The CVSSv2 Shortcomings, Faults, and Failures Formulation

Subject: An Open Letter to FIRST
From: Carsten Eiram, Risk Based Security and Brian Martin, Open Security Foundation

This is an open letter to FIRST regarding the upcoming Common Vulnerability Scoring System (CVSS) version 3 proposal. While we have not been formally asked to provide input, given our time spent in the world of vulnerability databases and the security industry, along with our extensive use of CVSS in scoring vulnerabilities, we feel we may provide valuable insight and hopefully help to address the shortcomings, faults, and failures of CVSSv2.

The following will solely discuss base metrics, but should FIRST wish it, we would be happy to engage in dialogue with them about temporal metrics as well.

While CVSSv2 saw improvements over CVSSv1, the scheme is still not adequately supporting real life usage, as it suffers from being too theoretical in certain aspects. Specific vulnerability types and vectors are not properly supported while others are not properly described, leading to subjective and inconsistent scoring, which CVSS was designed to prevent.

While many companies may request CVSS compliance, most don't really use it due to the shortcomings, or they are not getting the desired value from it. Organizations and companies tasked with doing CVSS scoring are either forced to use a flawed scheme or make their own internal tweaks and changes to the scheme in order to make it somewhat work. Some companies go so far as to create their own vulnerability scoring systems, similar to CVSS.

As a few examples of public entities breaking from CVSS, regardless of intention:

- NVD is often inconsistent in their scoring and provides incorrect scores frequently. We have sent over a dozen corrections to NVD that were accepted.

- OSVDB sets CVSSv2 scores immediately when publishing entries. These scores attempt to follow the CVSSv2 standard, but due to the ambiguity certain definitions and scoring tips, they are slightly tweaked. Breaking from a strict adherence to the CVSS rules can allow OSVDB's scores to be more accurate, consistent, and reliable than the scores from some other parties.

- Secunia provides two versions of CVSSv2 to their customers: The NVD score when it is available, and one based on their own custom definitions which attempts to make scores more accurate and in keeping with their own 5 level scoring system. As Secunia bundles multiple vulnerabilities into a single entry, their internal CVSSv2 scores only reflect the most severe of the vulnerabilities covered by an advisory, meaning multiple vulnerabilities of varying impact may only receive one score.

- Oracle does not seem to consistently follow CVSS scoring rules and has, furthermore, created a whole new scoring level referred to as "Partial+", which adds more granularity, but makes their scores non-compliant with CVSSv2.

- IBM, HP, Cisco, and other major vendors frequently release CVSSv2 scores in their advisories that do not appear to follow scoring guidelines. In some cases, issues are scored too high, as ultimately demonstrated when more details are released. In other cases, scoring is too low, as reflected when a vendor scores an issue 0.0, but subsequent details show considerably more risk. Some vendors also assign a single score to two distinct vulnerabilities, e.g. Cisco for cisco-sa-20130206-ata187, which goes against guidelines.

CVSSv3 must be better structured to account for granular scoring, and provide a system that can be easily implemented by a wide variety of companies. This must be suitable to companies outside the security business who are responding to vulnerabilities in their software. While no scoring system will ever be perfect, FIRST must strive to address the weaknesses in the current model that lead to the system failing to properly support users.

In the following pages, we highlight several weaknesses in the current scoring model, and offer ideas for improving the system in the future.

Respectfully submitted,

Carsten Eiram (Risk Based Security)
che@riskbasedsecurity.com

Brian Martin (Open Security Foundation)
brian@opensecurityfoundation.org

The 4th Level Granularity Consideration

The current 3-level scoring system simply does not add sufficient granularity to vulnerability scoring. While CVSS technically provides scoring between 0.0 to 10.0, it is based on a 3-level system. This leads to disparate vulnerabilities ultimately receiving the same score, because that score is derived from a limited number of variables.

The Cyberspace Five-O Problem

As an example, a path disclosure flaw in a web application would be scored as (AV:N/Au:N/AC:L/ C:P/I:N/A:N) for a total score of 5.0. A vulnerability that allows an attacker to traverse the file system and read any file accessible by the web server would be scored as (AV:N/Au:N/AC:L/C:P/I:N/A:N) for a total score of 5.0. These two flaws obviously pose a significantly different risk, yet by CVSSv2 standards are no different. This illustrates the problem with a single 'Partial' category that essentially ranges from "just a bit" to "almost everything but not quite". While outside the scope of CVSS, this has ultimately led to considerable headaches in other aspects of information security. A 5.0 scored path disclosure is enough to fail an organization for PCI DSS compliance.

The Plus-sized Scoring Problem

Instead of the current 3-level scoring system (None, Partial, Complete), we suggest a fourth score to add granularity. This could be as simple as the Oracle-invented 'Partial+'. The current definition of 'None' and 'Complete' would remain unchanged, but would make a more granular distinction between 'Partial' and 'Partial+':

> Confidentiality:
> Partial - Subset of application accessible data can be disclosed.
> Partial+ - All application accessible data can be disclosed.
>
> Integrity:
> Partial - Subset of application accessible data can be manipulated.
> Partial+ - All application accessible data can be manipulated.
>
> Availability:
> Partial - Subset of application functionality can be impaired or application can only be impaired for a shorter period of time.
> Partial+ - Whole application can be impaired persistently until e.g. a restart/reboot.

Adding more granularity would also help to better cover code execution vulnerabilities within sandboxes. A code execution vulnerability within a sandbox would be scored P/P/P, a code execution vulnerability outside a sandbox with application privileges would be P+/P+/P+, and a full system compromise would be C/C/C.

The Application vs. System Uncertainty

One of the big factors that leads to subjective scores is uncertainties about the software or configuration. Best practices dictate that exposed services should run with the least privileges possible, while still being able to perform. This leads to web servers and applications being run under their own privileges, rather than as root. In turn, some vulnerabilities are then more limited in impact. Consider a web application running with root privileges versus user privileges. An arbitrary file access directory traversal flaw would be either 'Complete' or 'Partial' confidentiality. For a web application that comes with no recommendation for running privileges, how do you score it? Per the CVSS guidelines:

> *SCORING TIP #3: Many applications, such as Web servers, can be run with different privileges, and scoring the impact involves making an assumption as to what privileges are used. Therefore, vulnerabilities should be scored according to the privileges most commonly used. This may not necessarily reflect security best practices, especially for client applications which are often run with root-level privileges. When uncertain as to which privileges are most common, scoring analysts should assume a default configuration.*

How are analysts to know what privileges are used most commonly or by default? While we may know what Apache or IIS installs as, there are thousands of other applications that run their own native web servers, or bundle common ones but execute them under a different privilege scheme. In our directory traversal example above, the difference in scoring between 'Partial' (5.0) and 'Complete' (7.8) is considerable, especially when converted into a 'stop light' system that puts one as 'Medium' risk and the other as 'High'.

This problem extends to context-dependent attacks as well. Consider the case of a PDF rendering vulnerability that leads to arbitrary code execution. The average desktop user or a server administrator that does not use a less-privileged account for browsing the Internet will result in a full system compromise, while others may only be affected in the limited context of their user account. Add in configuration specific options, sandboxing, and other features that affect privileges and access, and this continuously proves to be a challenging aspect of CVSS scoring.

The Access Complexity Segregation

The current abstraction for Access Complexity (AC) all but ensures that only 2 of the 3 choices get used with any frequency. The bar between Medium and High is simply set too high in the current implementation. To support more accurate and granular scoring, it would be advantageous to have a 4-tier system for AC.

Another concern is that some aspects of vulnerability scoring are not outlined at all, particularly in this category. Access Complexity is likely the most subjective and problematic area for CVSS scoring, which is designed to provide objective and uniform scores and should, therefore, be subject to very clear scoring tips and rules.

The Man-in-the-Middle Complexity

One type of attack that lacks clear scoring guidelines is the Man-in-the-Middle (MitM) attack. As an example, CVE-2012-5762 requires a MitM attack, and the victim must be using Internet Explorer (IE). Microsoft IE's market share is currently around 30% based on several sites that collect such data. Worse, the likelihood of a victim using IE will change over time, making scoring subjective based on the perceived chance of that browser being used. Even worse, the MITM component is not explicitly factored into a category. Some may consider it part of Access Complexity, while other analysts may consider it part of Location (e.g. 'Adjacent Network').

Consider that NVD scores MitM examples ranging from Low, to Medium, to High. If the same group of analysts cannot consistently score the same complexity, it highlights the weak guidance available.

The Context-Dependent Conundrum

As CVSSv2 does not specifically support context-dependent attacks, it leaves analysts with AC as the only option for tweaking these to ensure the resulting scores differ from direct attacks. The lack of clear guidelines causes entities to score these differently across the board. Some analysts score as "AC:M", while others stick with "AC:L" even though limited social engineering is required (i.e. tricking a user into visiting a web page or opening a malicious file).

Based on the CVSSv2 description for AC:M (*"The attack requires a small amount of social engineering that might occasionally fool cautious users"*), this score does seem most appropriate. The advantage of using AC:M is that the resulting score for a context-dependent attack is slightly lower than a true remote attack. The disadvantage is that for context-dependent attacks, CVSSv2 then only supports two levels of AC granularity (AC:M and AC:H) since AC:L would never be used.

One proposed solution is to provide very clear guidelines for not only how to rate context-dependent attacks, but also based on the level of user interaction required. Some attacks may just require a user to visit a website, while other attacks also require the user to take certain actions. To ensure a difference in scoring from true remote attacks, AC:M should be the lowest score possible for context-dependent attacks, but in order to provide sufficient granularity, AC should support a fourth score that splits what "Medium" covers today.

The second proposed solution is related to Access Vector (AV), discussed in the next section.

The Access Vector (AV) Limitation

Access Vector (AV) currently supports three different vectors: 'L' (Local), 'A' (Adjacent network), and 'N' (Network). This current three-component breakdown for AV is too limited in the context of modern networks and access options.

The "Let's Get Physical" Proposal

The current definition for attacks considered 'Local' is: "*A vulnerability exploitable with only local access requires the attacker to have either physical access to the vulnerable system or a local (shell) account.*"

As indicated in the definition, the CVSS metric does not properly differentiate between physical attacks and attacks requiring an account on a system, instead just covering both as 'AV:L' despite there being a notable difference as far as the access requirements. This is further compounded by the fact that the "Authentication" metric also fails to provide a proper distinction as discussed later.

To provide a proper distinction, we propose that CVSSv3 specifically includes a category for attacks requiring physical access e.g. AV:P.

The Context-Dependent Differentiator

Scoring tip #6 clarifies that context-dependent attacks (i.e. targeting client applications where some user interaction is required e.g. in the form of opening a file) should be scored as "AV:N". As a result, AV has no accommodation for context-dependent attacks. Instead, it possibly (as there are no clear scoring tips or definitions) factors it into complexity, which has its own problems as already outlined above.

We already discussed context-dependent attacks in term of AC and proposed a solution. As a second proposal for covering context-dependent attacks better, we suggest that instead of bundling context-dependent attacks as "Remote", they should be covered by a separate score e.g. "AV:CD" for "Access Vector: Context-Dependent".

One advantage would be that AC provides more granularity since it would be possible to use the lowest score and still have the resulting score differentiate context-dependent attacks from direct attacks.

Another advantage would be that it would be possible to build in more accurate scoring for context-dependent attacks into the CVSSv3 calculator - something the CVSSv2 calculator currently lacks. This would include support for context-dependent DoS issues and other attacks where the severity of a direct attack greatly differs from a context-dependent attack against client applications.

As an example, consider a case where it is possible to crash a browser when a user views a specially crafted web page. In such cases, the resulting score would be either 4.3 or 5.0 depending on whether AC:L or AC:M was used. While it is outside the scope of this letter to discuss whether or not such should be considered vulnerabilities or just stability bugs, it should be noted that some browser vendors completely disregard browser crashes as having a security impact while others flag them as "Low". In the security industry in general, some consider such crashes as a minor concern at best while others just a stability bug.

In either case, it is, however, important to note that the CVSSv2 scores do not even come close to matching the perception of the industry.

The Mobile and Wireless Range Restriction

A third and growing area that AV also fails to properly accommodate for are wireless or mobile issues. While these are certainly network, they may be severely limited in range. It should,

therefore, be considered to have an additional vector to cover wireless-based attacks and similar, which are becoming more pervasive.

The Authentication Bifurcation

The current model of factoring in authentication seems to be overly reliant on the mentality that there are only two statuses of being authenticated to an application: Once or twice. In reality, there are several cases where this model completely fails as a viable scoring mechanism.

The Attacker vs. Victim Exclusion

The CVSSv2 guideline currently describes the "Authentication" metric as:

> "*This metric measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability.*"

The key word is "*attacker*", which means that the "Authentication" metric only covers if an attacker should be authenticated to an application or service; not if a target is required to be authenticated, which exploitation of some C/D (Context-Dependent) vulnerabilities require.

Based on the current definition and using CSRF (Cross-Site Request Forgery) vulnerabilities as an example, the "Authentication" metric should not be "Au:S" as the attacker does not need to be authenticated, but "Au:N". However, the target would need to be authenticated for administrative-based attacks, which adds a bit more complexity to a vulnerability, but is currently not covered under current guidelines.

Similarly, a reflected XSS (Cross-Site Scripting) vulnerability requiring a target to be authenticated to an application is currently rated exactly the same as one where the target does not need to be authenticated. Again, the full range of CVSSv2 scores is not utilized unless using AC (Access Complexity) as some sort of adjuster, which leads to inconsistent and seemingly random scores between parties scoring the same vulnerability.

CVSSv3 should take into account whether a target would have to be authenticated by e.g. providing scorings like "Au:TS" (i.e. "Target Single") and "Au:TM" (i.e. "Target Multiple"). It is, however, considered very unlikely that "TM" would be used much. Therefore, adding a single new score to cover authenticated target users may be adequate i.e. "Au:T" for "Authentication: Target". Alternatively, very clear guidelines for how to set AC (Access Complexity) in such cases should be provided, but would also require adding more scoring levels to AC in order to provide proper granularity.

The User vs. Administrator Conflict

Another problem with the "Authentication" metric is that it does not factor in the permission level granted by authentication. There is a severity difference in vulnerabilities requiring normal user or guest permissions versus trusted user permissions like administrator access. The "Authentication" metric should take this into account when scoring. Having scores like "Au:SP" ("Authentication: Single-Privileged") and "Au:SU" ("Authentication: Single-Unprivileged") would

improve scoring and add more value than the current "Multiple" designation, which is increasingly removed from today's software deployments.

The Locality Certainty

Authentication requirements should be absolute. If an attacker or victim is required to authenticate, it should be factored into scoring. The CVSSv2 guidelines specify that "*authentication requirements are considered once the system has already been accessed*". This means that a local attack that requires access is scored as "Au:N", when in reality it takes authenticating as a user to carry out the attack. Under this guideline, it further highlights the lack of distinction between requiring physical versus shell access, as both would be scored "Au:N".

This could be enhanced by having a classification such as "Au:SY" to cover designated local attacks requiring a system account. Alternatively, this could be done using the current CVSS mechanisms, but changing the rule of scoring (e.g. AV:L + Au:S). Further, the scoring guideline should be very careful in the use of "system" as quoted above, as this word is murky in the context of the security vernacular. Some consider the system to mean the operating system, while more generically, the word could mean the system and any application or service it runs, even if it does not require authentication.

The Common Vulnerability Scoring Minutiae

The Vulnerability Scoring Interdependency Isolation

When scoring separate vulnerabilities, they should be scored completely independently of each other and not take into account any interaction. Per CVSSv2 scoring tip #1:

> "*Vulnerability scoring should not take into account any interaction with other vulnerabilities. That is, each vulnerability should be scored independently.*"

While this policy makes sense in order to understand the threat of a vulnerability by itself, it is important to be aware of the limitation it imposes. In real life, two or more seemingly innocuous vulnerabilities may be combined to form a critical issue. The Google Chrome Pwnium full compromises are excellent examples where a string of vulnerabilities are combined - many of them only providing very limited gain by themselves - to a full sandbox escape resulting in arbitrary code execution.

Today, many attacks do not utilize a single vulnerability to fully compromise a system, but instead a string of vulnerabilities. In the real world, there is such a thing as vulnerability dependency, and vulnerabilities can raise the severity of each other. Failing to recognize this in a scoring system imposes a significant limitation. As security mechanisms and vendor software becomes more mature, these attacks will be increasingly important.

For the sake of argument, consider two examples of exploitation that result in a full system compromise, but are dependent on low-risk issues. The first a remote attack against AIX, the second a multi-step attack against a web application:

showmount info disc – (AV:A/AC:L/AU:N/C:P/I:N/A:N) = 3.3 (LOW)
Unpriv mount to default exported FS – (AV:A/AC:L/AU:N/C:P/I:P/A:P) = 5.8 (MED)
Add SUID binary to FS = (part of 5.8 above)
Add .rhosts to home directory = (part of 5.8 above)
Login to victim account (intended functionality)
chsec local privilege escalation to root – (AV:L/AC:M/AU:S/C:C/I:C/A:C) = 6.6 (MED)

Path disclosure – (AV:N/AC:L/AU:N/C:P/I:N/A:N) = 5.0 (MED)
Gives temporary file location – (part of 5.0 above)
Upload image manipulation – (AV:N/AC:L/AU:S/C:N/I:P/A:N) = 4 (MED)
Execute uploaded code – (AV:N/AC:L/AU:S/C:C/I:C/A:C) = 9 (HIGH)

In the first example, it shows how a series of lower risk vulnerabilities can turn a local privilege escalation into a remote root compromise. In the second example, we have an attack that is only successful if knowledge of where files are uploaded is gained, making the lower risk path disclosure an invaluable step in the process of achieving remote code execution.

The Sandbox Escape Reality Deviation

One of the vulnerability types that is poorly supported by CVSSv2 due to the above-mentioned limitations is sandbox escapes in applications like Google Chrome and Adobe Reader.

The problem is that the scoring for a sandbox bypass does not adequately reflect how serious such issues are perceived, and leveraged, in real life. While memory corruption vulnerabilities may be a dime a dozen, sandbox bypasses are not, and we are seeing more and more focus on having a mature sandbox to limit exploitation. Any accomplished penetration tester asked if they would prefer to have five use-after-free vulnerabilities in WebKit or a single Chrome sandbox bypass would know which one to pick.

Similarly, it is becoming more important for companies to address a possible sandbox bypass in a given application rather than a memory corruption issue in the same application when no sandbox bypass exists. However, CVSSv2 scores do not reflect this. In the CVSS world, the memory corruption vulnerability is the biggest concern; in the real world, it has become the sandbox bypass.

One may way of addressing interdependent vulnerabilities like sandbox escapes would be to rate them based on their full potential, but provide a "Vulnerability Dependent" flag similar to what OSVDB has provided for many years.

The Null Score Threat

While rare, there are times when a vulnerability is scored as a 0.0 by current CVSS standards. These are often vulnerabilities or weaknesses that do pose some limited threat, even if just a minor concern. Regardless, if the issue is considered a vulnerability by many in the industry, scoring should reflect this by attributing it a real score.

One such example is arbitrary site redirect issues. While not critical vulnerabilities, these are a concern even though people may be left with a different impression when large organizations like Google fail to address these in their own websites - which in the case of Google is ironic as they at the same time warn about such issues and provide recommendations on how to avoid them. Take the following arbitrary site redirection:

http://[target]/squirrel.php?cn=hungry&goingheretogetnuts=http://[attacker]/

Per current CVSSv2 scoring rules, this would yield (Av:N/Ac:M/Au:N/C:N/I:N/P:N) because:

* There is no impact to the confidentiality of the target system.
* There is no impact to the integrity of the target system.
* There is no impact to the availability of the target system.

One could argue that the integrity of the system is partially impacted from a reputation standpoint. If a user navigates to a business' website, and is quietly redirected to a site hosting malware or pornography, the user may assume that the business is compromised. However, the system being used to redirect has not really been affected in any way, giving a score of 0.0.

The Unknown Impact Scoring Uncertainty

Another challenge is scoring vulnerabilities that do not come with details. CVSSv2 assumes that a certain level of detail is available when scoring, when in reality, this is often not the case. Based on NVD's scoring approach, the status quo is to "assume the worst" and score it based on the worst-case impact. If there are absolutely no actionable details, this results in many issues getting a score of 10.0, even if in reality it is significantly less. While this should encourage vendors to provide minimal details, or their own CVSS score, that certainly does not happen most of the time. Given the number of such vague disclosures, it skews the count to have a high number of 10.0 scores. This makes it more difficult for administrators to accurately prioritize mitigation efforts.

The CVSS guidelines should clearly document how to deal with vague entries; either not scoring them unless a certain minimum level of detail is available, allowing scoring based on the analysts' best judgment (but likely resulting in different scores from different entities), or go with a worst-case impact.

The Immediate vs. Follow-up Impact Polarization

CVSSv2 does not properly deal with immediate impacts vs. follow-up impacts - neither to the target system itself or a user's system - and this often leads to confusion with regards to scoring a vulnerability where the immediate impact may be minor, but the follow-up impact more severe.

The Dialog Obstacle

As an example of the confusion on how to rate issues with clearly different severities and the lack of granularity, consider the following example.

OSVDB 63667: Adobe Reader / Acrobat Crafted PDF File Open Launch Sequence Arbitrary Program Execution Weakness (CVE-2010-1240)

The core problem with this vulnerability is that it allows partially spoofing a displayed 'File Launch' warning dialog. This in turn may lead a user into believing that it is safe to accept, which would result in arbitrary code execution on the user's system.

In this case, NVD scored this as a complete compromise with AC:M, meaning the rating was based on the follow-up impact **if** a user was successfully duped and not the immediate partial manipulation of the 'File Launch' warning dialog. However, the exact same rating was assigned to Foxit Reader, which was similarly affected (CVE-2010-1239), but with the exception that it did not display a warning message, rather it simply executed the embedded file directly.

Clearly, there is a distinction between the two vulnerabilities. NVD, however, failed to reflect this with the assigned CVSS rating, making it seem as if the vulnerabilities were equally severe in the two applications. While it can be argued that this in part is due to NVD having assigned an invalid score to the Adobe Reader issue, it is also due to the CVSSv2 guidelines not being clear on how to deal with follow-up impacts. Should the core problem be scored alone or should follow-up impact be considered?

In the case of Foxit Reader, it is immediate code execution. In the case of Adobe Reader, it is partial spoofing of the dialog with the potential of code execution if a user is successfully tricked.

Both approaches can be argued for with valid points:

1) Rating a vulnerability based on its immediate impact makes sense from a technical standpoint; in case of the Adobe Reader vulnerability, it is partial spoofing of a warning dialog, which should be properly reflected in the score in order not to mislead users. The follow-up impact would only be relevant **if** a user is successfully tricked. Scoring vulnerabilities based on a potential ultimate impact, if successful, is irrelevant for the score and if taken into account, there will be less granularity between issues as many issues would end up with the same potential system compromise scoring. Also, if follow-up impacts should be considered, where should the line be drawn when evaluating what a realistic follow-up impact is?

If we follow this approach, the Adobe Reader vulnerability should be rated as: (AV:N/AC:M/Au:N/C:N/I:P/A:N) = 4.3.

2) Rating based on any realistic/expected follow-up impact makes sense as that provides the most value to users. Not scoring the ultimate impact to properly reflect the threat may result in improper proper prioritization by system administrators when addressing vulnerabilities; a severe issue could seem minor and be erroneously dismissed. As such, full potential impact should be scored, and AC should be used to tweak the score based on the requirements and amount of interaction needed to successfully achieve the ultimate impact.

If we follow this approach, the vulnerability should be rated as: (AV:N/AC:H/Au:N/C:C/I:C/A:C) = 7.6.

The Manipulation Meets Disclosure Awkwardness

Another case to consider is a Cross-Site Request Forgery (CSRF) attack that allows an attacker to manipulate an administrative password if a user is successfully tricked into following a specially crafted link.

The baseline for scoring a CSRF vulnerability is (AV:N/AC:M/Au:N/C:N/I:P/A:N) = 4.3. However, if an attacker can successfully change an administrative password to gain access to the web application, this could possibly allow uploading and executing arbitrary code with the web application's permissions. If that is the case, should the score be higher in order to reflect this e.g. (AV:N/AC:M/Au:N/C:P/I:P/A:P) = 6.8? Worse, should it be scored like default administrator credentials often are, e.g. (AV:N/AC:L:Au:N/C:C/I:C/A:C) = 10.0?

It is clear that steps need to be taken to ensure that CVSSv3 better supports follow-up impacts in the form of more detailed guidelines and/or by providing better scoring options as previously discussed.

The Library Predicament

Software is an increasingly complex beast. Gone are the days of a thousand lines of code making up a desktop application. Instead, we now see common applications built on the back of 50 or more open source products, integrating hundreds of thousands of lines of code from hundreds of developers that did not sit down to work together.

A vulnerability that results in a context-dependent denial of service in a popular library, as audited with just that library in mind is relatively easy to score. What isn't considered by many, is what happens when that library is used in other software. That exact same vulnerability may have a considerably more severe impact depending on how the functionality is implemented. Scoring on the "base" vulnerability is a moving target when considered in the context of each individual implementation.

Unfortunately, this may truly be one element that can never fully be addressed by any scoring system.

The Conclusion Certainty

The one thing we can be certain of, is that CVSS needs an overhaul. As outlined, CVSSv2 has too many shortcomings to provide an adequate and useful risk scoring model. With the upcoming CVSSv3 standards, we hope that the working group evaluates these weaknesses, as well as how scoring has held up over the last 89,000 vulnerability disclosures.

Risk Based Security, Inc.

Risk Based Security was established to support organizations with the technology to turn security data into a competitive advantage. Using interactive dashboards and search analytics, RBS offers a first of its kind risk identification and security management tool. RBS further complements the data analytics and vulnerability intelligence with risk-focused consulting services, to address industry specific information security and compliance challenges including ISO/IEC 27001:22005 consulting. http://www.riskbasedsecurity.com

Open Security Foundation

Open Security Foundation is a 501(c)(3) non-profit public organization founded and operated by information security enthusiasts. We exist to empower all types of organizations by providing knowledge and resources so that they may properly detect, protect, and mitigate information security risks. http://www.opensecurityfoundation.org