Why Can't We All Get Along?

Bridging the Gap in Vulnerability Prioritization Standards

Yotam Perkal

Where are we now?

About Me

Yotam Perkal

Head of Vulnerability Research

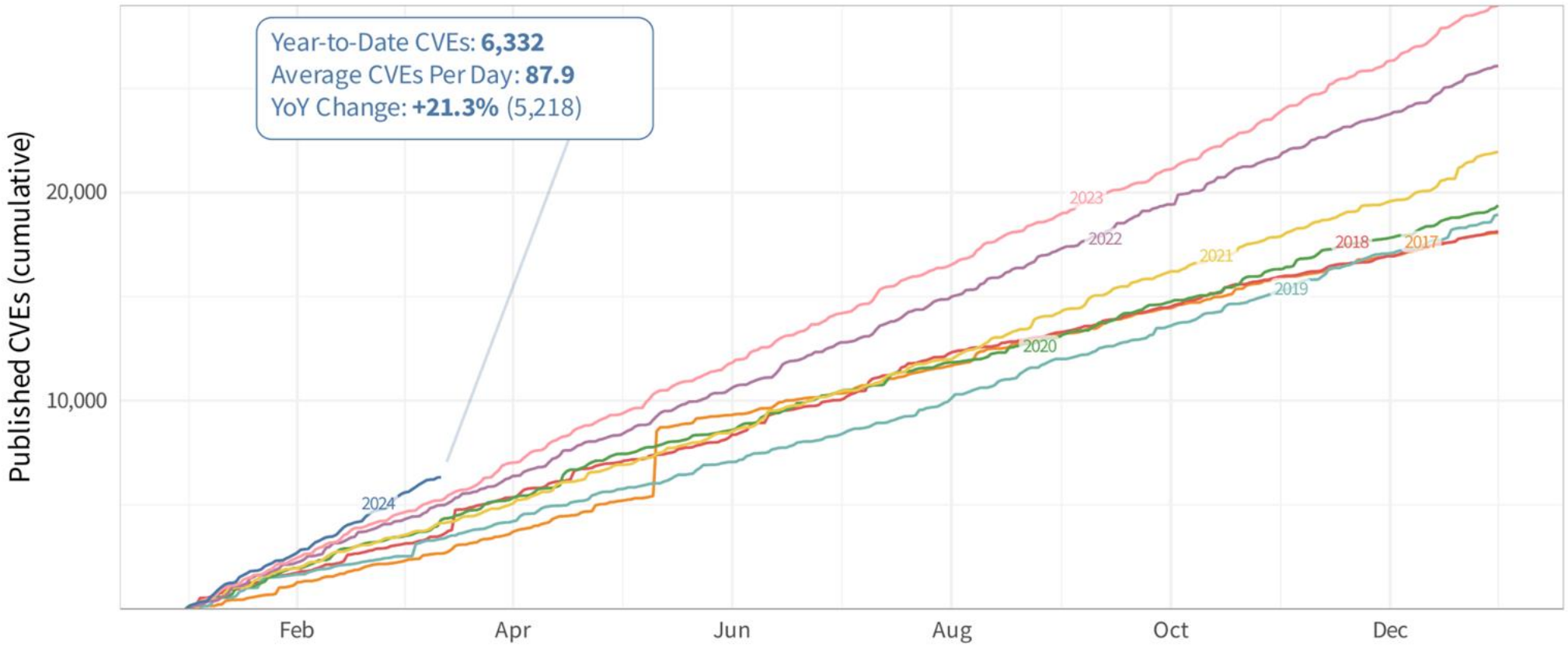
Rezilion

@pyotam2

The problem

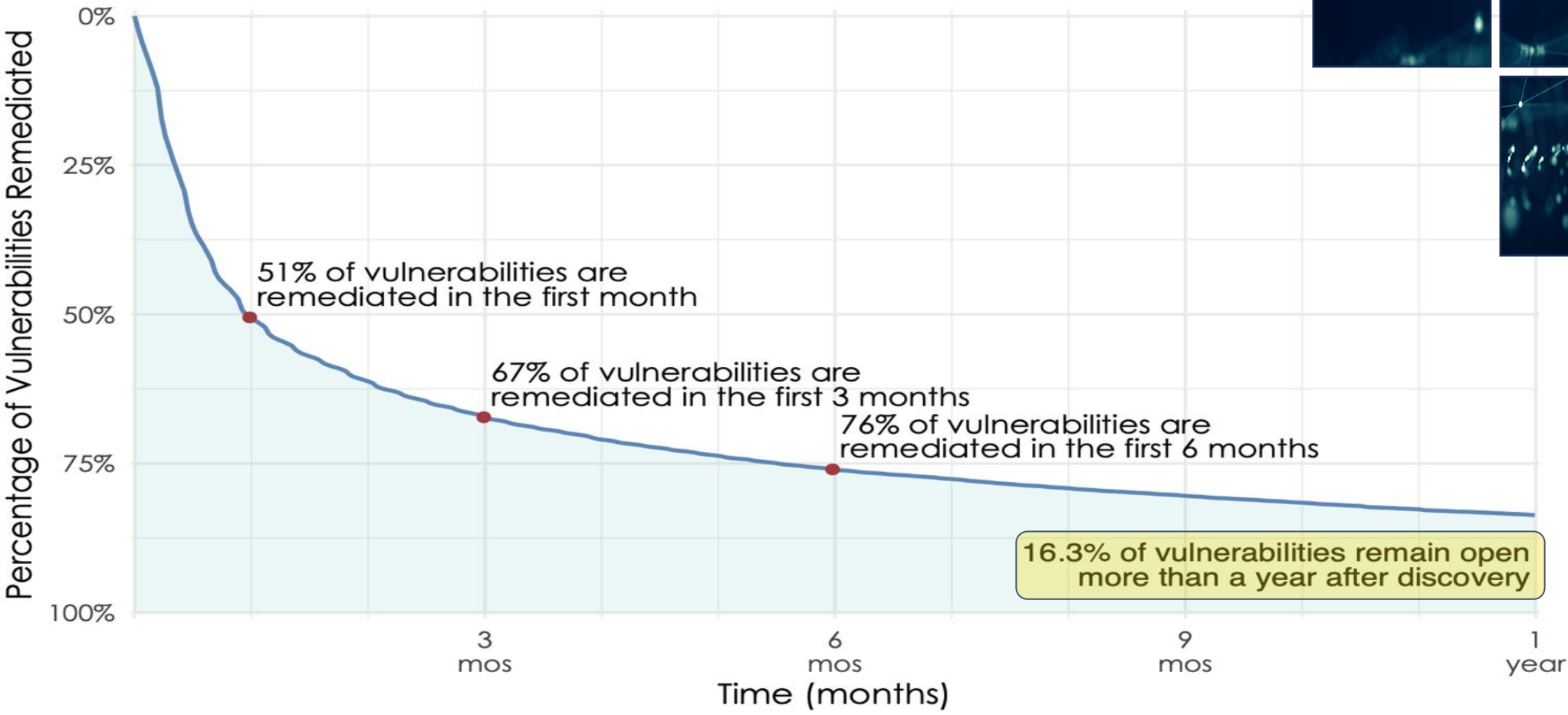
Year-to-date CVE publications (MITRE CVE List)

Lines showing the daily cumulative count of published CVEs on MITRE's CVE List, https://cve.mitre.org/cve/



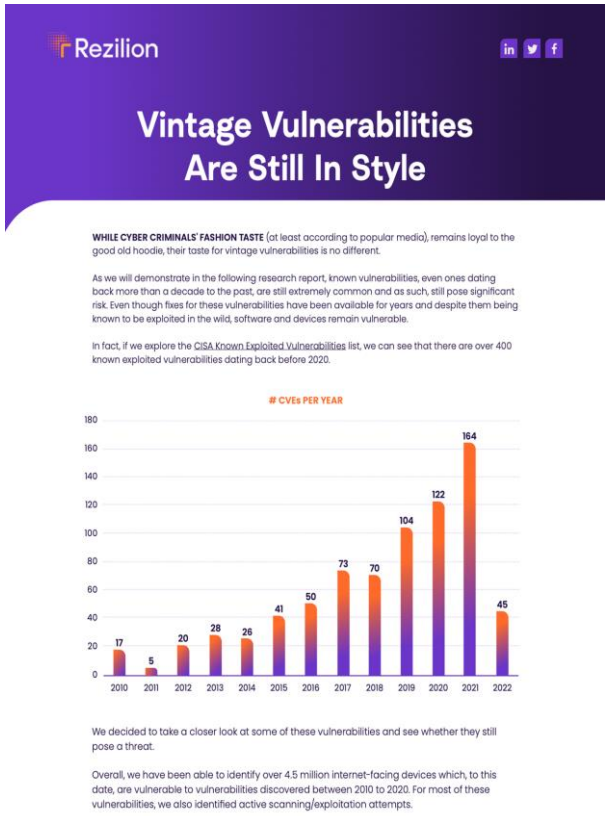
Source: https://first.org/epss/data_stats, 2024-03-12

Which leads to...



Remediation Velocity - Cyentia Institute, Patching, Fast and Slow

And Attackers Are Taking Advantage


Rezilion

Vintage Vulnerabilities Are Still In Style

WHILE CYBER CRIMINALS' FASHION TASTE (at least according to popular media), remains loyal to the good old hoodie, their taste for vintage vulnerabilities is no different.

As we will demonstrate in the following research report, known vulnerabilities, even ones dating back more than a decade to the past, are still extremely common and as such, still pose significant risk. Even though fixes for these vulnerabilities have been available for years and despite them being known to be exploited in the wild, software and devices remain vulnerable.

In fact, if we explore the CISA Known Exploited Vulnerabilities list, we can see that there are over 400 known exploited vulnerabilities dating back before 2020.

We decided to take a closer look at some of these vulnerabilities and see whether they still pose a threat.

Overall, we have been able to identify over 4.5 million internet-facing devices which, to this date, are vulnerable to vulnerabilities discovered between 2010 to 2020. For most of these vulnerabilities, we also identified active scanning/exploitation attempts.

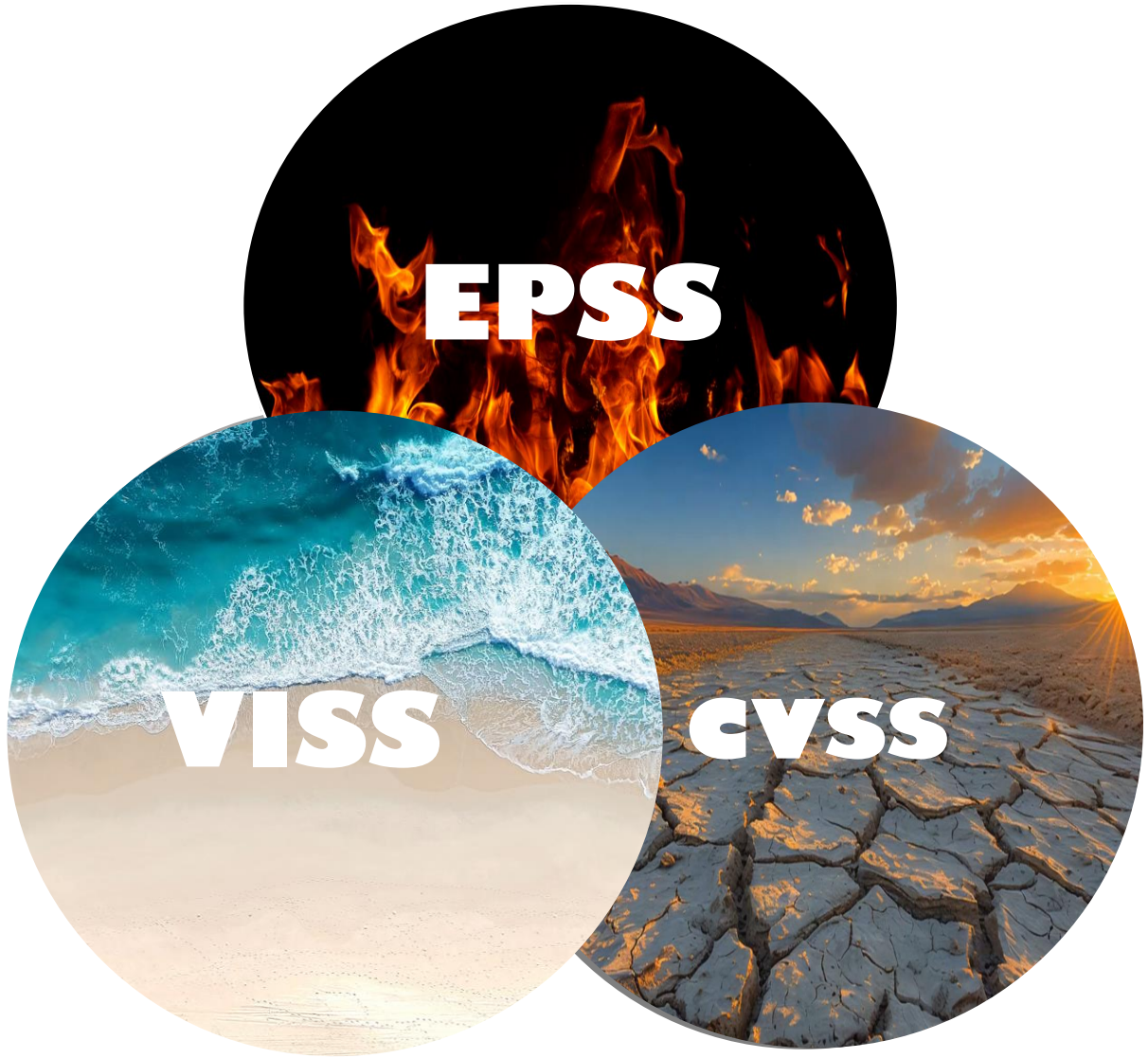
- CISA KEV contains ~46% "Vintage Vulnerabilities"
- VulnCheck KEV contains ~45% "Vintage Vulnerabilities"
- 32% of the top 100 exploited vulnerabilities on The Shadowserver Foundation are "Vintage Vulnerabilities"

Our Research found **over 15 million publicly accessible vulnerable instances** to ~200 CISA KEV CVEs catalog.

What Can We Do?

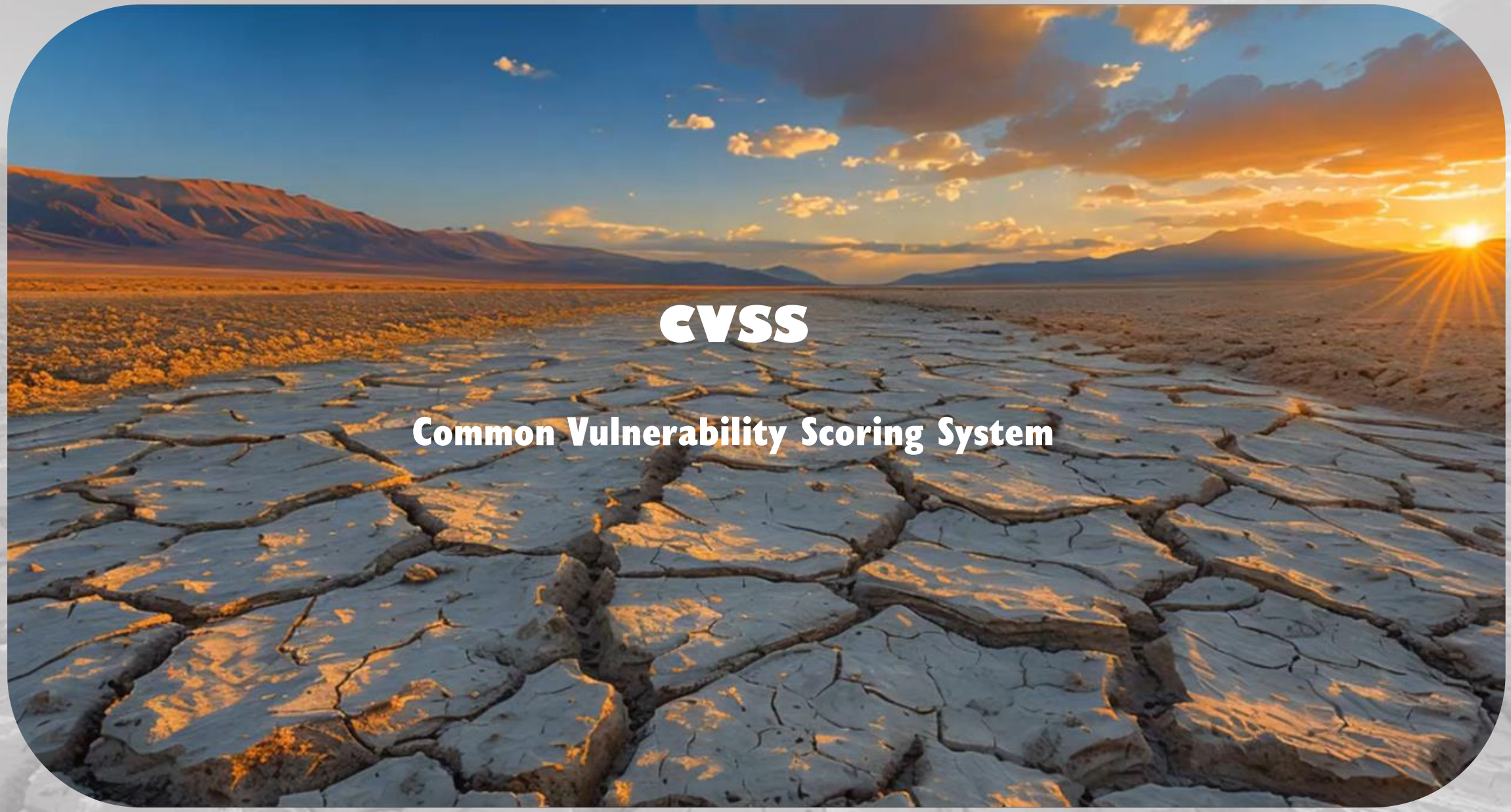
We Want to Minimize Risk

RISK



RISK = Threat X Vulnerability X Impact

CVSS


CVSS
Common Vulnerability Scoring System

CVSS - Common Vulnerability Scoring System

The CVSS framework aims to standardize communicating the severity of software vulnerabilities. It captures the principal characteristics of a vulnerability and produces a numerical score reflecting its technical severity.

- By far the most common prioritization method

CVSS - Common Vulnerability Scoring System

A single score can then be broken down into a qualitative representation:

Rating	CVSS Score
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

CVSS Version 4.0

Consists of four metric groups:

Base - represents inherent vulnerability qualities that remain constant across different environments and over time.

Threat - (known as Temporal in CVSS v3) reflects characteristics evolving in time but not necessarily across user environments. For example, the resulting CVSS score will be lower upon confirmation that the vulnerability has not been exploited and has no proof of concept code publicly available.

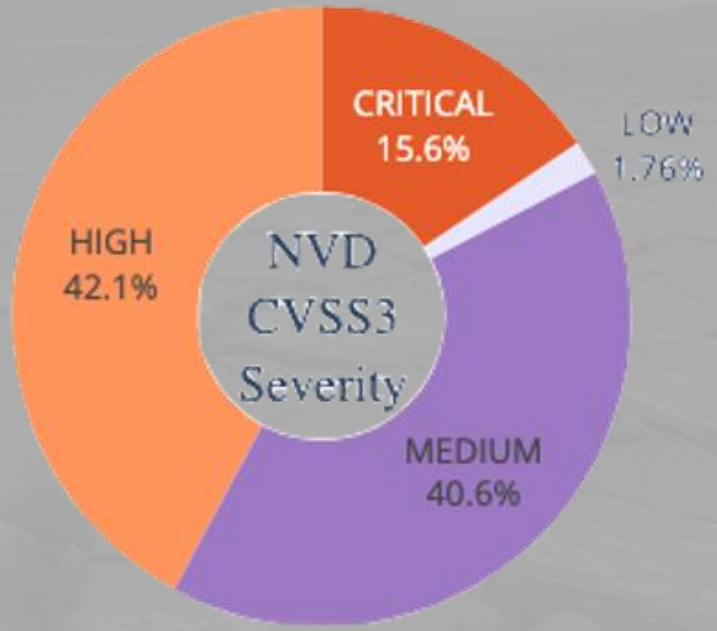
Environmental - accounts for the unique aspects of a vulnerability in the context of a specific user's environment.

Supplemental - may be used to provide extra insights but do not affect the final severity score.

CVSS Strengths

- Standardized - allowing for consistent and uniform assessment across different systems and organizations
- Widely Adopted - globally recognized and used by many organizations, including government agencies, making it a common language for discussing vulnerability severity
- A Simple Quantitative Measurement
- Open and Transparent
- Customizable

CVSS Limitations
It Isn't Scalable



It Isn't Effective

Source: https://blog.qualys.com/product-tech/2023/07/11/an-in-depth-look-at-the-latest-vulnerability-threat- landscape-part-1

It Doesn't Reflect Actual Risk

"CVSS Base scores (CVSS-B) ***represent Technical Severity*** *[which] only takes into consideration the attributes of the vulnerability itself.* ***It is NOT recommended to use this alone to determine remediation priority****."*

It Doesn't Reflect Actual Risk

CVE-2023-21237 Detail

Description

In applyRemoteView of NotificationContentInflater.java, there is a possible way to hide foreground service notification due to misleading or insufficient UI. This could lead to local information disclosure with no additional execution privileges needed. User interaction is not needed for exploitation.Product: AndroidVersions: Android-13Android ID: A-251586912

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 5.5 MEDIUM **Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

QUICK INFO

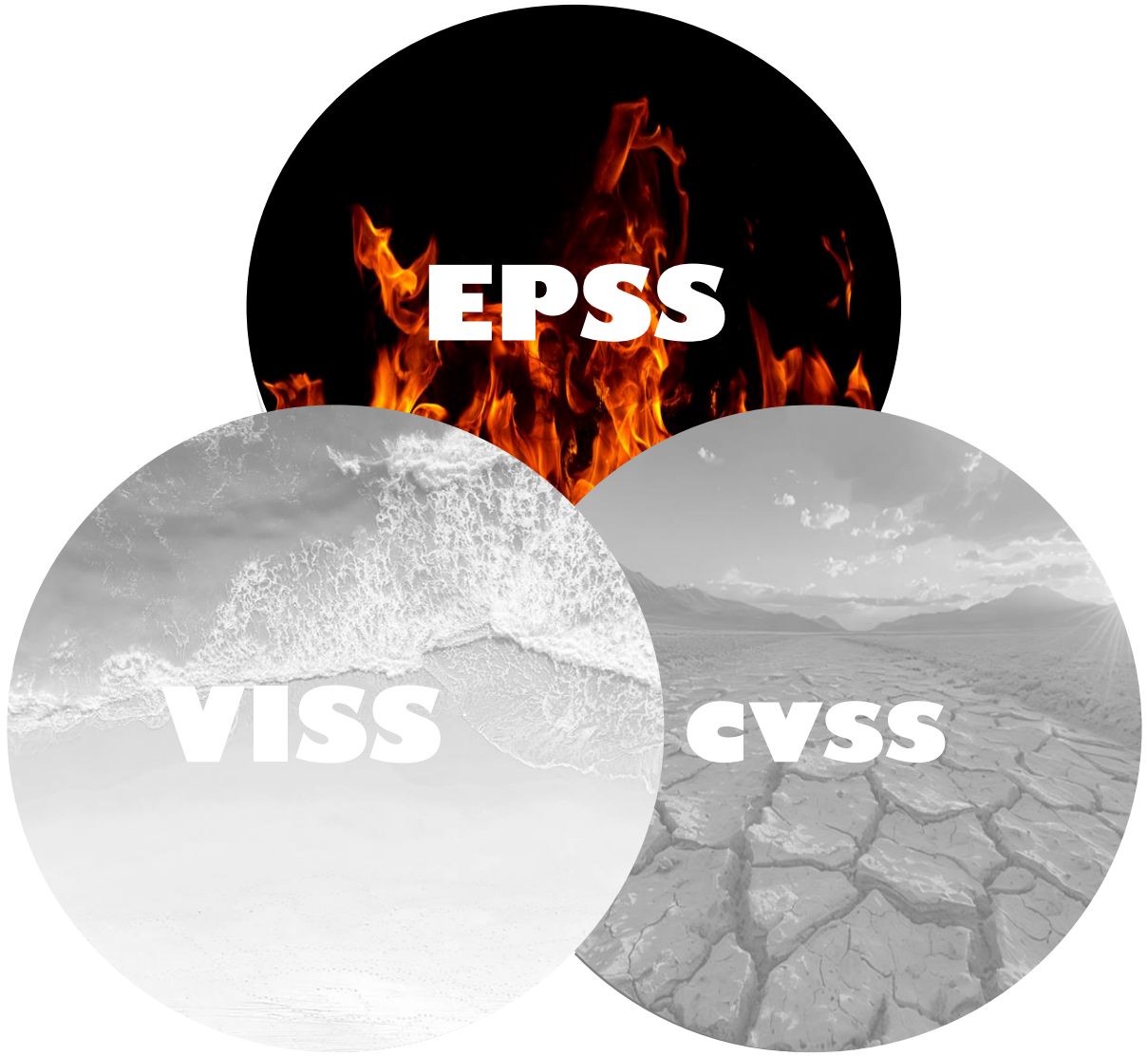
CVE Dictionary Entry:
CVE-2023-21237
NVD Published Date:
06/28/2023
NVD Last Modified:
03/05/2024
Source:
Android (associated with Google Inc. or Open Handset Alliance)

This CVE is in CISA's Known Exploited Vulnerabilities Catalog

Reference CISA's BOD 22-01 and Known Exploited Vulnerabilities Catalog for further guidance and requirements.

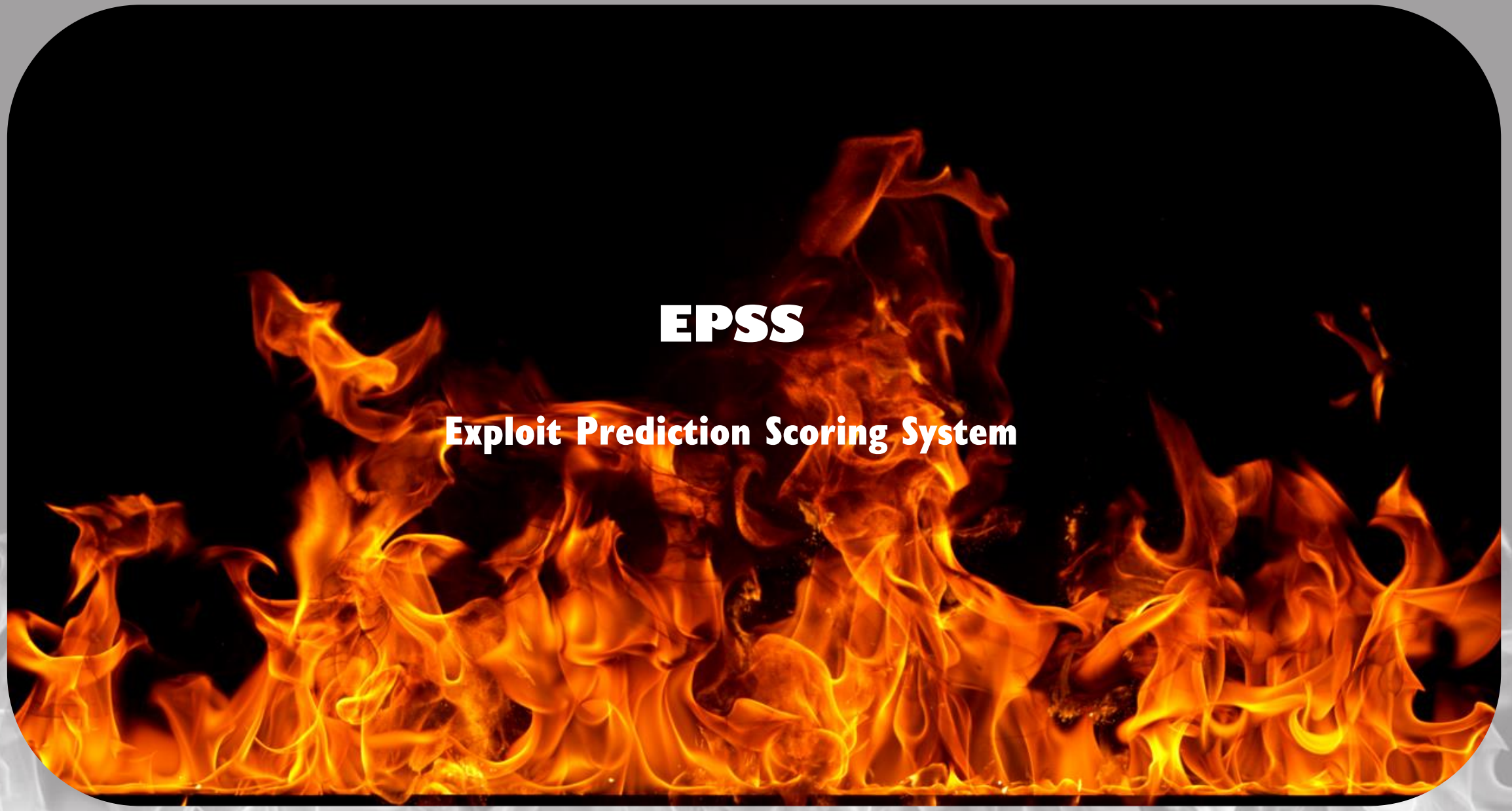
Vulnerability Name	Date Added	Due Date	Required Action
Android Pixel Information Disclosure Vulnerability	03/05/2024	03/26/2024	Apply mitigations per vendor instructions or discontinue use of the product if mitigations are unavailable.

RISK



RISK = Threat X Vulnerability X Impact

EPSS


EPSS
Exploit Prediction Scoring System

EPSS - Exploit Prediction Scoring System

The Exploit Prediction Scoring System (EPSS) is a data-driven effort for estimating the likelihood (probability) that a software vulnerability will be exploited in the wild within the next 30-day period.

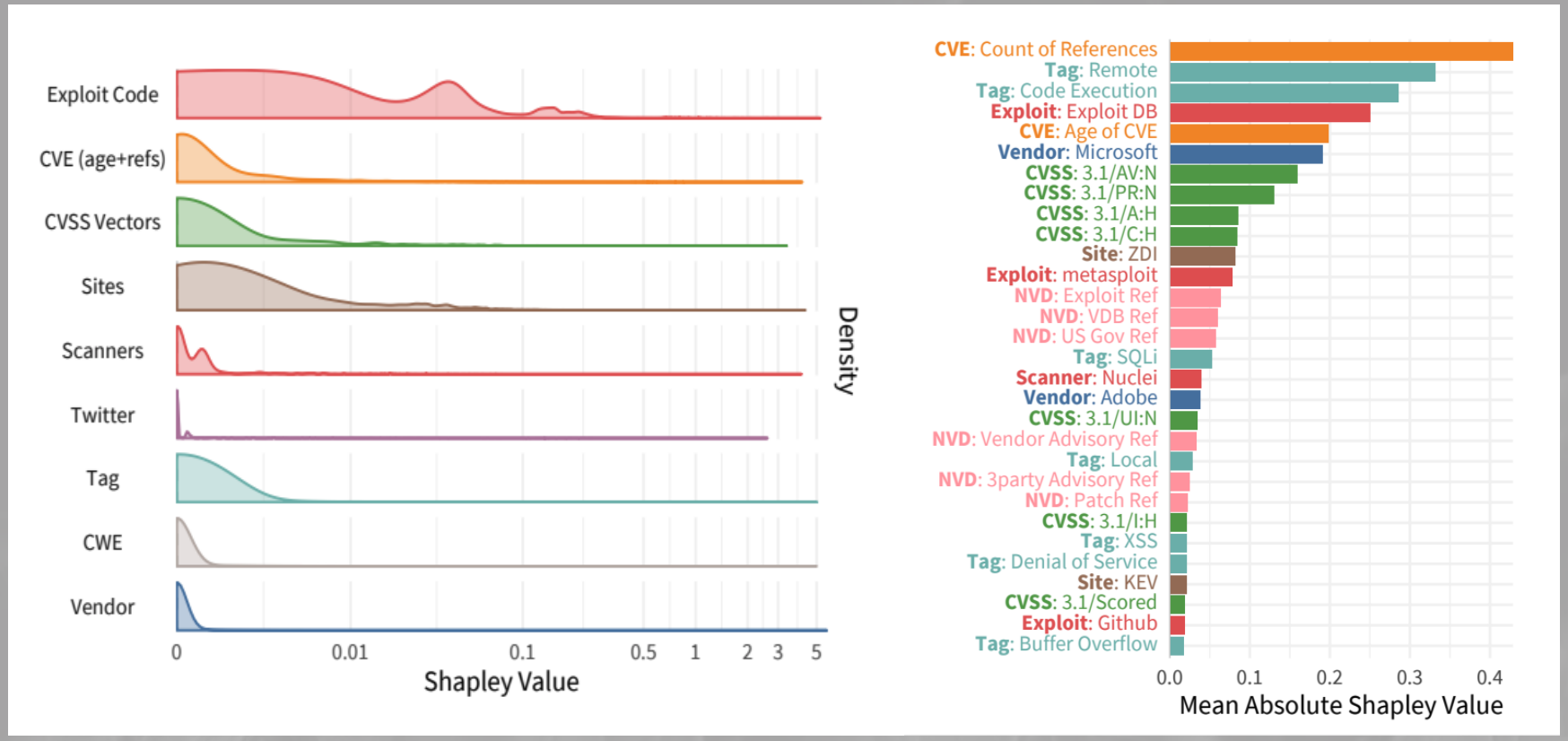
EPSS - How Does It Work?

The EPSS model relies on a dataset based on more than 6.4 million observed exploit attempts with data contributions from organizations such as:

GREYNOISE

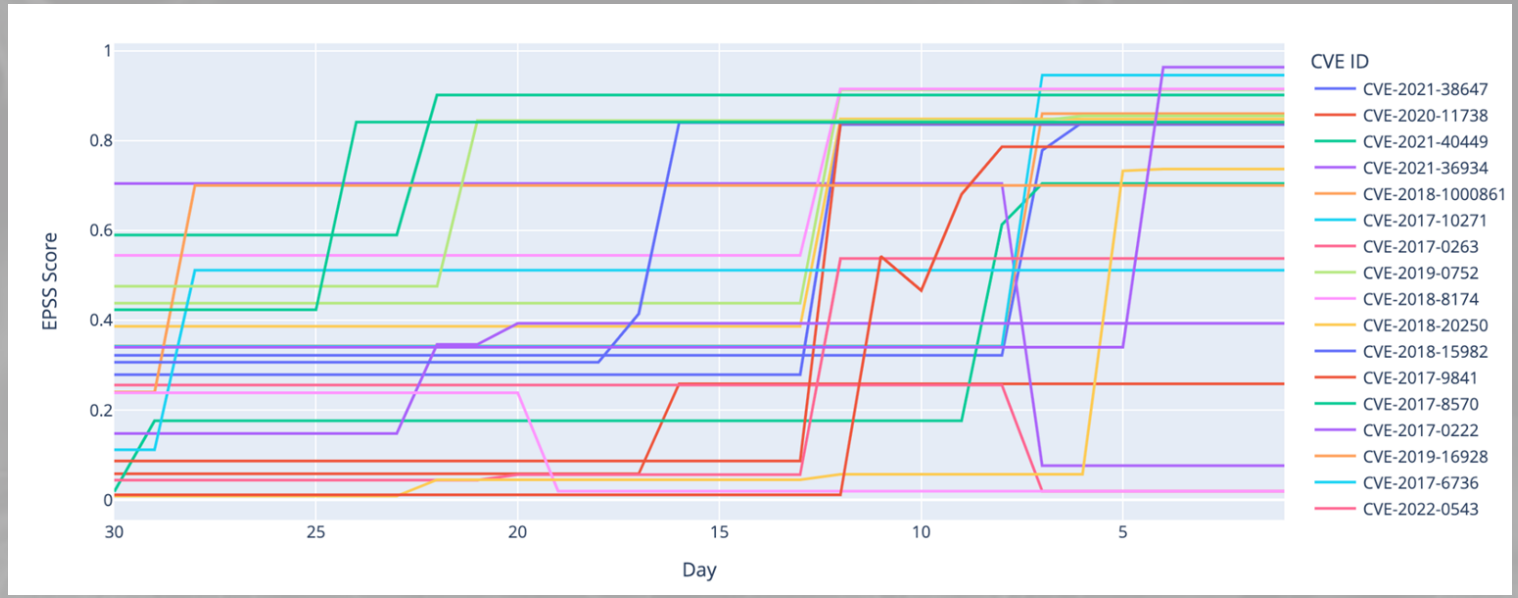
Note: An exploit attempt is defined as any recorded attempt to exploit a vulnerability, regardless of whether the attempt was successful.

EPSS - How Does It Work?



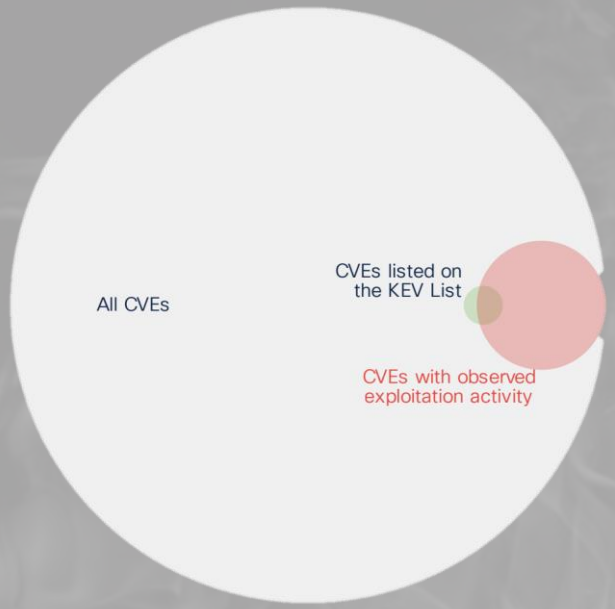
EPSS Strengths

Temporal Aspect - Adapts to new information published after the initial disclosure of a vulnerability.



EPSS Strengths

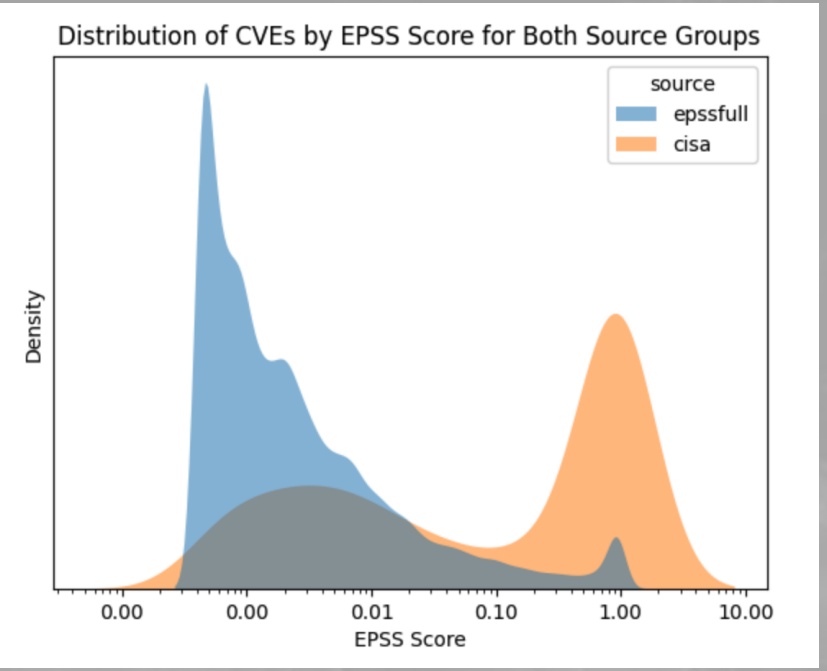
High Coverage compared to most Threat Intelligence feeds



source: Prioritization to prediction, vol. 9

EPSS Strengths

Strong Signal



EPSS Limitations

Reliance on cve.org

- Timeliness. May not capture vulnerabilities exploited before they appear on the CVE List (a process that can sometimes take weeks).
- Coverage. Vulnerabilities without a CVE ID or which are listed on other platforms such as GitHub Security Advisory (GHSA) database or the Global Security Database (GSD) will currently not receive an EPSS score

A Probabilistic Model

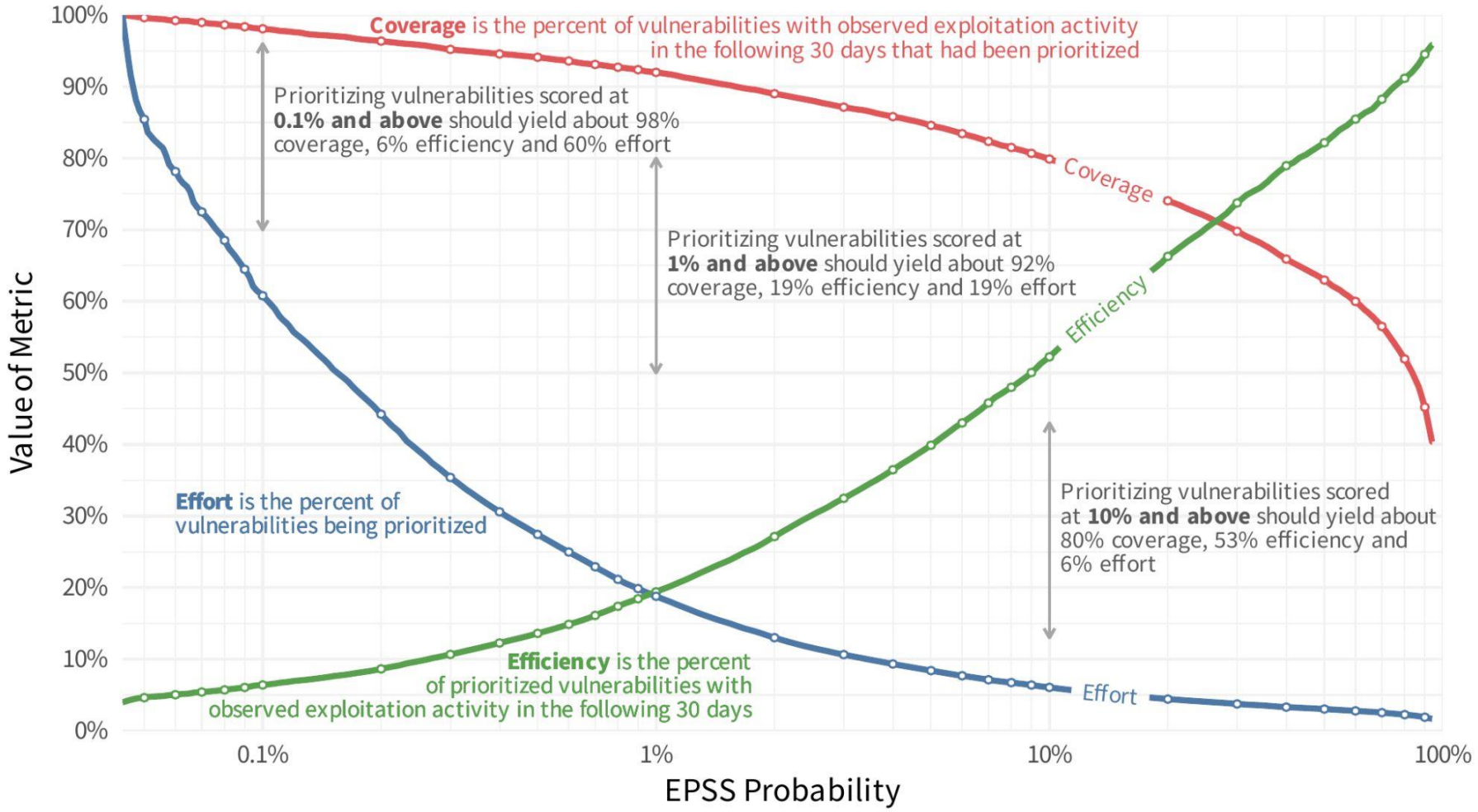
- No definitive "Yes"/"No" answers
- There is a level of uncertainty in the predictions. An actively exploited CVE might potentialy get a "low" EPSS score

Training Data Limitations/Bias

- While EPSS relies on a broad dataset of exploitation data, it is limited to observed exploitation attempts.
- Bias Toward Network-Based Attacks.
- Inclusion of research activity. Some of the threat-intelligence data sources EPSS relies on might contain benign (non-malicious) scans being conducted for research purposes
- Adversarial Considerations

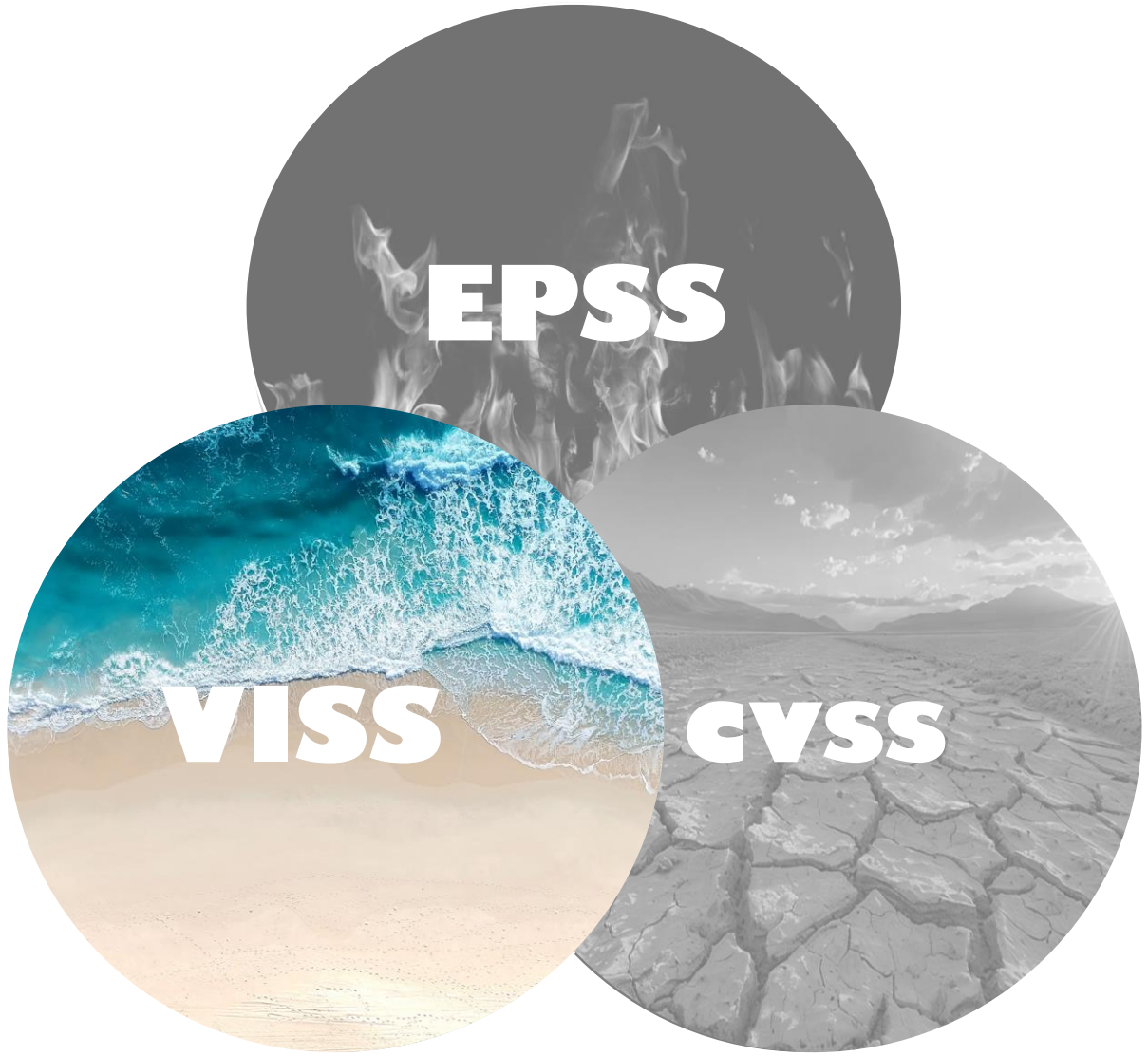
Picking Thresholds for EPSS

Select a threshold for EPSS along the horizontal and trace it to each metric to determine the coverage, efficiency and level of effort. This represents the performance of EPSS from March 7 to November 1, 2023.



Source: https://first.org/epss

RISK



RISK = Threat X Vulnerability X Impact

VISS

VISS

Vulnerability Impact Scoring System

VISS - Vulnerability Impact Scoring System

A customizable framework for evaluating the **impact** of security vulnerabilities.

Assumes the same vulnerability can impact different environments/organizations differently.

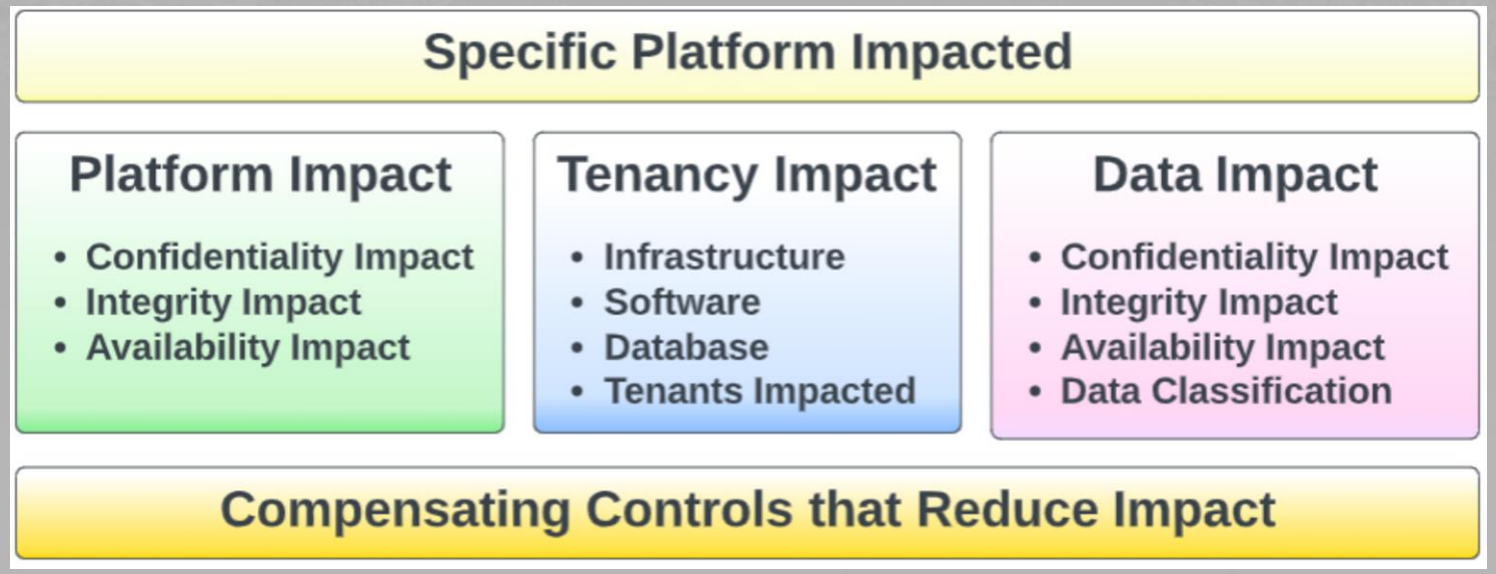
Free to use with proper attribution, conformance to the guidelines and providence of both the score and the scoring vector

VISS scores are typically produced by the organization maintaining the system, environment, network, or product in which a vulnerability has been identified, or an external party performing the evaluation on their behalf, such as a bug bounty triage team.

“VISS is not meant as a replacement for CVSS, but rather is a complementary system of evaluation from a different perspective”

VISS - Vulnerability Impact Scoring System

VISS analyzes 13 different aspects of impact for each vulnerability, segmented into impact groups:

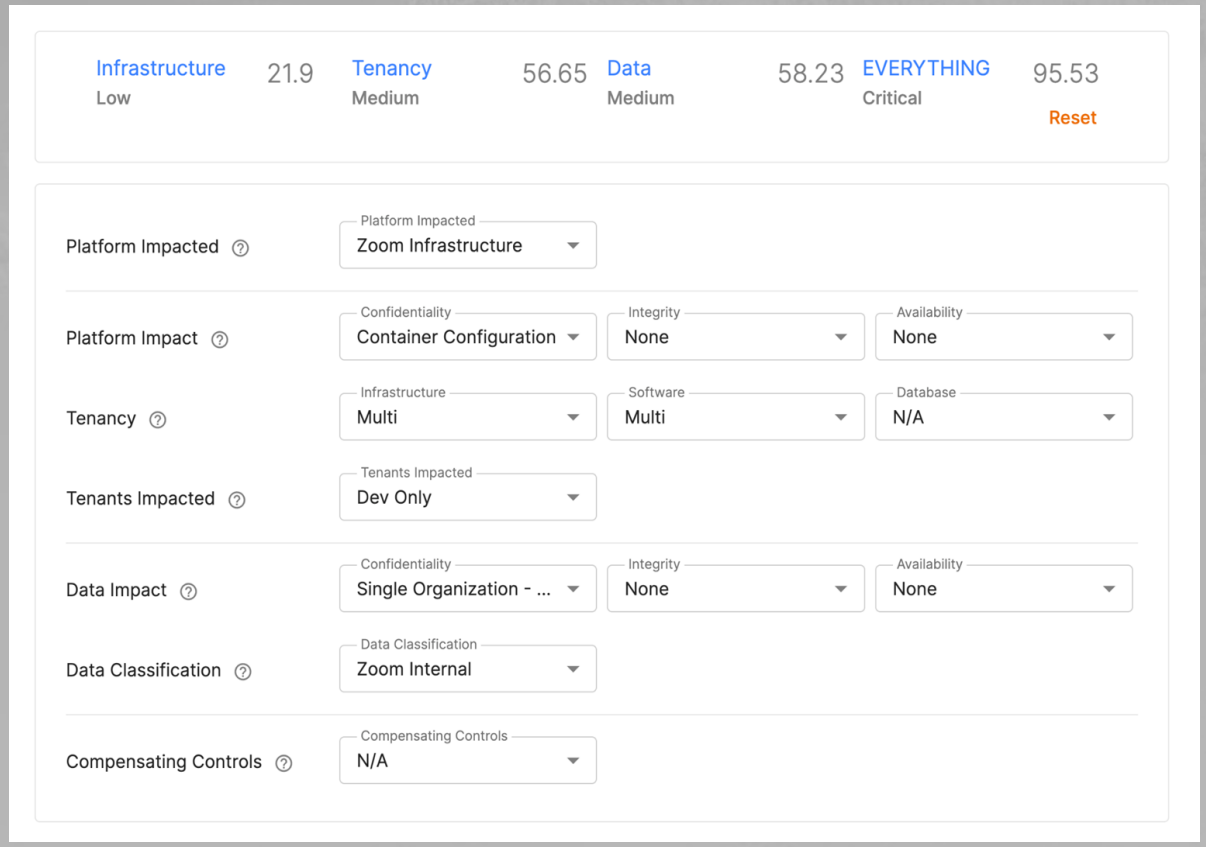


VISS - Vulnerability Impact Scoring System

The VISS score is calculated using a set of equations that take into account the weight assigned to each variable and their relation and impact on each other.

Rating	VISS Score
None	0 - 9
Low	9.01 - 39
Medium	39.01 - 69
High	69.01 - 89
Critical	89.01 - 100

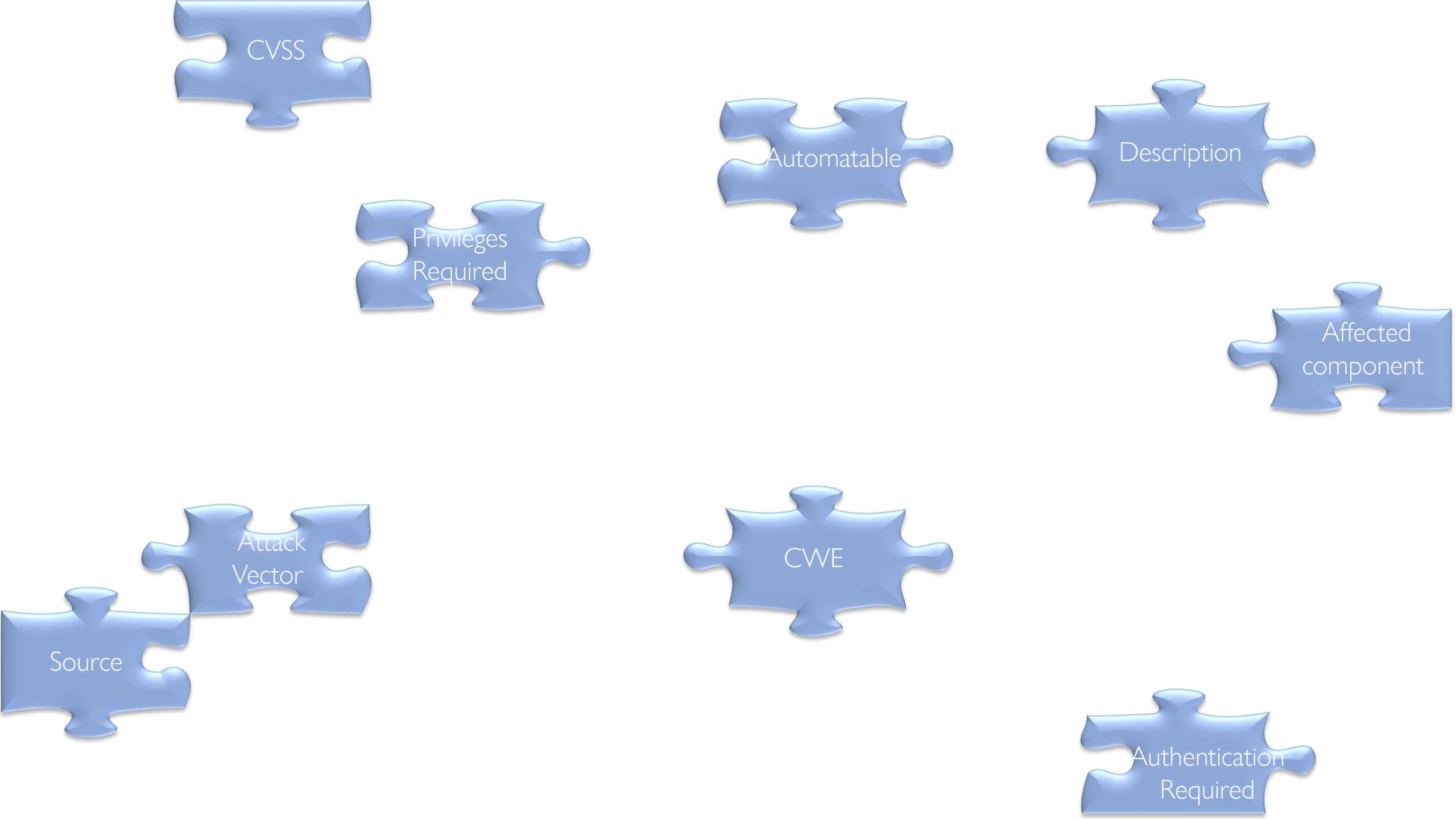
VISS - Vulnerability Impact Scoring System

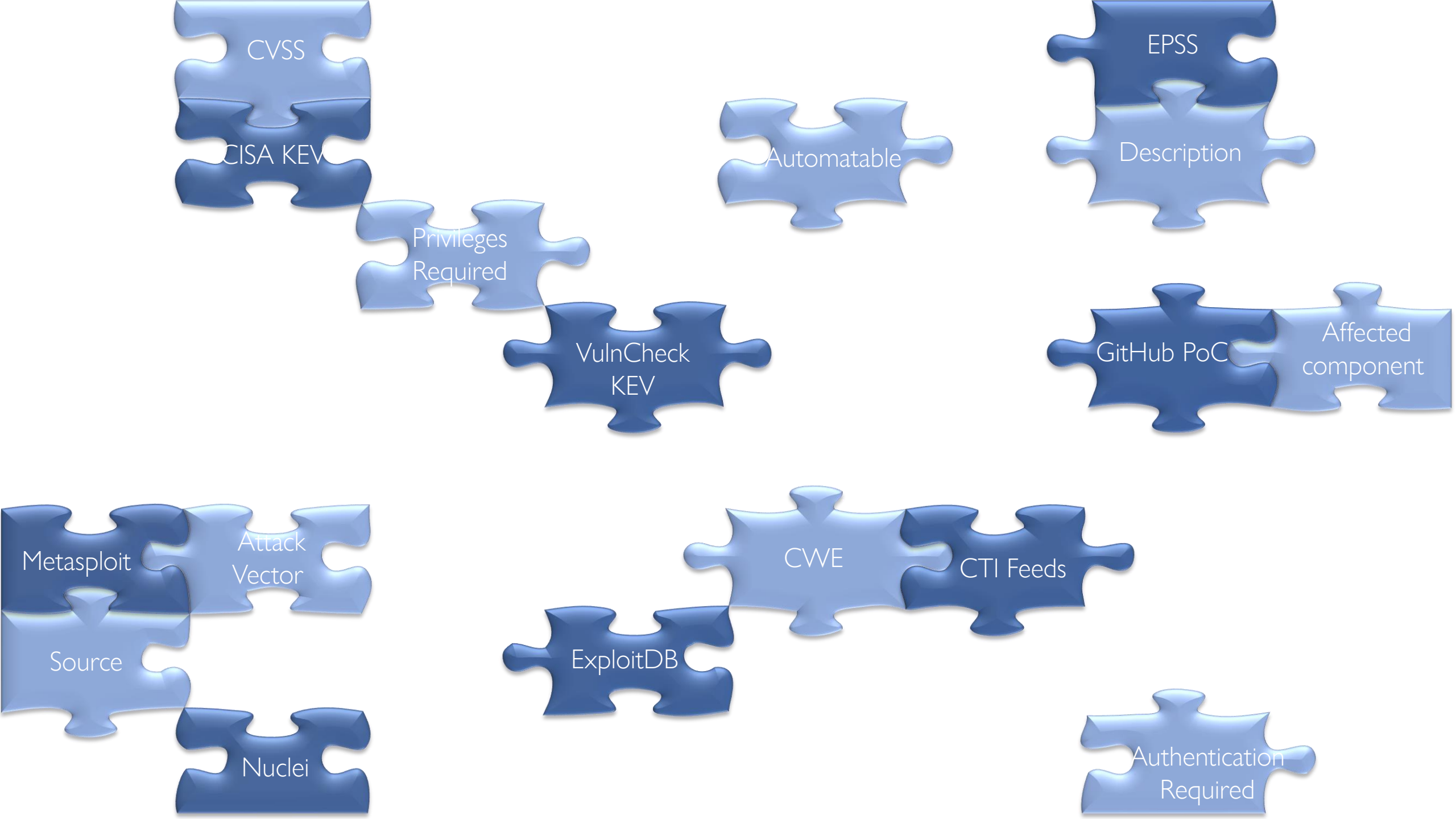


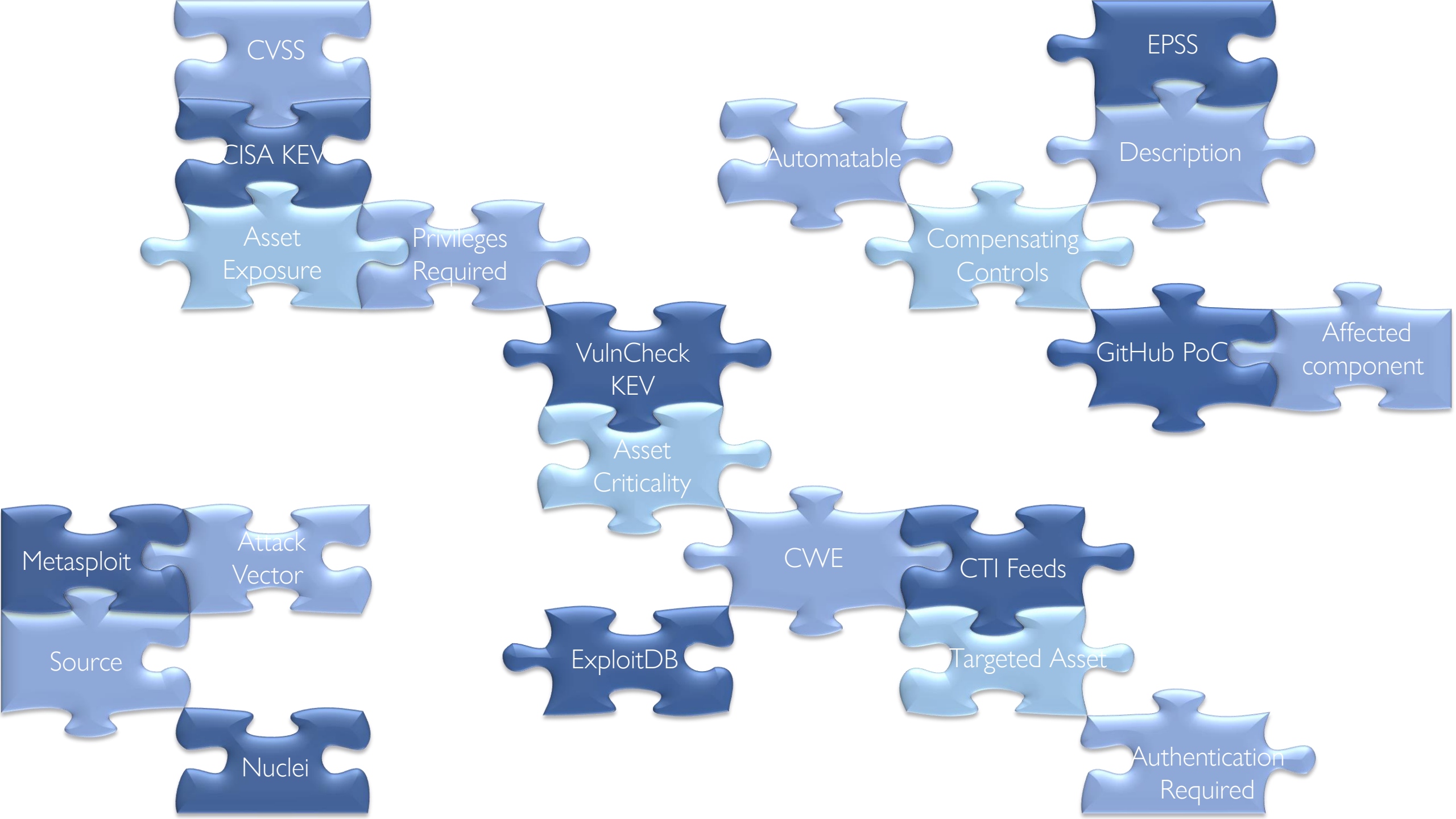
VISS Limitations

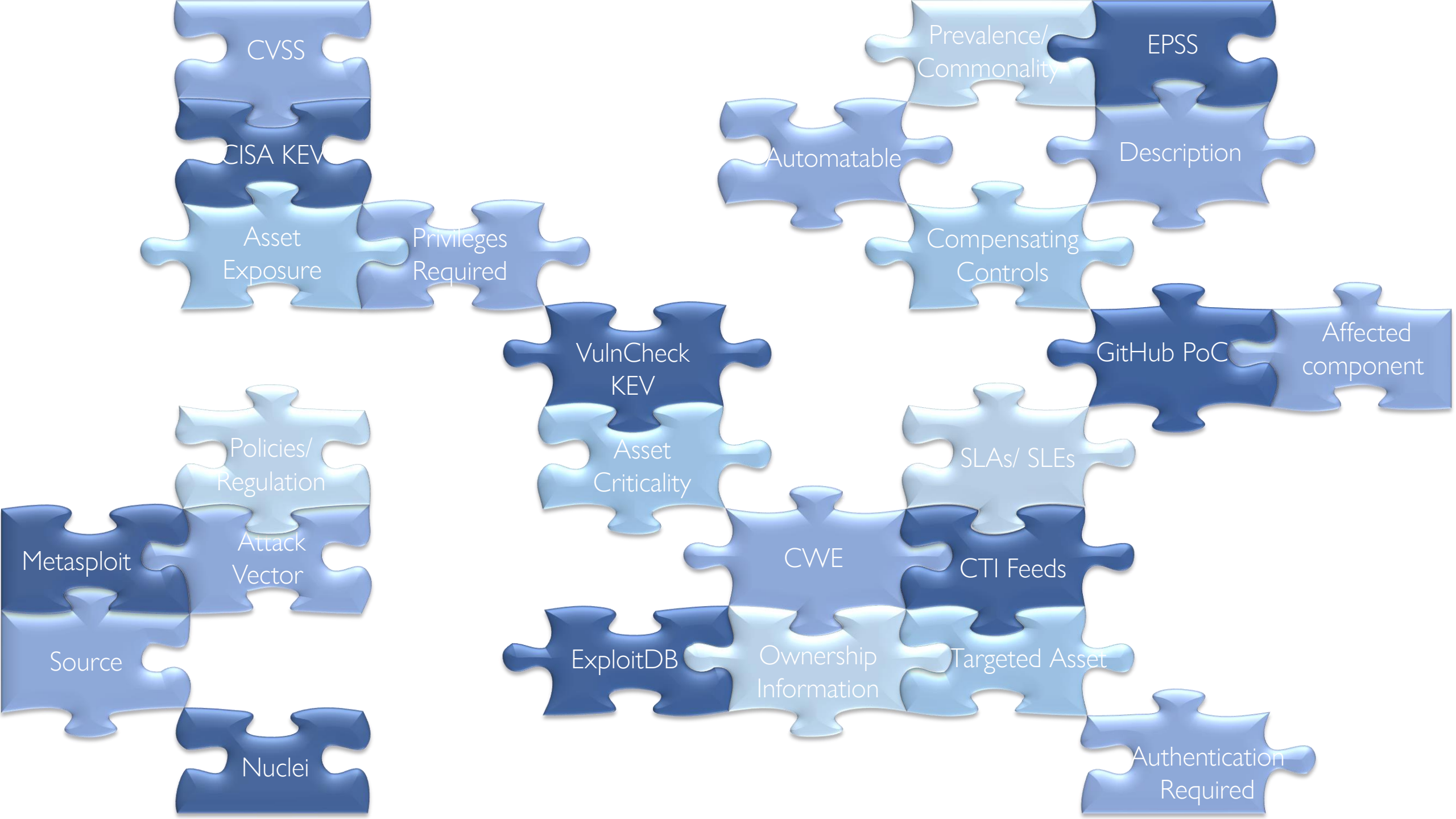
- No proven track record, not yet widely adopted
- Requires human Analysis
- Lacking several aspects of impact such as: number of affected customers, potential financial loss

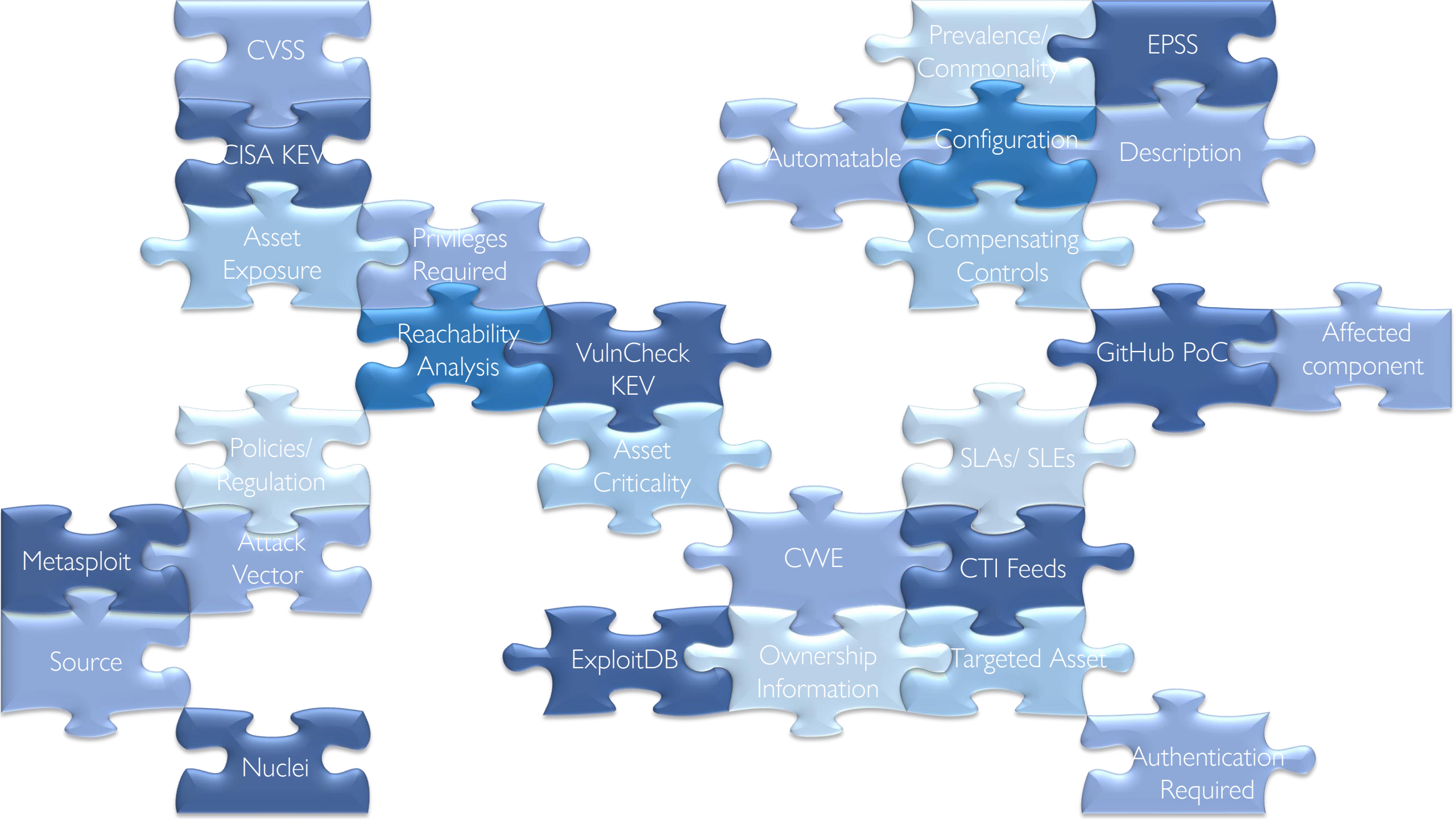
Additional Context

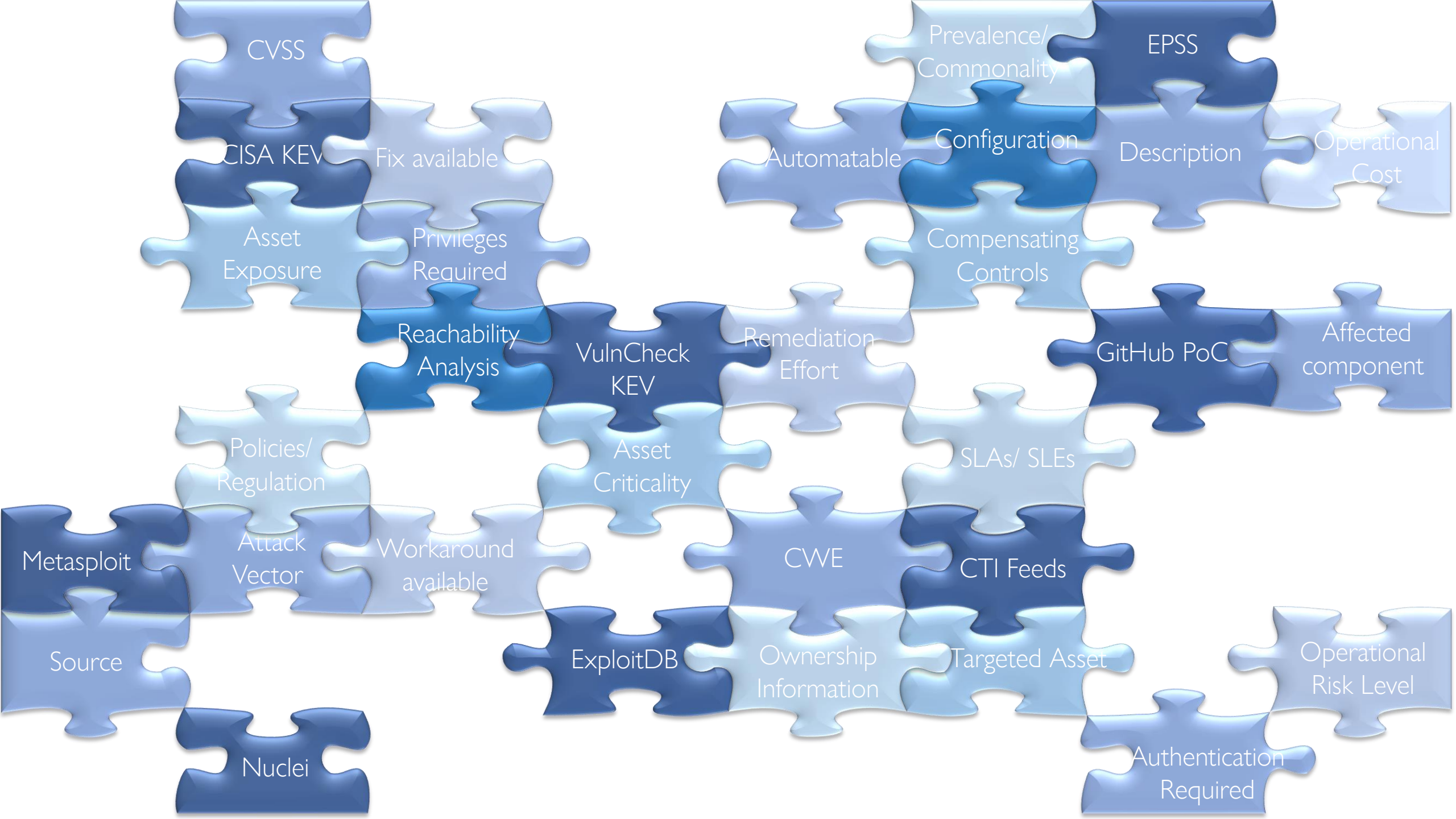

CVSS
Automatable
Description
Privileges Required
Affected component
Attack Vector
CWE
Source
Authentication Required


CVSS
CISA KEV
Automatable
EPSS
Description
Privileges Required
VulnCheck KEV
GitHub PoC
Affected component
Metasploit
Attack Vector
Source
Nuclei
CWE
CTI Feeds
ExploitDB
Authentication Required


CVSS
CISA KEV
Asset Exposure
Privileges Required
Automatable
EPSS
Description
Compensating Controls
GitHub PoC
Affected component
VulnCheck KEV
Asset Criticality
Metasploit
Attack Vector
Source
Nuclei
CWE
CTI Feeds
ExploitDB
Targeted Asset
Authentication Required


CVSS
CISA KEV
Asset Exposure
Privileges Required
Prevalence/ Commonality
EPSS
Automatable
Description
Compensating Controls
VulnCheck KEV
GitHub PoC
Affected component
Policies/ Regulation
Asset Criticality
SLAs/ SLEs
Metasploit
Attack Vector
CWE
CTI Feeds
Source
ExploitDB
Ownership Information
Targeted Asset
Nuclei
Authentication Required


CVSS
CISA KEV
Asset Exposure
Privileges Required
Reachability Analysis
VulnCheck KEV
Asset Criticality
Policies/ Regulation
Attack Vector
Metasploit
Source
Nuclei
Prevalence/ Commonality
EPSS
Automatable
Configuration
Description
Compensating Controls
GitHub PoC
Affected component
SLAs/ SLEs
CWE
CTI Feeds
ExploitDB
Ownership Information
Targeted Asset
Authentication Required


CVSS
Prevalence/ Commonality
EPSS
CISA KEV
Fix available
Automatable
Configuration
Description
Operational Cost
Asset Exposure
Privileges Required
Compensating Controls
Reachability Analysis
VulnCheck KEV
Remediation Effort
GitHub PoC
Affected component
Policies/ Regulation
Asset Criticality
SLAs/ SLEs
Metasploit
Attack Vector
Workaround available
CWE
CTI Feeds
Source
ExploitDB
Ownership Information
Targeted Asset
Operational Risk Level
Nuclei
Authentication Required


CVSS
Qualys TruRisk
Prevalence/ Commonality
EPSS
CISA KEV
Fix available
Automatable
Configuration
Description
Operational Cost
Asset Exposure
Privileges Required
Compensating Controls
Coalition ESS
Reachability Analysis
VulnCheck KEV
Remediation Effort
GitHub PoC
Affected component
Policies/ Regulation
Asset Criticality
SLAs/ SLEs
Metasploit
Attack Vector
Workaround available
CWE
CTI Feeds
Source
ExploitDB
Ownership Information
Targeted Asset
Operational Risk Level
Nuclei
Tenable VPR
Authentication Required


CVSS
Qualys TruRisk
Prevalence/ Commonality
EPSS
CISA KEV
Fix available
Automatable
Configuration
Description
Operational Cost
Asset Exposure
Privileges Required
Compensating Controls
Coalition ESS
Reachability Analysis
VulnCheck KEV
Remediation Effort
GitHub PoC
Affected component
Policies/ Regulation
Asset Criticality
SLAs/ SLEs
Metasploit
Attack Vector
Workaround available
CWE
CTI Feeds
Source
ExploitDB
Ownership Information
Targeted Asset
Operational Risk Level
Nuclei
Tenable VPR
Authentication Required


How can we create a process that takes all of these aspects into account?


Enters SSVC

SSVC - Stakeholder Specific Vulnerability Categorization

A methodology for prioritizing vulnerabilities based on the needs of the stakeholders involved in the vulnerability management process. **SSVC is designed to be used by any stakeholder in the vulnerability management process**, including finders, vendors, coordinators, deployers, and others.

Created in 2019, by the Carnegie Mellon University Software Engineering Institute (SEI), in collaboration with CISA.

It's Important to differentiate between CISA's SSVC process and SSVC as a framework.

Stakeholder Specific Vulnerability Categorization (SSVC)

- Different stakeholders can have different decision intersections, have different risk tolerance, and have access to different information
- Allows for communication of the decision process to internal and external stakeholders
- Transparency into the decision process
- A ground for discussion
- The process can include interviewing Stakeholders within the organization

SSVC - Core Concepts

- Stakeholders - Different participants in the vulnerability response process have different needs and priorities:
    - Suppliers
    - Deployers
    - Coordinators
- Decisions - Each decision is made based on a set of inputs, or decision points which should be, independent, discrete, and well-defined.
- Outcomes - Each decision has a set of possible outcomes
- A Policy - A mapping from each combination of decision point values to the set of outcome values.

SSVC - Strengths

- Transparent
- Explainable
- Modular
- Can take into account multiple facets of risk

SSVC In Practice

Example #1 - CISA Coordinator

Exploitation

None

There is no evidence of active exploitation and no public proof of concept (PoC) of how to exploit the vulnerability.

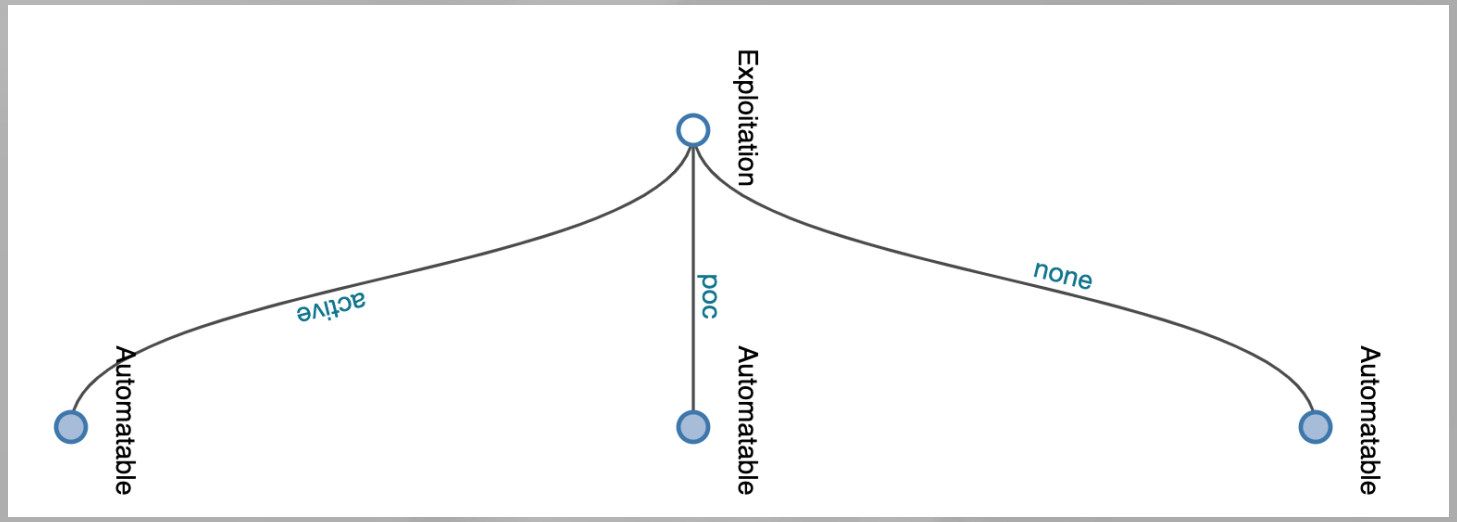
Poc

One of the following cases is true: (1) private evidence of exploitation is attested but not shared; (2) widespread hearsay attests to exploitation; (3) typical public PoC in places such as Metasploit or ExploitDB; or (4) the vulnerability has a well-known method of exploitation. Some examples of condition (4) are open-source web proxies serve as the PoC code for how to exploit any vulnerability in the vein of improper validation of TLS certificates. As another example, Wireshark serves as a PoC for packet replay attacks on ethernet or WiFi networks.

Active

Shared, observable, reliable evidence that the exploit is being used in the wild by real attackers; there is credible public reporting.

Example #1 - CISA Coordinator



Example #1 - CISA Coordinator

Automatable

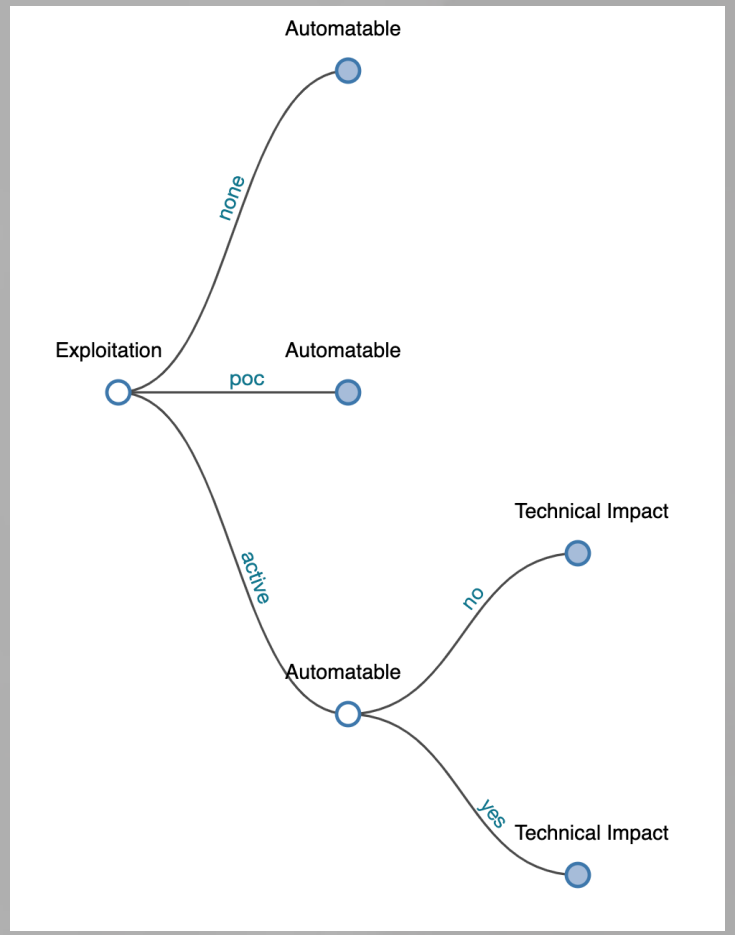
No

Steps 1-4 of the kill chain cannot be reliably automated for this vulnerability for some reason. These steps are reconnaissance, weaponization, delivery, and exploitation. Example reasons for why a step may not be reliably automatable include (1) the vulnerable component is not searchable or enumerable on the network, (2) weaponization may require human direction for each target, (3) delivery may require channels that widely deployed network security configurations block, and (4) exploitation may be frustrated by adequate exploit-prevention techniques enabled by default; ASLR is an example of an exploit-prevention tool.

Yes

Steps 1-4 of the of the kill chain can be reliably automated. If the vulnerability allows unauthenticated remote code execution (RCE) or command injection, the response is likely yes.

Example #1 - CISA Coordinator



Example #1 - CISA Coordinator

Technical Impact

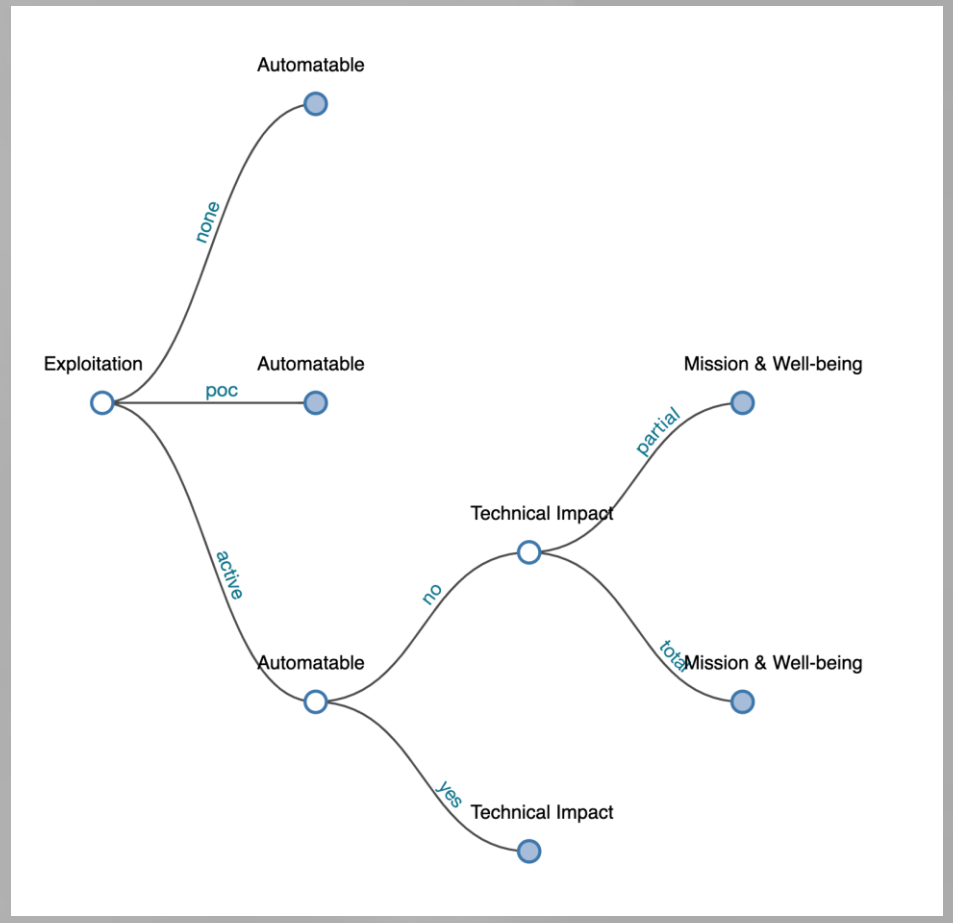
Partial

The exploit gives the adversary limited control over, or information exposure about, the behavior of the software that contains the vulnerability. Or the exploit gives the adversary an importantly low stochastic opportunity for total control. In this context, "low" means that the attacker cannot reasona-bly make enough attempts to overcome the low chance of each attempt not working. Denial of service is a form of limited control over the behavior of the vulnerable component.

Total

The exploit gives the adversary total control over the behavior of the software, or it gives total disclosure of all information on the system that contains the vulnerability.

Example #1 - CISA Coordinator



Example #1 - CISA Coordinator

Mission & Well-being

(Complex Decision)

Low

Mission Prevalence is Low and Public well-being impact is Minimal

Medium

Mission Prevalence is Medium and Public well-being impact is in Material

High

Mission Prevalence is Essential and Public well-being impact is Irreversible

Depends on 1

Public Well-being Impact

Minimal

Type of harm is "All" (Physical, Environmental,Financial,Psychological). The effect is below the threshold for all aspects described in material.

Material

Any one or more of the conditions (Physical, Environmental,Financial,Psychological) hold. "Physical harm" means "Physical distress or injuries for users of the system OR introduces occupational safety hazards OR reduction and/or failure of cyber-physical system's safety margins." "Environment" means "Major externalities (property damage, environmental damage, etc.) imposed on other parties." "Financial" means "Financial losses that likely lead to bankruptcy of multiple persons." "Psychological" means "Widespread emotional or psychological harm, sufficient to be cause for counselling or therapy, to populations of people."

Irreversible

Any one or more of the following conditions hold. "Physical harm" means "Multiple fatalities likely OR loss or destruction of cyber-physical system of which the vulnerable component is a part." "Environment" means "Extreme or serious externalities (immediate public health threat, environmental damage leading to small ecosystem collapse, etc.) imposed on other parties." "Financial" means "Social systems (elections, financial grid, etc.) supported by the software are destabilized and potentially collapse."

Mission Prevalence

Minimal

Neither support nor essential apply. The vulnerable component may be used within the entities, but it is not used as a mission-essential component nor does it support (enough) mission essential functions.

Support

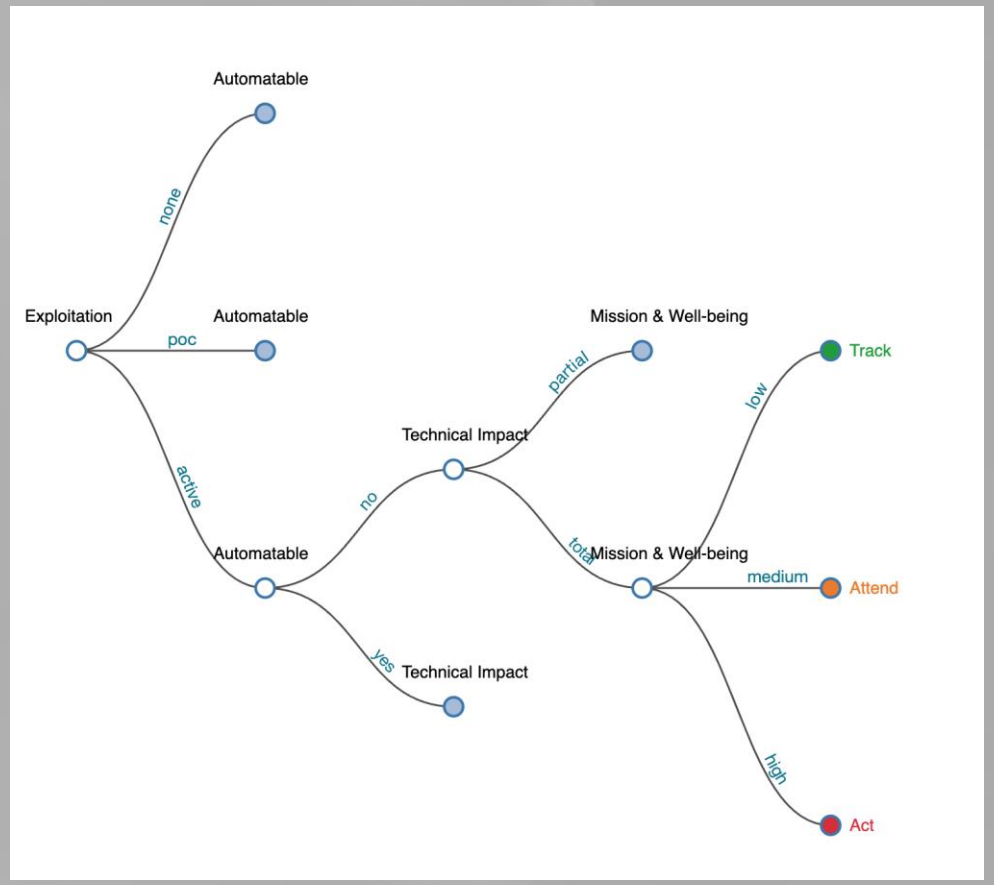
The operation of the vulnerable component merely supports mission essential functions for two or more entities.

Essential

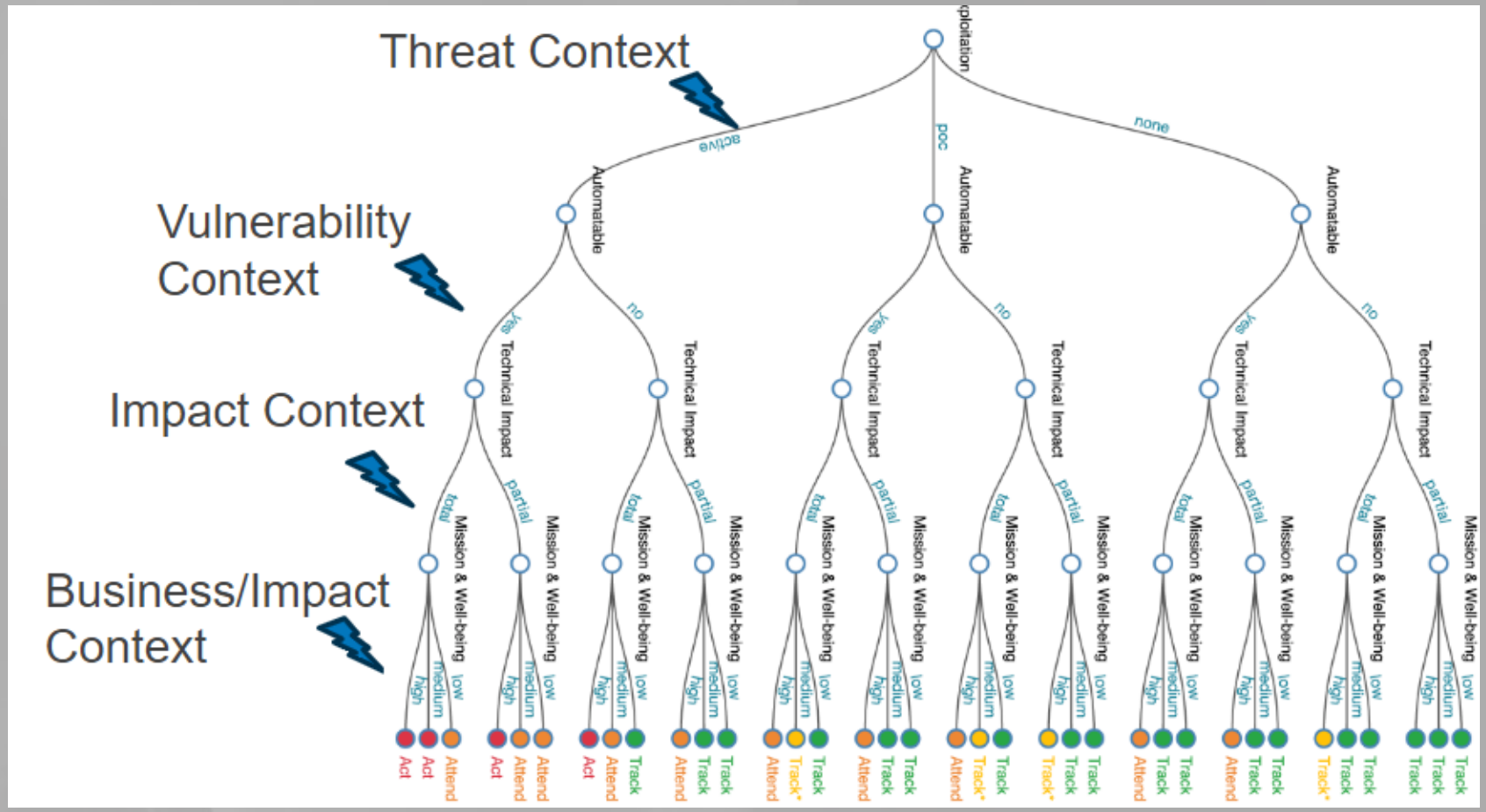
The vulnerable component directly provides capabilities that constitute at least one MEF for at least one entity, and failure may (but need not) lead to overall mission failure.

Depends on 2

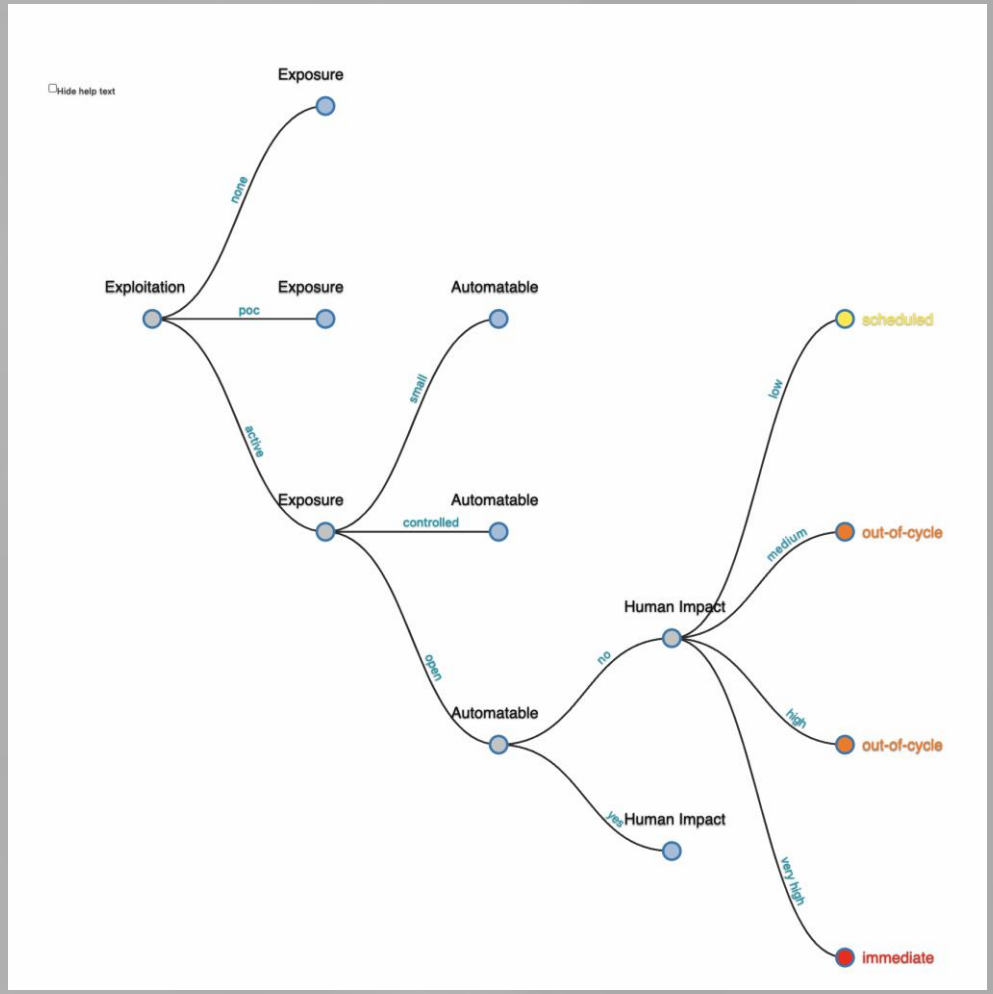
Example #1 - CISA Coordinator



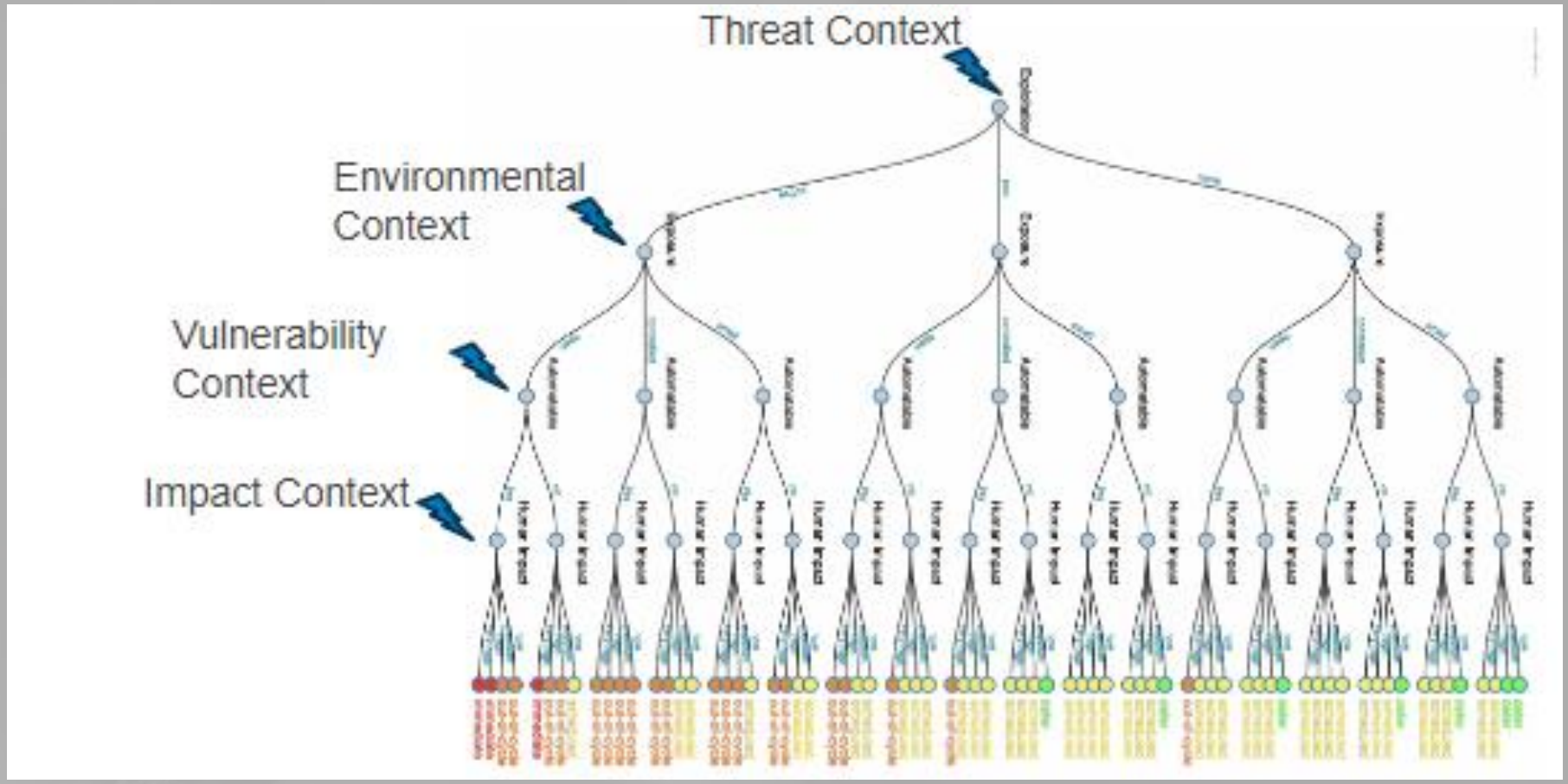
Example #1 - CISA Coordinator



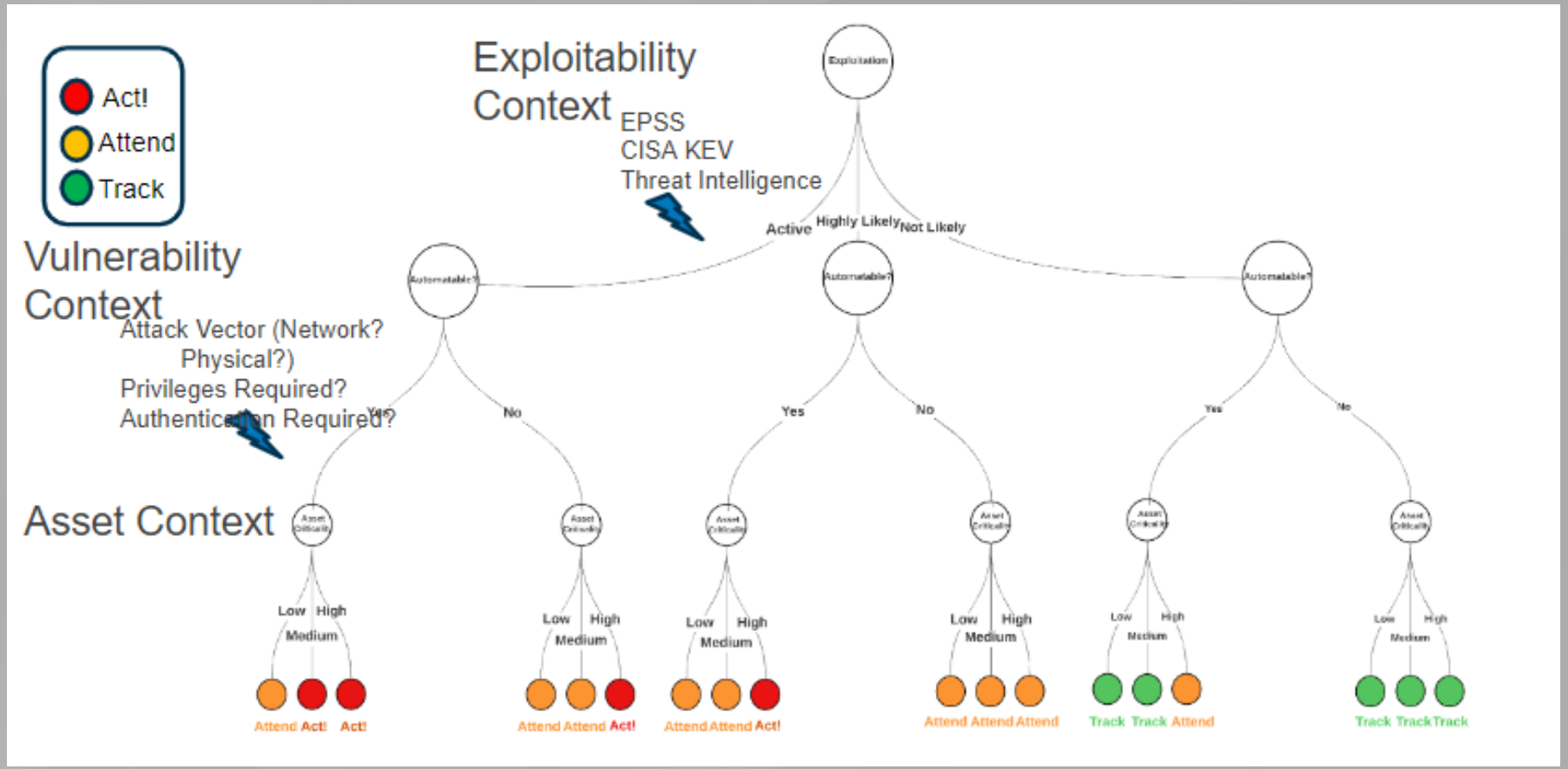
Example #2 Deployer



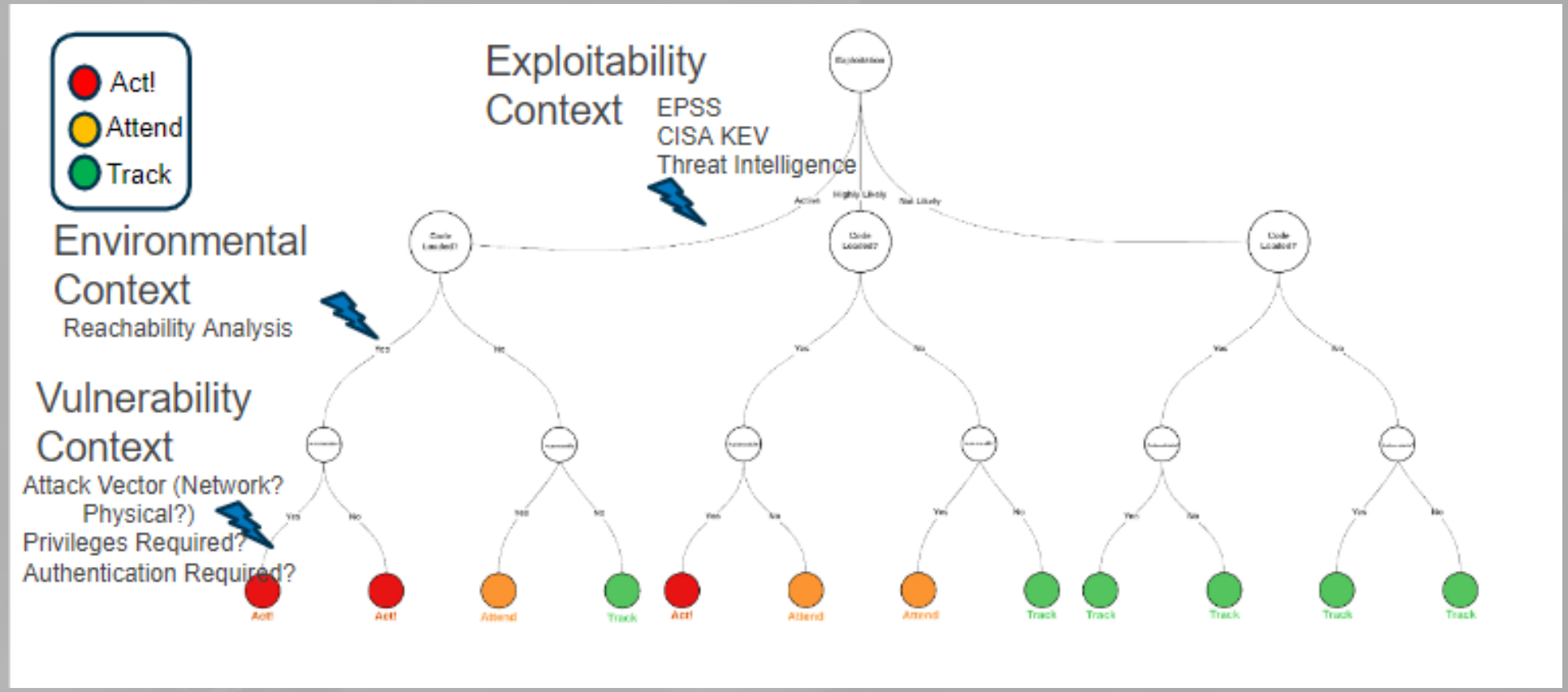
Example #2 Deployer



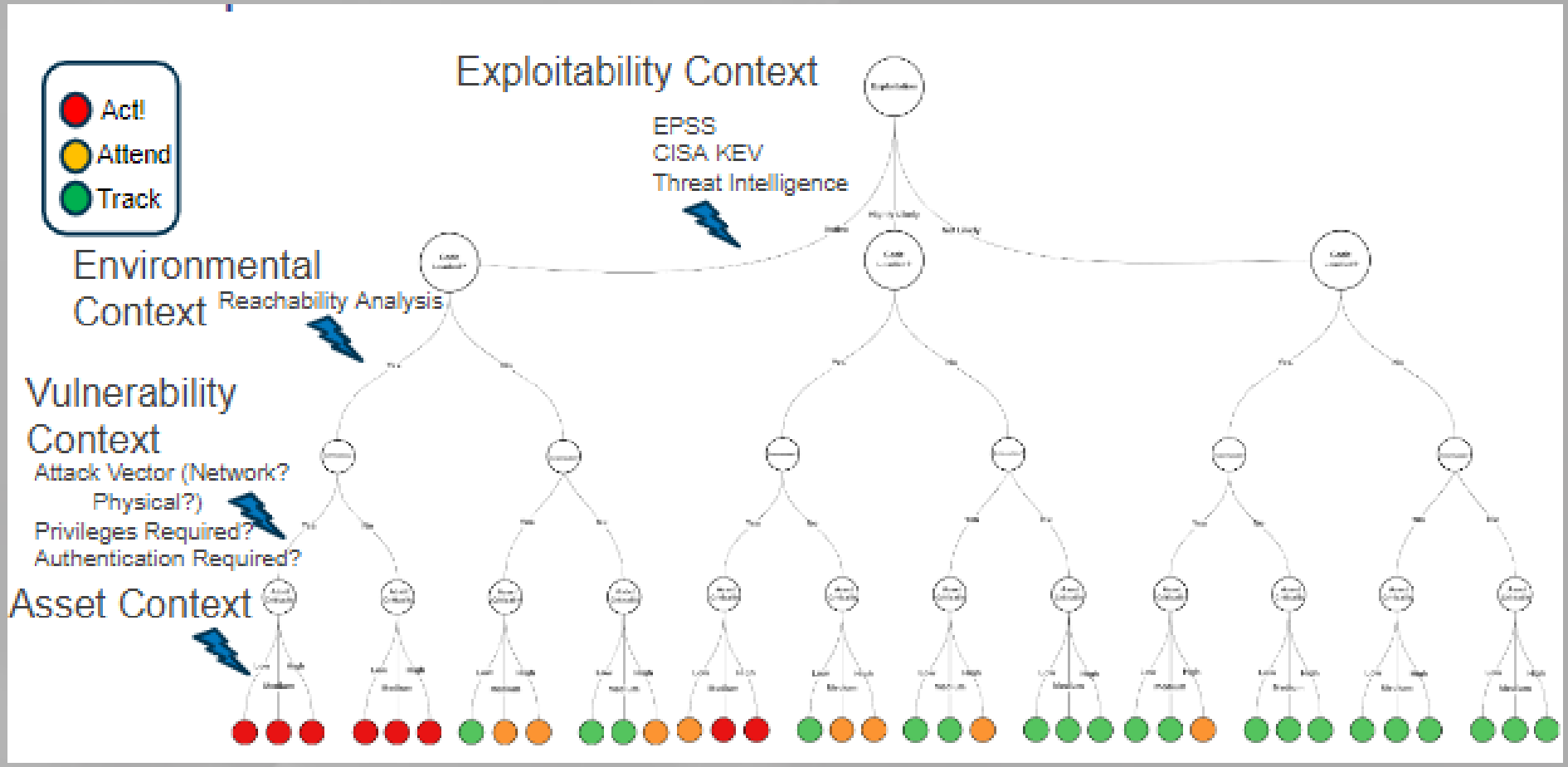
Example #3 - Custom SSVC



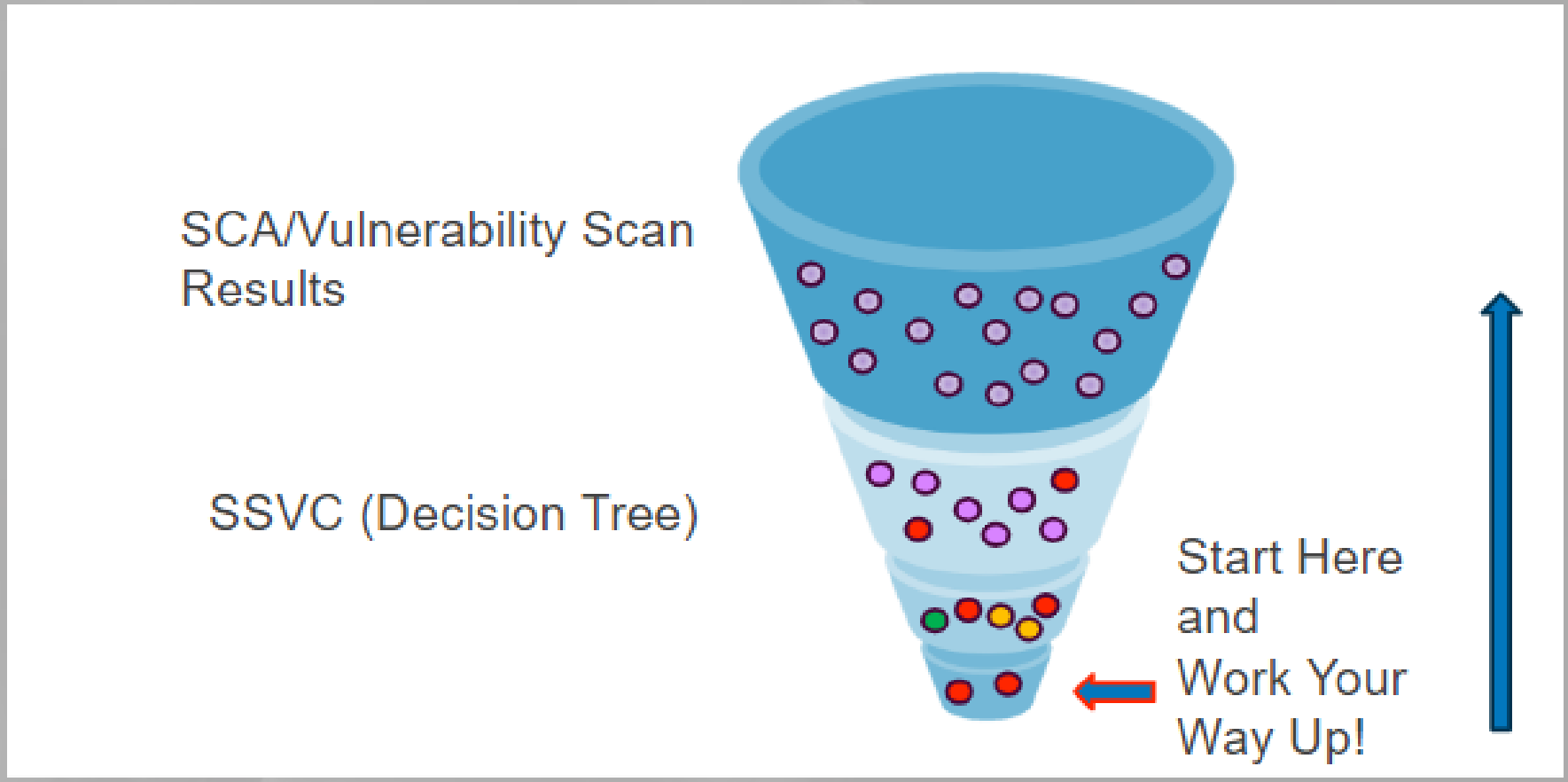
Example #4 - Custom SSVC



Example #5 - Custom SSVC



Priority Funnel

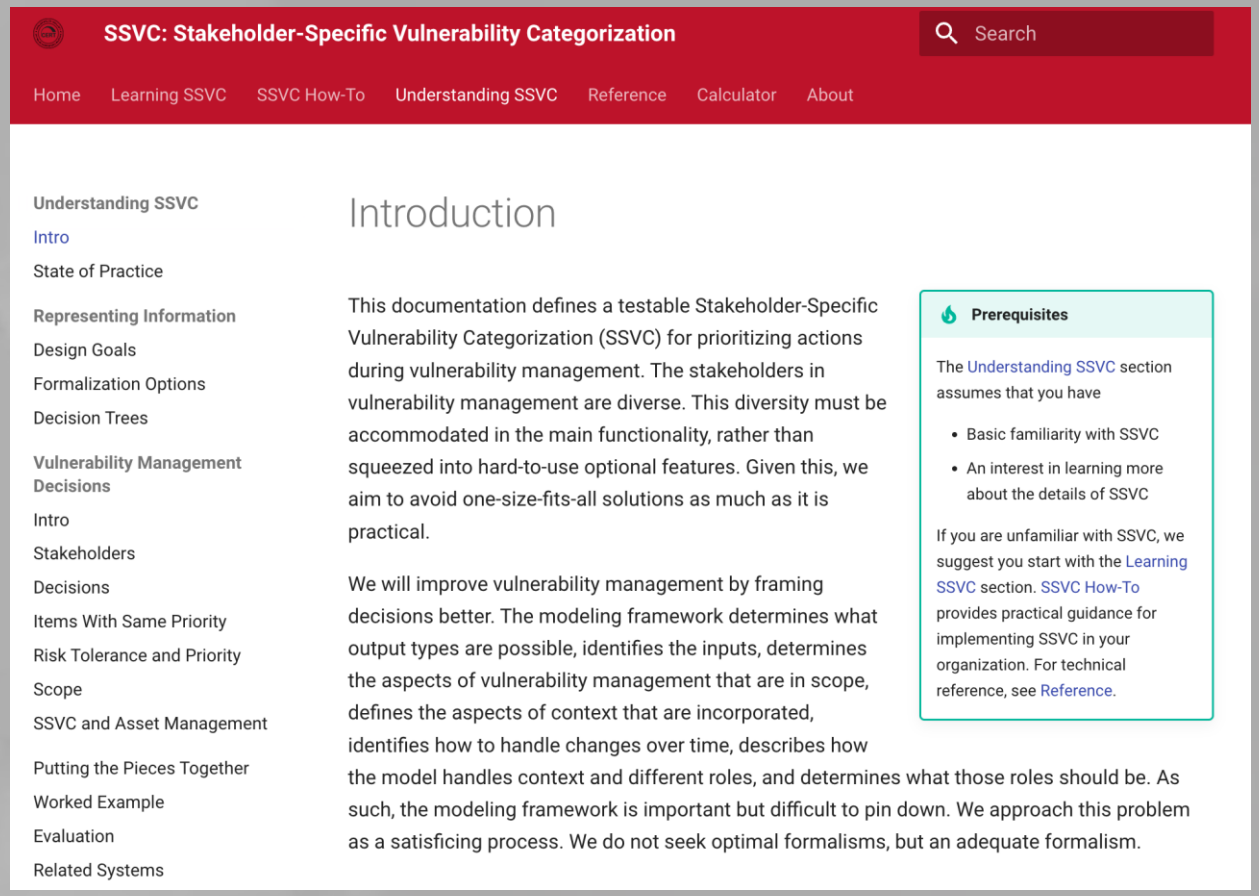


Learning More

A great new resource:
certcc.github.io/SSVC/

Provides components for building
your own decision models,
and some examples to help you get you started.

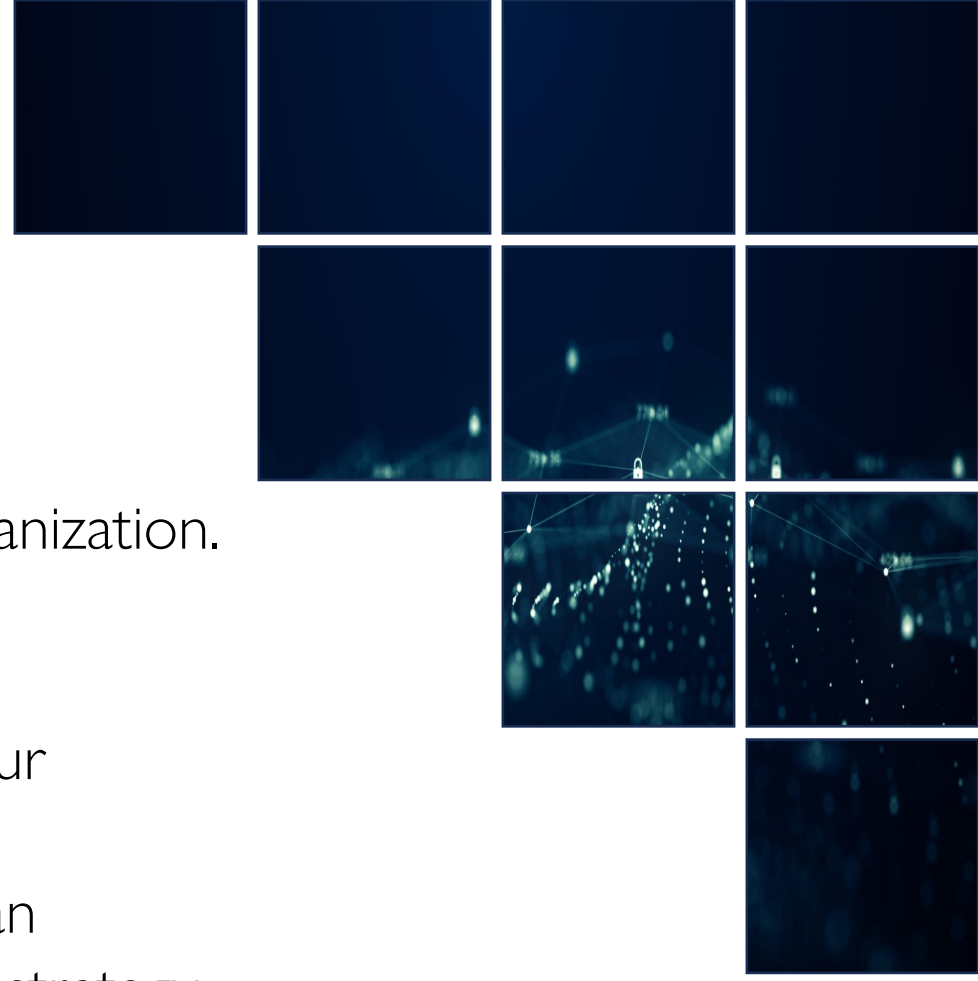
github.com/theparanoids/PrioritizedRiskRemediation



Is it Automatable?

Final Thoughts

- Start by thinking about how you define risk in your organization. This process alone is valuable!
- Avoid "Analysis Paralysis" Start simple and go for there.
- Even a simple decision tree can significantly improve your efficiency
- People often expect a single magic number that they can prioritize by, but an effective Vulnerability Management strategy has to integrate different sources of context in the decision and prioritization process.
- As we saw, each framework focuses on a specific aspect of risk. It isn't going to magically account the other aspects.




Questions?
Thank you