their real-world product context.

Reasons why include that:

- The vulnerable component isn’t present
- The vulnerable code isn’t present
- The vulnerable code can’t be controlled by an adversary
- The vulnerable code isn’t in the execute path
- Inline mitigations already exist

VEX data (which is provided by the software supplier) complements other vulnerability inputs (like EPSS, CVSS, and reachability analysis) to provide a more accurate picture of security risk.

Vulnerability Exploitability Exchange

VEX Data

```
"published": "2024-03-06T22:15:57.000Z",
"affects": [
  {
    "ref": "pkg:deb/ubuntu/libcrypt20@0:1.10.2-3ubuntu1?arch=amd64&distro=ubuntu-23.10",
    "versions": [
      {
        "version": "amd64@0:1.10.2-3ubuntu1",
        "status": "affected"
      }
    ]
  }
],
"analysis": {
  "state": "not_affected",
  "justification": "Inline mitigations already exist",
  "detail": "Inline mitigations exists outside the usage of this package ensuring our production deployment is not affected by this vulnerability. Please review - https://dave-fossa.atlassian.net/browse/DAVE-172, for detailed analysis."
}
```
Answer the basic, adminy questions first

- Do I know the high level functions of the application(s) that I support? List these functions.
- What programming languages, frameworks and package managers are used?
- How well do I know about the open source packages that are used in the applications?
- Are there existing processes in place to block code changes if there are vulnerabilities present?
Recommendations

- Look at vulnerabilities identified in direct dependencies
- Filter by CVSS score
- Is there a fix available?
  - Start with patches or minor upgrades
- Focus on vulnerabilities listed in the KEV catalog
- Sort by the highest EPSS score within the detected KEV CVEs.
Next level recommendation

Explore/review results from **reachability analysis** (for contextual risk assessment)
- Previous inputs may or may not apply
Let’s tackle vuln overload in an actual SCA tool.....

Oh, yeah, like FOSSA
Some takeaways

**Combine Severity and Exploitability to Assess Risk**

Use **CVSS scores** for severity and **EPSS scores** for the likelihood of exploitation to assess overall risk.

**Prioritize Actively Exploited Vulnerabilities**

Prioritize vulnerabilities in the **CISA KEV catalog** and use **VEX data** to understand exploitability and mitigation availability.

**Contextualize with Reachability Analysis**

Incorporate **reachability analysis** to determine if vulnerabilities can be exploited in your environment, ensuring a focused and relevant risk assessment.

*Combine reachability insights (if available/detected) with CVSS, EPSS, KEV, and VEX data to form a comprehensive risk assessment. This ensures that you prioritize vulnerabilities that are not only severe and exploitable but also relevant to your specific threat landscape.*
Email **chelsea@fossa.com** for a **FREE**
pdf on **vulnerability resolution!!!!!!**

*BONUS open source compliance resolution ⋆°_inside°φ°*
QA time!!

Open to discuss...

- Share your process around resolving vulnerabilities
  - As an OSS maintainer
  - You’re actually working at a company (startup, enterprise?)
- As a *new* open source contributor, how can I help?
Additional references

**CISA.gov justifications:**

1. Component_not_present
2. Vulnerable_code_not_present
3. Vulnerable_code_cannot_be_controlled_by_adversary
4. Vulnerable_code_not_in_execute_path
5. Inline_mitigations_already_exist

**FOSSA justifications:**

1. Component_not_present
2. Incorrect_data_found
3. Inline_mitigations_already_exist
4. Vulnerable_code_cannot_be_controlled_by_adversary
5. Vulnerable_code_not_in_execute_path
6. Vulnerable_code_not_present
7. Other (with space for your text)

**CycloneDX's VEX options:**

1. code_not_present
2. code_not_reachable
3. requires_configuration
4. requires_dependency
5. requires_environment
6. protected_by_compiler
7. protected_at_runtime
8. protected_at_perimeter
9. protected_by_mitigating_control
Additional references

Learn more about FOSSA:

https://github.com/fossas/fossa-cli/tree/master

https://docs.fossa.com/

https://fossa.com/blog/using-cisa-kev-catalog/