HOMELAND SECURITY
SYSTEMS ENGINEERING & DEVELOPMENT INSTITUTE™

Prepared for:
Department of Homeland Security

Suggestions for Simplifying the Common Vulnerability and Exposures (CVE) Counting Method

March 22, 2016

Version 1.0

Abstract

Common Vulnerability and Exposures (CVE) is a dictionary of publicly known information security vulnerabilities and exposures. An important part of CVE operations is vulnerability counting. Counting is a complex process due to the increasing complexity in software design and the significant growth in software vulnerabilities disclosed each year. In this paper, simplifications to the CVE counting process are proposed, with the goal of increasing the rate of issuance of CVE identification numbers to more closely match the increasing rate of disclosed software vulnerabilities. Pros and cons for each suggested change are included.

Record of Changes

No.	Date	Reference	A=Add M=Modify D=Delete	Description of Change
1	3/21/2016		A	Material consolidated into single document

HSSEDI

Table of Contents

This page intentionally left blank

1 Purpose

This paper proposes changes to simplify the Common Vulnerability and Exposures (CVE) counting process. Pros and cons for each suggested change are included, specifically addressing what impact the changes will have on timeliness and quality of the CVE identification number (CVE ID) assignment process and the resulting CVEs.

The nature and accuracy of the counting process underpins the value of a CVE. Correct counting reduces the likelihood of duplicate CVE IDs being assigned to a single vulnerability. Also, some reports of vulnerabilities may confuse or conflate multiple, separate software problems, and the counting process differentiates between unique vulnerabilities.

2 Current CVE Counting Process

This section describes the definitions, inclusion rules, and processes that are currently used for CVE counting decisions.

2.1 Definitions

The CVE program is valuable to stakeholders because it uniquely identifies vulnerabilities, thereby establishing a dictionary of publicly known information security vulnerabilities and exposures. Counting should be performed consistently across analysts and CVE Numbering Authorities (CNAs). [1] Without consistency, the quality and utility of CVE will be reduced due to misunderstandings regarding what is considered a vulnerability and why, which defeats the purpose of the CVE program.

Vulnerability counting is the process by which CVE decides whether a reported vulnerability represents a single vulnerability, multiple vulnerabilities, or is even a vulnerability at all. The counting process is used to assign the correct number of CVE IDs to a disclosed vulnerability.

A **vulnerability** is a flaw or design oversight in software that could be used by an attacker to gain unintended access to a system or network. CVE considers a flaw a vulnerability if it allows an attacker to use it to violate a reasonable security policy for that system (this excludes entirely "open" security policies in which all users are trusted, or where there is no consideration of risk to the system).

The flaw or design oversight leading to a potential vulnerability is often referred to as a **bug**.

A **configuration issue** is where a purposeful customization in the behavior of software results in an unintended state.

A **codebase** is a software component that is shared among products.

A software **product** is a collection of installable software distributed under a unique name by a particular vendor or development project.

An **executable** file causes a computer to perform indicated tasks according to encoded instructions, as opposed to a data file that must be parsed by a program to be meaningful.

A **software version** is a unique name for a particular revision of computer software. This includes commit IDs and other versioning identifiers. Within the CVE process, the specific version or versions affected by a vulnerability are key factors in the counting process. A **software package** is a collection of separate, self-contained software components that are distributed as a single, monolithic object.

A vulnerability is **publicly known** when the issue has been published or divulged publicly (or is scheduled to be published by a researcher or vendor who has been in communication with the CVE Team regarding the issue).

A product is **publicly available** when anyone can purchase or obtain legitimate access to it. This includes freeware, shareware, open source, and commercial products.

The **U.S. Information Technology (IT) Sector** is defined by a set of functions performed by the entities that comprise the sector. Those functions provide: a) IT products and services; b) incident management capabilities; c) domain name resolution services; d) identity management and associated trust support services; e) Internet-based content, information, and communications services; and f) Internet routing, access, and connection services. [2]

An **access vector** or **extent** describes how a vulnerability may be exploited. Examples of this include a local exploit (by having a presence on the system), a physical exploit (by having physical access to the system), or a network-based exploit (where the vulnerability can be exploited through a network connection).

A **vulnerability type** is defined by a combination of attack model (e.g., symlink attack) and the type of mistake that causes the vulnerability (e.g., the product does not properly check permissions).

2.2 Inclusion

Inclusion is the process of determining if a bug should be considered a vulnerability, and thus a candidate for a CVE ID. One criterion for CVE ID assignment is if the exploitation of the bug provides an attacker with extra privileges, additional information, or the ability to cause a software fault previously unavailable before they attempted to exploit the bug. This is the first consideration during inclusion; if a bug satisfies this criterion, it is tested against the following requirements:

- **Publicly Known** - The bug is published or divulged publicly (or is scheduled to be published).
- **Publicly Available, Licensed Software** - The bug must affect software that is licensed and made generally available to the public. If the bug only affects a version of software that was never made generally available to the publisher’s or vendor's customers, the bug should not be assigned a CVE ID. CVE IDs are not assigned to bugs in malware.
- **Installable Software and Libraries** - The bug must be present in an installable, distributed software product or library and not be a site-specific issue. In this case, the meaning of site-specific is that the bug only affects an online service (software-as-a-service), a specific

website, or only offered through hosting solutions that are under the full control of the vendor.

- **Utilized by U.S. IT Sector** – The bug must affect software that can be reasonably assumed to be used by or will directly affect the U.S. IT Sector, as defined by the Products and Sources list. [3] For example, U.S. IT Sector software includes English-language versions of software items that enable commerce and are used by the U.S. Government and industry. Security tools or software used for security awareness within the U.S. IT Sector are also included.

2.2.1 Vulnerability Investigation

When researching a reported vulnerability, a balance must be struck between performing CVE ID assignment in a timely manner, and delaying CVE ID assignment to allow for additional information gathering to increase the confidence in an assignment decision. Often, vulnerability disclosers include few details about the vulnerability they identified. If those details are not sufficient for making a determination, additional research may be required. CNAs should set limits as to how deeply a vulnerability will be researched. The CVE Team uses a five minute rule of thumb to limit the investigation time for each piece of missing information. The CNA role is not to discover vulnerabilities; CNAs exist only to distribute CVE IDs to newly-discovered vulnerabilities. An organization acting as a CNA may do vulnerability research as part of its normal business, but it is not part of the CNA role.

At the minimum, analysts need enough information to determine if something is a vulnerability, if it amounts to more than one vulnerability, and if it affects a product that is in scope for the CNA in which they work. Also, there must be enough information to determine if a request for a new CVE ID is for an existing vulnerability already assigned a CVE ID (i.e., a duplicate) or part of an existing vulnerability that has a CVE ID.

At a minimum, analysts must be able to answer the following questions:

- Is what is reported a vulnerability?
- What are the affected products? Are the products within CVE's scope or listed on the CVE Product and Sources lists? [3]
- Is there more than one vulnerability? If so, are they different types of vulnerabilities? Can each vulnerability be fixed independently, or would a single fix affect all the disclosed issues?
- Does the vulnerability already have a CVE ID assigned to it? (Is the vulnerability disclosure a duplicate?)

The CVE Team may conduct additional research to find and document the following details if they are available:

- Affected software versions
- Specific vulnerability types

- Extent/Access vector
- Impact
- Attack vector
- Affected components
- Authentication/privilege requirements
- Acknowledgement from vendor
- Initial disclosure (link to public documentation of disclosure, e.g., mailing list archive)
- Environmental conditions that enable the vulnerability

The CVE Team does not research beyond this set of attributes. This work is done to increase the confidence that a vulnerability is unique and not duplicated in earlier CVE ID assignments.

Other CNAs may choose to perform up to the CVE Team's level of research (or more) based on the resources they have available and the level of comfort they strive for when performing counting. That said, CNAs are only expected to document the information required to answer the minimum set of questions, as listed above.

2.3 Vulnerability Categorization and Counting

The fidelity of CVE increases as the confidence about the nature of disclosed vulnerabilities increases. It is important to determine whether a disclosure describes a bug that involves a single vulnerability or multiple vulnerabilities. This process is known as counting.

Correctly counting vulnerabilities is important. Correct counting reduces the likelihood of duplicate CVE IDs being assigned for a single vulnerability. Also, some reports of vulnerabilities may confuse or conflate multiple, separate software problems, and the counting process differentiates between unique vulnerabilities.

2.3.1 Categorizing Vulnerabilities for Counting

Specific vulnerability type information is useful when determining the correct number of vulnerabilities to assign to a reported issue, but it is not required to perform counting. Vulnerability categorization is not intended to be an official statement about the vulnerability.

The important aspect of vulnerability categorization is understanding when individual vulnerabilities may be rooted in different software errors. It is more important to know that two different vulnerabilities are different types, than it is to know specifically what types they are. The more information that is known, the more confident an analyst would be about their counting, but some uncertainty in this process is acceptable.

2.3.2 Counting Vulnerabilities

The current method for deciding the number of CVE IDs to assign for a reported vulnerability is:

1. Create separate CVE IDs for separate types of bugs. For example, a buffer overflow vulnerability should have a different CVE ID than a directory traversal problem.
2. If the same vulnerability of a single type is present in a range of versions for the same software product, then the vulnerability is assigned only a single CVE ID.
3. If two vulnerabilities have the same vulnerability type and are both present in the same version of software, create a single CVE ID for the two vulnerabilities referencing that version of software.
4. If two vulnerabilities in separate software products or versions can be linked to a common codebase that is shared by those products or versions, a single CVE ID is created for these vulnerabilities. (The root issue is a single vulnerability, even if it manifests in multiple software packages or versions.)
5. If none of the above conditions can be determined, create separate CVE IDs for each reported vulnerability.

If there is a possibility that there are multiple similar bugs as seen through (2), then analysts must clearly distinguish between them in the CVE ID description. Some rationale used for counting decisions should be stated in the CVE ID description to allow consumers of CVE to understand the general nature of the vulnerability and how it relates to other issues with the software, versions, or codebases.

2.4 Other Considerations

When multiple disclosers independently find what appears to be the same vulnerability, the reported vulnerabilities receive the same CVE ID if it is clear they are the same vulnerability. If it is not clear, or if it is clear they discovered different vulnerabilities, the vulnerabilities are each given a different CVE ID.

If a single discloser makes multiple vulnerability disclosures within 48 hours, the disclosures are treated as a single disclosure. The number of CVE IDs to assign is determined based on the established counting rules. This is done to avoid the unnecessary creation of CVE IDs for individual disclosers. For example, if a single discloser publishes five vulnerabilities over a 24 hour period that affect the same version of the same product and share a vulnerability type, the vulnerabilities will be assigned a single CVE ID, even though they were disclosed in separate reports.

3 CVE Counting Simplification

This section proposes a set of changes to the current counting process. Each proposed change affects a different part of the counting process. They can be implemented individually, as a whole, or in any other combination. The goal of the changes is to make the counting process easier to understand and reduce the amount of processing time to create a CVE.

3.1 Change 1: Vulnerability Definition

CVE IDs should only be assigned to vulnerabilities, which means that CVE must define what a vulnerability is, and analysts must ensure that each bug satisfies the definition for vulnerability. The current definition requires knowledge of the software's security policy.

3.1.1 Current Definition

A vulnerability is a mistake or design oversight in software that could be used by an attacker to gain unintended access to a system or network. CVE considers a mistake a vulnerability if it allows an attacker to use the mistake to violate a reasonable security policy for that system. (This excludes entirely "open" security policies in which all users are trusted or where there is no consideration of risk to the system.)

3.1.2 Current Process

A disclosed issue is considered a vulnerability if:

- There is a mistake or design oversight in software reported.
- The mistake or oversight violates the security policy of the system.

3.1.3 Problem

Understanding what the security policy and intended access rules are for a system requires a great deal of knowledge about a product. This information may be undiscoverable without the vendor's cooperation, and, even with the vendor's cooperation, getting that information will, on average, add several days to processing time of a CVE ID request.

In addition, behavior that product users feel is a risk may not be considered a risk by the software vendor. Product users may believe the vulnerability exists and the vendor may insist that it does not, which confuses and delays CVE ID assignment.

3.1.4 Proposed Change

CNAs can use either the security model definition CVE currently uses, or a claim-based definition. A claim-based definition is one where a discloser's claim of a vulnerability with a negative impact is sufficient. The claim-based definition removes the need to understand the product's security model and removes the vendor from the process. Section 3.1.6 describes the implications of the claim-based model.

3.1.5 Proposed Process

CNAs (including MITRE) can choose to use either the security model definition or the claim-based definition of a vulnerability.

Security model definition:

- There is a mistake or design oversight in software reported.

- The mistake violates the security policy of the system.

Claim-based definition:

- A mistake or design oversight in software is reported.
- Someone claims there is a vulnerability, and either:
  - The vendor agrees; or
  - There is a demonstrated negative impact.

The MITRE CVE Team will operate using the claim-based definition if this model effectively serves the needs of CVE stakeholders.

3.1.6 Expected Impact

Using the claim-based definition has tradeoffs. Currently, approximately 15 percent of all MITRE-assigned CVE IDs require investigation into the product's security model. This investigation is not performed when using the claim-based definition, significantly decreasing the time needed to perform assignments. However, a product's security model is often a balance between usability and security. There will be instances where someone will claim there is a vulnerability in a product, but the vendor and the users of the product have already accepted the risk. For example, many people know that strcpy is a potentially dangerous function and is used incorrectly in practice, but the security model is such that the user of strcpy is expected to validate the input before using the function.

Some of the impacts of assigning CVE IDs to vulnerabilities that do not violate the product's security model are:

- Security software developers, if they are not subject to specific regulatory requirements, could waste resources on designing or implementing remediation processes for product behaviors that are either required for product functionality, or are intentional risk-management choices.
- Security software owners, if they are subject to specific regulatory requirements such as the Payment Card Industry Data Security Standard, could face direct economic losses, either by choosing to remove properly functioning software from their systems, or by choosing to continue using software despite its association with a spurious CVE ID.
- Vendors could face an increase in support costs along with an increase in the number of CVE IDs assigned because, during the hours or days following publication of a CVE ID, many customers would initiate contact to demand a patch or request information about the CVE ID.
- Maintainers of security tools and vulnerability databases would propagate the disputable CVE ID's information to their own customers. If these new CVE entries were erroneously included in the CVE list because of the claim-based definition, this could result in a substantial increase in the amount of time needed for a spurious CVE ID's negative effects to be resolved after the issue is better understood (e.g., after the CVE ID is disputed).

- CVE IDs assigned to products without regard for their security models can be expected to result in an increase in disputed CVE IDs, causing confusion as to whether the CVE IDs cover legitimate vulnerabilities.

3.2 Change 2: The Affected Product Must be Installable Software or an Installable Library

CVE IDs are assigned to vulnerabilities to ease communication about them. If little to no communication across multiple stakeholders is required to solve the vulnerability, then a CVE ID is not required. For example, a vendor needs only to communicate internally to take an action to mitigate a vulnerability specific to the vendor's website.

3.2.1 Current Definition

The reported vulnerability must be present in an installable, distributed software product or library, and not be a site-specific issue. The software in question must be customer-controlled. In this case, the meaning of site-specific only affects an online service (e.g., a Software-as-a-Service product), a specific website, or a product only offered through hosting solutions that are under the full control of the vendor. In these situations, there is no mitigating action, such as patching, for the end user of the software to take.

3.2.2 Current Process

- If a user can take action to mitigate the vulnerability, then it should get a CVE ID.
- If it is unclear if the user can mitigate the vulnerability, processing of the vulnerability is delayed until clarity is obtained.
- If only the vendor of the product can take an action to fix the vulnerability, a CVE ID should not be assigned.

3.2.3 Problem

It has become harder to identify whether a product is vendor maintained or customer maintained. Many vendors provide cloud-based services, and it is not always clear if they offer a locally-managed product. Even when there is locally installable software, it is becoming more common that the software incorporates vendor controlled services.

3.2.4 Proposed Change

CVE IDs will be assigned when there is uncertainty about whether the product is customer controlled. This will reduce the time needed to process CVE ID requests for vulnerabilities in products where it is not clear if the customer controls or maintains the software.

3.2.5 Proposed Process

- If a user can take action to mitigate the vulnerability, a CVE ID should be assigned as it is done now.

- If it is unclear if the user can mitigate the vulnerability, a CVE ID should be assigned.
- If only the vendor of the product can take an action to fix the vulnerability, a CVE ID should not be assigned, as is currently practiced.

3.2.6 Expected Impact

- The CVE Team estimates that 20 vulnerabilities will no longer be indefinitely delayed until further evidence is found, which is rarely ever forthcoming.
- In the short term, the benefit of this change will not be significant because of the use of the Products list. By having a defined list of products, the probability is low that a CNA (e.g., the CVE Team) will not know if a product is customer controlled. Since vendors are constantly developing new products and changing their existing products, there will be an increase in the overall time savings related to this change.
- There will be approximately a five percent increase in the number of CVE IDs that are rejected because the software is later found to be site-specific.
- CVE IDs are more likely to be assigned to vulnerabilities that the CVE stakeholders would not typically consider relevant, such as the inadvertent assignment of a CVE ID to a Software-as-a-Service product. This may result in an inappropriate expenditure of resources by those who rely on CVE to identify vulnerabilities that require action. For a more thorough explanation of what the expenditures might be, see the bulleted list in Section 3.1.6.

3.3 Change 3: Using Affected Product to Decide How Many CVE IDs to Assign

Once the vulnerabilities have been verified as meeting the CVE inclusion requirements, the number of CVE IDs that should be assigned must be determined. One method used to determine the number of IDs to assign is to assign a CVE ID to each affected product. This criterion meets stakeholders' expectations because the nature of vulnerabilities and their solutions tend to vary based on the affected product.

3.3.1 Current Definition

CVE ID requests are split into multiple CVE IDs based on the affected product or products. However, some products share the specific code containing the reported vulnerability. In these cases, a CVE ID is assigned instead to the shared codebase used by the affected products.

3.3.2 Current Process

When considering how many CVE IDs to assign based only on the affected product or products, the CVE Team uses the following criteria:

- If the vulnerabilities affect a single product, assign one CVE ID.

- If the vulnerabilities share the same affected codebase, such as a library, executable, or third-party software package that is used by many vendors, then assign a CVE ID to each affected codebase.
- If the vulnerabilities result from conforming to a design or protocol that has a problem, assign a single CVE ID.
- If the vulnerabilities are caused by an implementation problem where many products accidentally make the same mistake, assign a CVE ID to each affected codebase.

3.3.3 Problem

Codebase relationships are often unclear, difficult to investigate, and require analytical judgment (e.g., how much does a software product have to change to be considered a different codebase?).

Here, too, vendor security models come into play. Referring to the strcpy example, a CNA must know the expectation for the user is to use strcpy safely to choose the fourth decision in the current process instead of the third. In many such cases, there are stakeholders advocating for one or the other decision, and the CNA is put in a position where they must adjudicate the disagreement.

3.3.4 Proposed Change

The CNA may choose to accept the affected product cited in the initial claim. Also, CVE will add a decision option to the process to assign CVE IDs when the relationships are unknown or uncertain.

3.3.5 Proposed Process

- If the initial claim says the vulnerabilities affect a single product, assign one CVE ID.
- If the initial claim says the vulnerabilities share the same affected codebase, assign a CVE ID to each affected codebase.
- If the codebase relationship is unknown or unclear, assign a CVE ID to each affected product.
- If the initial claim says the vulnerabilities result from conforming to a design or protocol that has a problem, assign a single CVE ID.
- If the initial claim says the vulnerabilities are caused by an implementation problem where many products accidentally make the same mistake, assign a CVE ID to each affected codebase.
- If it is unclear whether the problem results from a design or protocol, assign a CVE ID to each affected codebase.

3.3.6 Expected Impact

- Processing time for new CVE IDs will be reduced by up to two percent and CVE ID assignment will no longer be delayed by research into code relationship concerns.
- The CVE Team estimates that approximately six codebase relationships will be missed each year because of the lack of investigation. Stakeholders using products with the affected code, but not the product identified in the CVE entry, will be unaware they are vulnerable. Also, the number of disclosed vulnerabilities whose codebase relationships are not investigated may increase due to CVE no longer requiring it. The risk associated with this however, is low.
- Multiple CVE IDs for the same underlying vulnerability will be assigned when the discloser does not investigate or specify the code relationships. The multiple IDs will result in the following:
  - Misallocation of resources by CVE users for handling vulnerabilities with which they have already dealt.
  - Additional expenditure of resources to update products when the CVE IDs are merged and rejected.

3.4 Change 4: Using Independently Fixable Vulnerabilities in Place of Vulnerability Type to Decide How Many CVE IDs toAassign

Another method currently used to determine the number of CVE IDs to assign is vulnerability type.

3.4.1 Current Definition

Vulnerability disclosures should be grouped by vulnerability type, and each group should be given a CVE ID. Vulnerability types are defined by a combination of attack model (e.g., cross-site scripting) and the type of mistake that causes the vulnerability (e.g., the product does not properly check permissions).

3.4.2 Current Process

Create separate CVE IDs for separate types of vulnerabilities. For example, a buffer overflow vulnerability should have a different CVE ID than for a directory traversal vulnerability.

3.4.3 Problem

There are multiple ways to define vulnerability type, such as cause, impact, or attack model. The security community has not agreed on a single way to define vulnerability types. In addition, the method CVE currently uses to categorize vulnerability type requires extensive research into, and understanding of, the precise nature of the problem. This requirement makes identifying the

vulnerability type difficult, and many stakeholders are not willing to invest the necessary time and effort into making this determination.

The concept of "vulnerability type" was useful when the counting practices were created and little was known about secure software development. For example, at that time, it seemed natural and useful to combine many buffer overflows into one CVE ID, because the developer had a single conceptual failure of not considering the possibility of long and invalid input. Today, a buffer overflow is usually understood in different ways. Buffer overflows can be understood in a complex way relating to a theory of vulnerability chains or in a simple way as the ultimate impact of an unrelated coding error.

3.4.4 Proposed Change

Replace the notion of vulnerability types with "distinct finding" or "independently fixable."

3.4.5 New Process

- If a vulnerability can be fixed independently of the others, assign a CVE ID to it.
- If the vulnerabilities cannot be fixed independently, assign a single CVE ID to the group of vulnerabilities.
- If it is not clear whether the vulnerabilities can be fixed independently, assign a single CVE ID to the group of vulnerabilities.

3.4.6 Expected Impact

- Processing time to assign CVE IDs will be reduced by approximately 10 percent. CVE ID assignment will no longer require investigating the exact type of vulnerability.
- CVE ID requesters will have a better understanding of what is expected of them, as the new process better conforms to the secure software development practices of today.
- There may be less specific technical information included in CVE ID entries. CVE consumers would need to seek that information from other sources, which may affect the utility of CVE for those who may have looked for CVE to provide that kind of information, such as system or network administrators who begin their research of a CVE entry with the CVE list.

Acronyms

Acronym	Definition
CNA	CVE Numbering Authorities
CVE	Common Vulnerabilities and Exposures
CVE ID	CVE Identification Number
HSSEDI	Homeland Security and Systems Engineering Development Institute
IT	Information Technology

References

[1] http://cve.mitre.org/cve/cna.html

[2] Information Technology Sector-Specific Plan 2010 http://www.dhs.gov/sites/default/files/publications/IT%20Sector%20Specific%20Plan%202010.pdf

[3] http://cve.mitre.org/cve/data_sources_product_coverage.html