Is Vulnerability Report Confidence Redundant? Pitfalls Using Temporal Risk Scores

François Boechat, Gabriel Ribas, Lucas Senos, Miguel Bicudo, and Mateus Schulz Nogueira | Federal University of Rio de Janeiro
Leandro Pfleger de Aguiar | Siemens Technology
Daniel Sadoc Menasché | Federal University of Rio de Janeiro

The Common Vulnerability Scoring System score is the de facto standard to assess risk of software vulnerabilities, with three temporal components: exploitability, remediation level, and report confidence. We discuss how the latter may be inferred from the first two, pointing practical and conceptual issues in the usage of temporal risk scores.

Confidence is at the core of information systems security. Confidence level assessment, e.g., with respect to sources or security advisory contents, is key for decision making. In essence, confidence relates to trust, being higher for authoritative sources, and to timeliness, given that advisories may eventually become outdated.

Assessing and conveying confidence in the realm of software vulnerabilities is a challenging task. In statistics, confidence in a given estimate, e.g., of risk, is captured through a confidence interval, which is reported together with the estimate of the metric of interest. This strategy of factoring out the confidence from the estimate has clear advantages, such as allowing the visualization of the most likely values of the metric of interest. Nonetheless, when assessing risk levels for software vulnerabilities, confidence typically has been factored into the estimate of risk itself.

The Common Vulnerability Scoring System (CVSS) score is one of the de facto solutions for assessing the risk associated with software vulnerabilities.[1,2] Its temporal score comprises three dimensions, namely *exploitability level* (*EX*), *remediation level* (*RL*), and *report confidence* (*RC*). The three dimensions capture the maturity level of known exploits, countermeasures, and security advisories for a given vulnerability. Correctly assigning CVSS scores has safety, economic, and legal implications.[3] The main goals of this article are to discuss, from practical and conceptual points of view, whether *RC* as considered by the CVSS is a redundant dimension, i.e., whether it can be inferred from the first two, and to motivate novel approaches to convey confidence levels for software vulnerability risk estimates.

A CVSS Primer

Each vulnerability has an identifier (ID) given by the Common Vulnerability and Exposures Identifier (CVE ID). The vulnerability with an ID of CVE-2021-0001 was the first vulnerability to be published in 2021. The CVSS score measures the risk of the vulnerability. The overall CVSS score is quantified in the range of [0, 10], with 0 being the least severe and 10 the most severe. CVSS scores are computed from base, temporal, and environmental metric groups. Base metrics are divided into exploitability and impact dimensions. Exploitability metrics comprise access vector (local, adjacent network, or network), access complexity (low, medium, or high), and authentication (none, single, or multiple). For impact metrics, confidentiality (none, partial, or complete), integrity (none, partial, or complete), and availability (none, partial, or complete) are used.

Given the base score, temporal and environmental metrics are used to produce final risk estimates. In this work, we focus on temporal metrics, which can perturb

Digital Object Identifier 10.1109/MSEC.2021.3070978
Date of current version: 5 May 2021

 Copublished by the IEEE Computer and Reliability Societies 1540-7993/21©2021IEEE

the CVSS base score to produce a final risk estimate ranging between 66 and 100% of the CVSS base score value. For temporal metrics, *RC*, *EX*, and *RL* are used.

RC

The *RC* of a vulnerability can be confirmed (C), reasonable (R), or unknown (U). It is defined as confirmed by the CVSS standards if it satisfies the following definition.

Definition 1 (Confirmed vulnerability). *A vulnerability is confirmed if a detailed report exists, or functional reproduction is possible (functional exploits may provide this).*

Note that this definition naturally implies a dependency between *RC*, as defined by CVSS standards, and exploit code maturity.

EX

The maturity of exploits is captured by CVSS temporal scores through its exploit code maturity level, or exploitability (*EX*). The maturity is set to its lower level, unproven (U), when no exploit is known. The next level, proof of concept (PoC), corresponds to a code that would require substantial modification by a skilled attacker to work in most systems. An exploit code has functional maturity level (F) if the code works in most situations where the vulnerability exists. A high maturity level (H) is reached if, in addition, the exploit code works in every situation or is actively being delivered via an autonomous agent.

Then the following corollary follows from the previous definition.

Corollary 1. *A vulnerability is confirmed if a functional or high maturity exploit exists.*

RL

The temporal scores also account for the *RL*, which ranges between unavailable (U), workaround (W), temporary fix (T), and official fix (O). The last three correspond to an unofficial nonvendor solution, an official but temporary fix, and a complete vendor solution, respectively.

Illustrative Example

As an example, consider CVE-2015-5374, which impacts Siemens SIPROTEC relays. The base subscores are 6.9 and 10.0 for impact and exploitability, respectively, producing a final base score of 7.8, which corresponds to a high risk, given that SIPROTEC has an important role in power distribution environments. It is well known that the vulnerability was exploited by a malware (Industroyer/Crashoverride). Therefore, at the Siemens website, the temporal metrics of *EX* and *RC* are set to high and confirmed, respectively, and the *RL* is an official fix. Overall, the temporal score is 6.8. Indeed, the existence of an official fix reduces the score from its base value of 7.8 to 6.8. Interestingly, at IBM X-Force, the same vulnerability is reported with a corresponding exploit maturity level of functional, which yields a temporal score of 6.4. The difference between CVSS score assignments at IBM X-Force and Siemens may be due to evolution or inconsistency. In this particular case, IBM X-Force reported its CVSS values before Siemens and did not update its values afterward. Whereas IBM X-Force is a general platform that reports CVSS values for most of the existing vulnerabilities, Siemens is a more focused source of information about vulnerabilities that directly or indirectly impact its products.

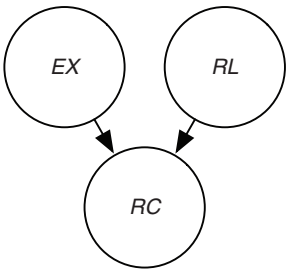
Reference Data Set for CVSS Temporal Scores

We consider as our reference data set the public IBM X-Force containing information about 72,545 vulnerabilities, identified by their CVEs, and the corresponding temporal score metrics (see https://www.ibm.com/security/xforce). The considered vulnerabilities were published in the National Vulnerability Database (NVD) between 1 January 1997 and 26 April 2019. When both CVSS 2.0 and 3.0 were available, we used the latter, assuming a natural matching between the subscores in the two versions; e.g., an uncorroborated *RC* in CVSS 2.0 would correspond to a reasonable *RC* in CVSS 3.0. Among those CVEs, 179 corresponded to vulnerabilities for which at least one of the three temporal score metrics was not set, and 72 had *RC* set to unconfirmed, unknown, or not defined, with little impact toward our goal. We removed those vulnerabilities and were left with 72,294 CVEs. Each of those vulnerabilities is classified with an *RC* of either reasonable (R) or confirmed (C).

While analyzing the preceding data set, we discovered that *RC* can be inferred with high accuracy from *EX* and *RL*. To illustrate that point, Figure 1 shows the Bayesian network obtained from the data set. It indicates that as far as there is a patch for a vulnerability, i.e., *RL* is different from unavailable, the vulnerability is confirmed.

Note that Corollary 1 is in sharp contrast against the first line of the table in Figure 1. According to Figure 1, in the absence of patches, a vulnerability is likely not confirmed even if a high maturity exploit exists. The disagreement between Corollary 1 and Figure 1 may be attributed to a number of reasons, including the mislabeling of *EX*, *RL*, or *RC*.

It is natural to assume that if *EX* is high and *RL* is unavailable, there is a high uncertainty with respect to one of those two dimensions. Releasing a CVE is a multistage process, and CVEs to exploitable vulnerabilities are typically released only after at least a workaround or a temporary fix is made available. Then *RC* may be set to



Exploit Code Maturity (*EX*)	Remediation Level (*RL*)	Vulnerability Report Confidence Confirmed $P(RC = C \mid E, RL)$
High	Unavailable	0.01
High	Workaround ~ Official Fix	0.98
Unproven ~ Functional	Unavailable	0.18
Unproven ~ Functional	Workaround ~ Official Fix	0.99

Figure 1. A Bayesian network captures the intuitive idea that as far as there is a remediation available, the vulnerability is confirmed.

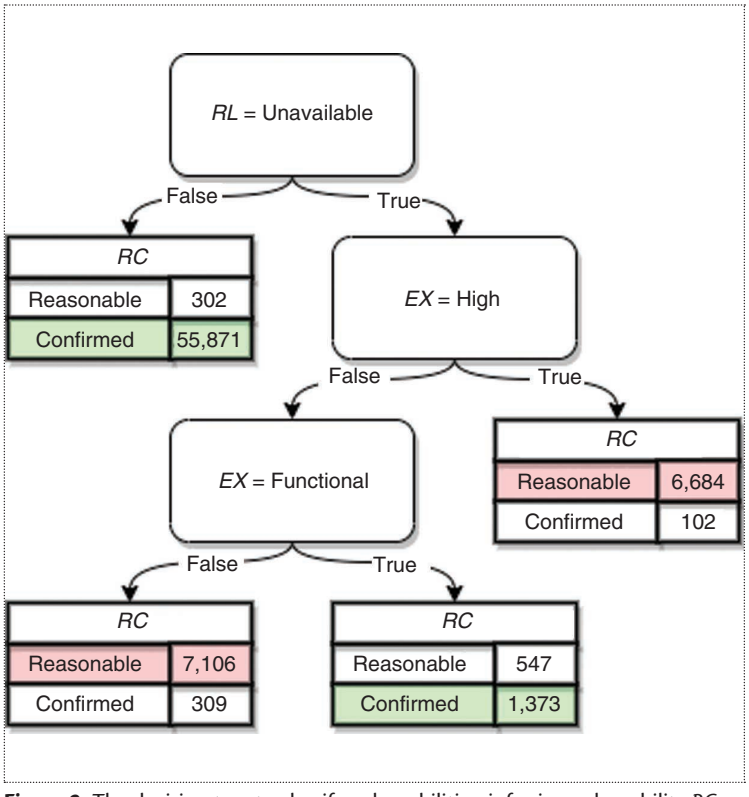


Figure 2. The decision tree to classify vulnerabilities, inferring vulnerability *RC* from *EX* and *RL*.

reasonable to indicate that *EX* or *RL* needs adjustment, reflecting that the CVSS score assignment requires adjustments in the near future.

In what follows, we leverage the predictability of *RC* to 1) motivate potential adjustments in existing data sets, indicating the challenges involved in such endeavor, and 2) foster a reflection on the role of *RC* as an independent dimension in the CVSS standard.

We summarize our key findings as follows:

1. Assigned values for *RC* and exploit maturity sometimes violate both intuition and the CVSS specification (Definition 1).
2. Leveraging public data in ExploitDB and NVD can help to avoid these violations.
3. Once temporal score assignments abide to CVSS specifications, the confidence score as defined by the CVSS specification is typically redundant.
4. Different organizations use different strategies to assign temporal scores, creating discrepancies.
5. The evolution of exploits and defenses accounts for some discrepancies, driving home the point that these metrics are temporal and should be kept up to date to maintain utility.

Inferring *RC*

Next, we report our results on the training and evaluation of decision trees to infer *RC* from *EX* and *RL*. We train a C4.5 decision tree, using 10-fold cross-validation, to classify our samples, with *RC* as the target class. The trained decision tree is shown in Figure 2. Each internal node in the tree corresponds to a test based on a feature, each branch corresponds to an outcome, and leaves correspond to target classes. Nodes closer to the root of the tree are associated with features that have higher discriminatory power. In the obtained tree, the root corresponds to the test of whether *RL* is unavailable ($RL = U$). If the answer to the test is negative, i.e., if there is a patch available, *RC* is set to confirmed.

A Counterintuitive Branch

In Figure 2, when *RL* is unavailable ($RL = U$), i.e., no patch is available, and exploitability is high ($EX = H$), the outcome is an *RC* of reasonable ($RC = R$). This is counterintuitive, given that if exploitability is reduced to functional ($EX = F$), we have a confirmed *RC* ($RC = C$).

Finding 1. *Assigned values for RC and exploit maturity sometimes violate both intuition and the CVSS specification.*

Why does the increase in the level of exploit code maturity from functional to high cause a decrease in the *RC* from confirmed to reasonable? One possible answer to that question is that *RC* actually reflects the confidence in the assignment of *EX* values.

Identifying the Need for Adjustment

We assume that vulnerabilities classified with a reasonable *RC*, for which an exploit with high maturity is marked as available, potentially need adjustment (see the rightmost branch in Figure 2). In light of Corollary 1, we pose the following questions: Toward which direction should the adjustment be implemented? Should the *RC* be increased? Or should the maturity level of the exploit be decreased?

We considered a number of different approaches to answer those questions, including the analysis of additional sources of temporal CVSS scores, such as Microsoft, McAfee, Cisco, and Siemens,[4] using the Common Vulnerability Reporting Framework (CVRF), and multiple sources with information about exploits and weaponization, including ExploitDB and NVD.[5]

As a first step, we consider the cross analysis of the maturity level of exploits against those reported by ExploitDB.

Leveraging ExploitDB to Assess Exploit Code Maturity

Next, we leverage ExploitDB to cross-validate the exploit code maturity for vulnerabilities that, according to IBM X-Force, have a high maturity exploit. To that aim, we matched vulnerabilities against exploits whose maturity we had independently established ourselves using the methodology described in the following section.

Determining EX From ExploitDB

For each vulnerability, we search for the corresponding exploits at ExploitDB. Then, we run a filtering algorithm to determine the corresponding *EX*. The filtering is implemented in layers.

At the first layer, we identify whether the author of the exploit is "Metasploit." If so, the corresponding vulnerability is filtered, and its exploitation maturity level is set as high, $EX = H$, given that the Metasploit framework typically publishes mature exploits. Otherwise, at the subsequent layer, we filter and label vulnerabilities as $EX = PoC$ if the corresponding exploit contains the keyword PoC in its title or description, or if we find an outline of how the exploit works rather than its low-level source code. For the vulnerabilities not filtered so far, we identify whether the corresponding exploitation was verified by the ExploitDB team and if a vulnerable application accompanies the exploit. If the two conditions are satisfied (respectively, none is satisfied), we let $EX = H$ (respectively, $EX = PoC$). In all of the other cases, we set exploitability to functional ($EX = F$), as shown in Figure 3.

Relabeling the Data Set

From the data we gathered from IBM X-Force, we have 6,786 cases where the *RL* is unavailable and exploitability is high. We let $\mathcal{M}$ be the set containing those vulnerabilities, with $|\mathcal{M}| = 6{,}786$ Those vulnerabilities, when crossed with ExploitDB, garner us 4,737 matches, from which we have 1,196 PoC, 3,537 functional, and four high maturity exploits. The set of 1,196 + 3,537 vulnerabilities contained in $\mathcal{M}$ for which *EX* estimated by our heuristics mismatched IBM X-Force is denoted as $\mathcal{M}'$.

We relabeled the *EX* of the vulnerabilities in $\mathcal{M}'$, from high to either PoC or functional. For the remainder of the vulnerabilities in $\mathcal{M} \backslash \mathcal{M}'$, we relabeled the *RC* from reasonable to confirmed. Then we retrained

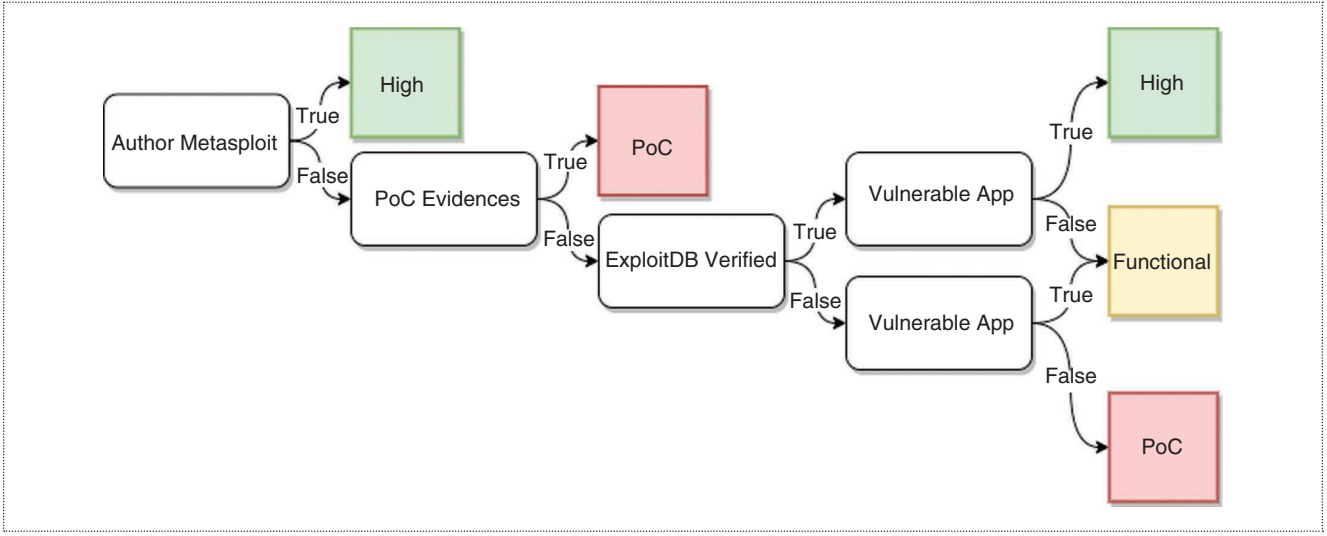


Figure 3. A decision tree to classify exploits, which is instrumental in fixing mislabeled samples through cross analysis of IBM X-Force data and ExploitDB.

our decision tree and obtained the same structure as the original one shown in Figure 2.

In the modified tree, the leaf corresponding to an available *RL* remains equal to that in Figure 2. This happens because we did not change the *RL* dimension of the data set. From left to right, the other leaves in the modified tree comprise 318, 1,379, and 2,625 CVEs marked as confirmed, and 8,227, 3,574, and zero CVEs marked as reasonable, respectively. The increase in the number of CVEs marked as confirmed as we move from left to right in the tree is in agreement with an increase in the level of *RC* as the exploit code maturity grows. In addition, in the resulting decision tree, if *RL* is unavailable ($RL = U$) and the exploit code maturity is high ($EX = H$), we have vulnerability *RC* confirmed ($RC = C$), which is in agreement with Corollary 1.

Finding 2. *Leveraging public data in ExploitDB and NVD can help to avoid violations in temporal score assignments.*

Limitations of ExploitDB

Although it is reassuring to verify that ExploitDB may be instrumental in refining the assessment of exploit code maturity, the previous analysis has at least two limitations.

- *Limited scope*: The fact that certain vulnerabilities have no corresponding exploit with high exploit code maturity at ExploitDB does not mean that these exploits are not available at other sources. Indeed, we also considered a number of other sources, including black hat forums, in search of exploits for the considered vulnerabilities. Our findings regarding those additional sources are discussed in the following section.
- *Persisting inconsistencies*: After the adjustments accounting for ExploitDB, we were still left with a branch in the decision tree wherein *RC* is reasonable, as opposed to confirmed, even in the presence of fully functional exploits. Such a branch goes counter to Corollary 1, requiring further attention. As we will indicate in what follows, leveraging additional sources of information, in particular the NVD and its external links, is instrumental in obtaining insight into that matter.

Additional Sources of Exploits: Black Hat Forums and Threat Feeds

We considered four additional sources for exploits: a Russian market,[6] a list of CVEs reported at the Exploit Prediction Scoring System (EPSS) special-interest group (SIG) mailing list,[7] CrimeBB,[8] and open (TLP white) TI feeds from MISP (Malware Information Sharing Platform).[9] We found a small number of matches among the CVEs reported by those sources and the CVEs corresponding to the contending rightmost branch in our decision tree. In particular, we were unable to find any matches among the CVEs reported by the first two sources and the CVEs corresponding to the contending rightmost branch in our decision tree. For the two latter sources, we found 13 matches, which serve to shed some insight on the challenge involved in the assignment of temporal risk scores based on those forums.

Both CrimeBB and MISP threat feeds are unstructured. In those platforms, information is provided in free text. After reading the posts and events citing the 13 CVEs referred to previously, we learned that five of the 13 elements were cited in a context unrelated to exploitability or weaponization incidents. The remaining eight CVEs are mentioned either at CrimeBB or at threat feeds in events associated to exploits or weaponization. They correspond to confirmed vulnerabilities, and for five out of the eight vulnerabilities, namely CVE-2008-0908, CVE-2008-2758, CVE-2012-5868, CVE-2013-0337, and CVE-2014-9438, the proposed heuristic discussed in the previous section allowed us to set the *RC* to confirmed.

Figure 4 provides further insight into the exploit code maturity for vulnerabilities cited at black hat forums and threat feeds. Using IBM X-Force to assign temporal scores to vulnerabilities, we find that the exploit code maturity at those forums is typically functional, whereas the exploit code maturity of vulnerabilities reported by official vendors is usually lower (PoC or unproven). A small fraction of the vulnerabilities cited in the considered forums has high exploit code maturity, in agreement with the results obtained with the heuristic discussed in the previous section as well as with the findings reported in what follows.

NVD Analysis

Given the challenges in gathering statistically relevant information about CVEs from black hat forums and threat feeds, in what follows we leverage the NVD for that purpose.[5] For each vulnerability, the NVD contains external links with additional information, some of which are classified with tags, such as "vendor advisory," "release notes," "patch," and "exploit." We adopted the following heuristics to assign temporal CVSS metrics based on NVD data:

1. *EX*: If the vulnerability has more than one link tagged with "exploit," *EX* is set to high. If there is a single link, *EX* is classified as functional. Otherwise, *EX* is assumed to be either PoC or unproven.
2. *RL*: If there is no link tagged with labels corresponding to remediation ("patch," "mitigation," "vendor advisory," or "release notes"), *RL* is classified as unavailable. Otherwise, we assume that at least a temporary fix is available.

3. *RC*: If the vulnerability contains at least one tagged link or at least five links, irrespective of being tagged, the vulnerability *RC* is set to confirmed. Otherwise, it is assumed to be reasonable.

We applied the preceding heuristics to the counterintuitive rightmost branch in Figure 2 to produce Figure 5(a). We also applied the heuristics throughout all vulnerabilities in the NVD data set to produce Figure 5(b). In both cases, we observe that the vast majority of vulnerabilities have a remediation. In addition, *RC* increases as *EX* grows. Finally, for most vulnerabilities, we have either a PoC or an unproven exploit, which is in agreement with Figure 4.

Finding 3. *Once temporal score assignments abide to CVSS specifications, the confidence score as defined by the CVSS specification is typically redundant.*

Additional Sources of Temporal Scores

Next, we further discuss four additional sources of temporal CVSS scores provided by Microsoft, Siemens, McAfee, and Cisco.

Microsoft

In November 2020, Microsoft started to share base and temporal CVSS scores for vulnerabilities affecting its products (see https://msrc.microsoft.com/update-guide). Of the 2,620 CVEs reported by Microsoft, 22 are set with a reasonable *RC*. From those 22 CVEs, 18 are from 2016, two are from 2017, one is from 2018, and one is from 2019. Those numbers suggest that CVEs classified with a reasonable *RC* occur because of legacy classification issues.

Siemens

Siemens reports CVSS scores for 480 CVEs affecting its products (see https://new.siemens.com/global/en/products/services/cert.html). All the vulnerabilities are marked as confirmed. According to IBM X-Force, there is a single CVE matching the rightmost branch of the decision tree of Figure 2: CVE-2017-12738. The vulnerability impacts Siemens SICAM Remote Terminal Units SM-2556 COM Modules, wherein the integrated webserver (port 80/TCP) of the affected devices could allow cross-site scripting attacks. Siemens provides an official security advisory for that vulnerability, with a corresponding mitigation.

McAfee

On 26 August 2020, we collected from the McAfee website information about 237 unique CVEs affecting its products (see https://www.mcafee.com/enterprise/en-us/threat-center/product-security-bulletins.html). Of those CVEs, four are marked with *RC* equal to unknown and 17 are marked with *RC* equal to reasonable. McAfee removes information from its website when it becomes stale. Of 17 CVEs marked with a reasonable *RC*, seven were removed from the McAfee website by the end of November, two were clear typos (as the same field had a different value at the same webpage), and three were explored with other CVEs marked as confirmed in the same group. This left us with five CVEs marked as reasonable that did not fall into the previous categories. Of those five CVEs, four are related to a known OpenSSL bug (CVE-2017-3735 up to CVE-2017-3738). In addition, we found a single match between the McAfee database and the CVEs corresponding to the rightmost branch in the decision tree in Figure 2: CVE-2014-2587, which is confirmed by McAfee but not classified so by IBM X-Force.

Cisco

Cisco reports temporal CVSS scores for 3,877 CVEs (see https://tools.cisco.com/security/center/publicationListing.x). Out of those vulnerabilities, we found eight matches within the rightmost branch in the decision tree in Figure 2. All of those vulnerabilities are reported as confirmed by Cisco.

Tracking Evolution

Table 1 compares the CVSS temporal subscores reported by IBM X-Force against those reported by the considered vendors. Whereas the *RC* and *RL* typically agree, *EX* has significant disagreements.[5]

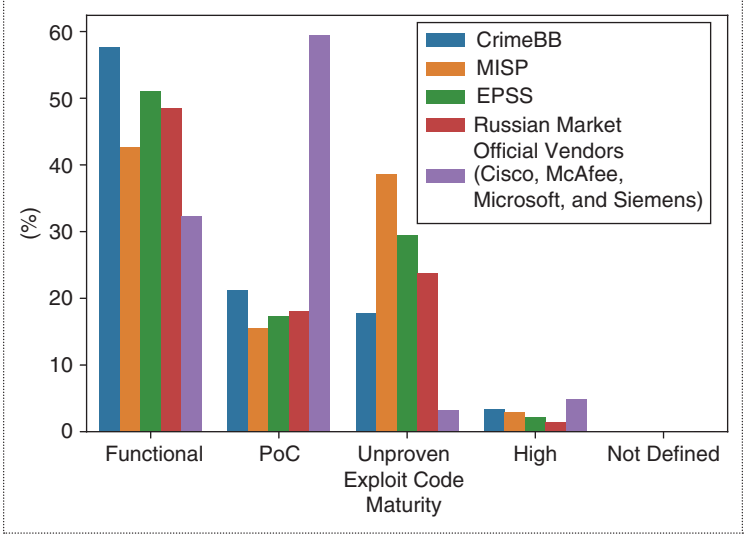


Figure 4. Black-hat forums and threat feeds provide additional sources of exploits. Using IBM X-Force to assign vulnerabilities temporal scores, we find that the exploit code maturity at those forums is typically functional, whereas exploit code maturity of vulnerabilities reported by official vendors is typically lower (PoC or unproven). A small fraction of vulnerabilities have high exploit code maturity.

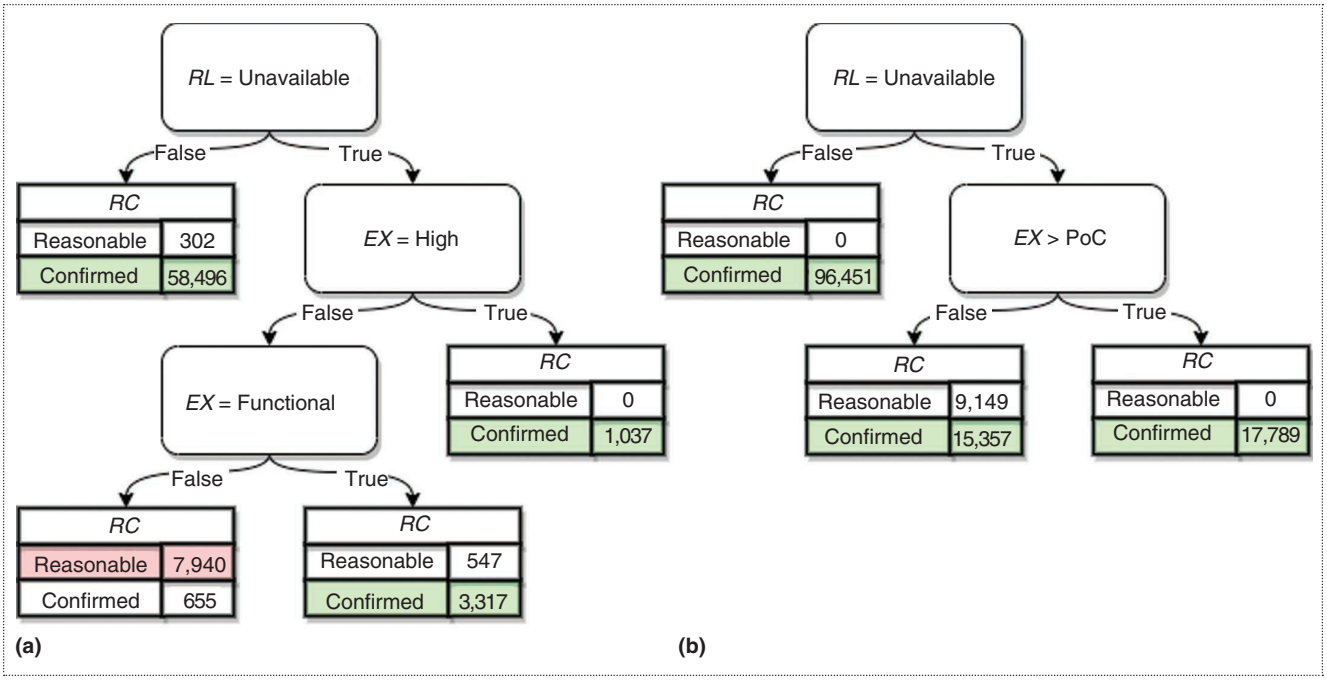


Figure 5. A decision tree to classify vulnerabilities, leveraging IBM X-Force and NVD.

Table 1. Tracking evolution of temporal scores across multiple vendors.

IBM X-Force	Microsoft	McAfee	Siemens	Cisco
Mean timespan	2.53 days	884 days	605 days	45.3 days
EX agree	45.36%	44.44%	9.36%	5.42%
RC agree	97.59%	84.92%	97.75%	99.15%
RL agree	98.85%	85.71%	90.26%	74.22%
A fraction of CVEs for which scores are attributed at different days and CVSS disagreement corresponds to evolution as opposed to inconsistency.				
EX evolved	0.34%	45.24%	57.68%	28.81%
RC evolved	0.11%	3.97%	1.12%	0.21%
RL evolved	0.46%	6.35%	3.75%	4.47%
A fraction of CVEs for which scores are attributed at same day but CVSS scores disagree.				
EX disagree, same day	54.07%	0.79%	17.98%	49.92%
RC disagree, same day	2.29%	0.00%	1.12%	0.32%
RL disagree, same day	0.69%	0.00%	3.75%	10.53%
A fraction of CVEs for which scores are attributed at different days and CVSS disagreement corresponds to inconsistency.				
EX inconsistent	0.23%	9.52%	14.98%	15.84%
RC inconsistent	0.00%	11.11%	0.00%	0.32%
RL inconsistent	0.00%	7.94%	2.25%	10.79%

Finding 4. *Different organizations use different strategies to assign temporal scores, creating discrepancies.*

To assess the root cause of such disagreements, we further analyzed the fraction of disagreements that may be attributed to temporal evolution as opposed to inconsistency. For example, if IBM X-Force reports a PoC, but at a future instance, Siemens reports the existence of an exploit with high exploit code maturity, such a change is attributed to an evolution as opposed to an inconsistency. More broadly, if two distinct sources assign CVSS temporal scores on different days, and such scores correspond to an increase in *EX*, *RC*, or *RL*, we classify such a change as an evolution as opposed to an inconsistency. Note that we were only able to track changes across different sources as individual sources usually do not report change history.

As shown in Table 1, between IBM X-Force and Siemens, we find that roughly 57% of disagreements in exploit code maturity may be due to temporal evolution. The disagreement between Microsoft and IBM X-Force, by contrast, is the most relevant, and only a small fraction of the mismatches correspond to CVEs for which scores are attributed at different dates, corresponding to an evolution.

Table 1 also reports the mean timespan between the date at which CVSS scores were attributed by IBM X-Force and other vendors. The mean timespan is computed as the average of the absolute values of the

differences between the dates at which the CVSS scores were reported by the vendors and by IBM X-Force. Interestingly, there is a positive correlation between the timespan and the fraction of disagreements between exploit code maturity levels that may be attributed to an evolution in the CVSS scores. This suggests that when scores are set close to the date at which the vulnerability is disclosed, there is typically no communication among the parties involved. Nonetheless, when the time gap increases, exploit code maturity levels are typically updated based on available information, capturing a natural tendency of the exploits to become more mature.

Figure 6 provides further insight into the evolution of CVSS temporal scores across different sources. Whenever two sources publish CVSS temporal scores about a given CVE, we produce a corresponding chronologically ordered pair. Each element of the pair comprises a source (Microsoft, IBM X-Force, Cisco, Siemens, or McAfee) and a CVSS temporal score. The width of the lines is proportional to the CVE flow between platforms accounting for the corresponding CVSS temporal score assignments, leveraging the produced ordered pairs. Only lines corresponding to more than 40 CVEs are reported (in Table 1, by contrast, all CVEs are taken into account). In addition, exploit code maturity and sources are reported in all cases, but *RL*s are explicitly reported only for scenarios wherein they are different from an official fix.

All the vulnerabilities that contribute to Figure 6 have a confirmed *RC*. As an example, the width of the line from "Unproven at IBM X-Force" to "Functional at Cisco" is proportional to the frequency of ordered pairs corresponding to vulnerabilities classified with an unproven exploit code maturity at IBM X-Force and later on with a functional exploit code maturity at Cisco. Blue and red lines correspond to evolution and inconsistency, respectively. Figure 6 shows that most vulnerabilities evolve across platforms, first being published by IBM X-Force and later on having their exploit code maturity adjusted. Nonetheless, a notable exception corresponds to CVEs first classified by IBM X-Force and eventually reclassified by Cisco as having a functional exploit code maturity. In particular, all CVEs first classified by IBM X-Force with a high exploit code maturity were eventually reclassified by Cisco with a functional exploit code maturity. Such inconsistency is in agreement with the discussion derived from ExploitDB, corroborating the analysis to produce Figure 5.

Finding 5. *Evolution of exploits and defenses accounts for some discrepancies in temporal score assignments, driving home the point that these metrics are temporal and should be kept up to date to maintain utility.*

For the vast majority of vulnerabilities, *RC* can be inferred from *RL* and *EX*. From a conceptual point of view, we believe that it is necessary to better define the role of *RC* in the CVSS framework, together with a data-driven methodology to assign its values. From a pragmatic point of view, we envision at least three roles for *RC*. First, *RC* may be helpful in fixing misclassified data sets, as illustrated in this article (see also the works by Thomas et al.[4] and Dong et al.[5]). Second, *RC* may be useful when both *RL* and *EX* are unknown. However, this may occur only during a short window of time, as vulnerabilities are typically disclosed to the public when at least one remediation is available or a PoC is known (e.g., from bug bounty programs). Third, it follows from Figure 2 that, given two elements from *EX*, *RC* and *RL*, the third can be roughly inferred. While assigning temporal CVSS scores, it may be easy to set *RC*, e.g., from security advisories, and *RL*, e.g., from the vendor website. In that vein, *RC* may be instrumental in inferring exploit code maturity.

RC is intrinsically correlated with the number of references to a vulnerability, as observed in websites such as GitHub and NVD[10] as well as underground markets,[6] black hat forums,[8] and threat intelligence feeds.[9] The number of references, in turn, is a key ingredient to infer exploit code maturity, e.g., in EPSS[7] and related

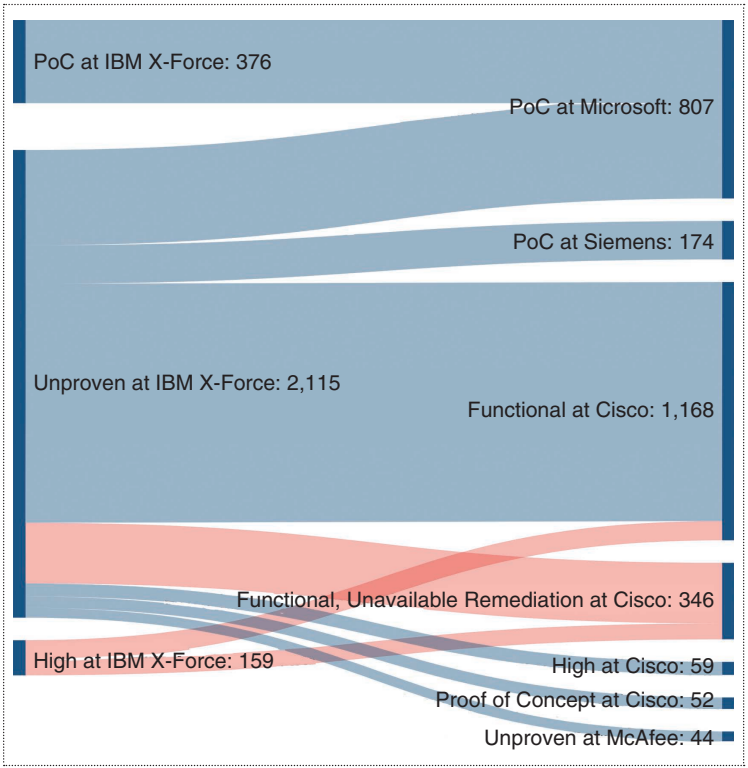


Figure 6. A Sankey diagram showing how different sources assign CVSS temporal scores. Blue and red lines correspond to evolution and inconsistency, respectively.

efforts.[11] Such interplay between *RC* and exploit code maturity, through the number of references, further evidences the opportunity to use the latter as a proxy to the confidence in the exploit code maturity.

Since the first version of the CVSS, released in 2003, the economic and legal implications of risk assessment have been expanding.[1,3,12] In particular, *RC* has been one of the three fundamental dimensions for temporal risk assessment under the CVSS framework, together with *EX* and *RL*.[2] While writing this manuscript, we learned from personal communication that the CVSS SIG has voted to remove the *RC* metric from the next iteration of the CVSS, namely CVSS 4.0, for considerations similar to those presented in this study. By organizing the subsumed ideas in a structured fashion and reporting findings emerging from data, we have taken an additional step toward evidencing the role of *RC* in risk assessment. We envision that this work opens up an interesting avenue for future research on automated and explainable assignment of temporal risk scores from data, with confidence intervals, following efforts that have done so for the base CVSS metrics.[13–15]

Acknowledgments

This work was supported in part by CAPES, CNPq and FAPERJ through grants E-26/203.215/2017 and E-26/211.144/2019.

References

1. J. Spring, E. Hatleback, A. D. Householder, A. Manion, and D. Shick, "Prioritizing vulnerability response: A stakeholder-specific vulnerability categorization," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, White Paper, 2019. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=636379
2. P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system," *IEEE Security Privacy*, vol. 4, no. 6, pp. 85–89, 2006. doi: 10.1109/MSP.2006.145.
3. "First American: DFS blames deficient security controls." Panaseer. 2020. https://panaseer.com/business-blog/first-american-deficient-security-controls/ (accessed Apr. 21, 2021).
4. R. J. Thomas, J. Gardiner, T. Chothia, E. Samanis, J. Perrett, and A. Rashid, "Catch me if you can: An in-depth study of CVE discovery time and inconsistencies for managing risks in critical infrastructures," in *Proc. 2020 Joint Workshop on CPS&IoT Security Privacy*, 2020, pp. 49–60. doi: 10.1145/3411498.3419970
5. Y. Dong, W. Guo, Y. Chen, X. Xing, Y. Zhang, and G. Wang, "Towards the detection of inconsistencies in public security vulnerability reports," in *Proc. 28th USENIX Security Symp. (USENIX Security 19)*, 2019, pp. 869–885.
6. L. Allodi, "Economic factors of vulnerability trade and exploitation," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Security*, pp. 1483–1499.
7. J. Jacobs, S. Romanosky, B. Edwards, M. Roytman, and I. Adjerid, "Exploit prediction scoring system (EPSS)," 2019, arXiv:1908.04856.
8. S. Pastrana, D. R. Thomas, A. Hutchings, and R. Clayton, "CrimeBB: Enabling cybercrime research on underground forums at scale," in *Proc. 2018 World Wide Web Conf.*, 2018, pp. 1845–1854.
9. C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "MISP: The design and implementation of a collaborative threat intelligence sharing platform," in *Proc. ACM Workshop on Inform. Sharing Collaborative Security*, 2016, pp. 49–56.
10. L. Miranda et al., "Measuring and modeling software vulnerability security advisory platforms," in *Proc. 15th Int. Conf. Risks Security Internet Syst.* (CRiSIS 2020), Paris, France, Nov. 4–6, 2020, p. 31.
11. O. Suciu, C. Nelson, Z. Lyu, T. Bao, and T. Dumitras, "Expected exploitability: Predicting the development of functional vulnerability exploits," 2021, arXiv:2102.07869.
12. A. D. Householder, J. Chrabaszcz, T. Novelly, D. Warren, and J. M. Spring, "Historical analysis of exploit availability timelines," in *Proc. 13th USENIX Workshop on Cyber Security Experimentation Test (CSET 20)*, 2020.
13. C. Elbaz, L. Rilling, and C. Morin, "Fighting n-day vulnerabilities with automated CVSS vector prediction at disclosure," in *Proc. 15th Int. Conf. Availability, Reliability Security*, 2020, pp. 1–10.
14. A. Khazaei, M. Ghasemzadeh, and V. Derhami, "An automatic method for CVSS score prediction using vulnerabilities description," *J. Intell. Fuzzy Syst.*, vol. 30, no. 1, pp. 89–96, 2016. doi: 10.3233/IFS-151733.
15. M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: Learning to classify vulnerabilities and predict exploits," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining*, 2010, pp. 105–114.

François Boechat is a B.S. undergraduate in computer science at the Federal University of Rio de Janeiro, Rio de Janeiro, 21941-590, Brazil. His research interests include model-based analysis of computer systems, with a focus on cybersecurity. Contact him at fran_5298@hotmail.com.

Gabriel Ribas is a B.S. undergraduate in computer science at the Federal University of Rio de Janeiro, Rio de Janeiro, 21941-590, Brazil. His research interests include analytical models and performance evaluation. Contact him at gabrielribas4@gmail.com.

Lucas Senos is a B.S. undergraduate in computer science at the Federal University of Rio de Janeiro, Rio de Janeiro, 21941-590, Brazil. His research interests include systems security and data analytics. Contact him at lucassenos@hotmail.com.

Miguel Bicudo is a senior B.S. undergraduate in computer science at the Federal University of Rio de Janeiro, Rio de Janeiro, Brazil. His research interests include modeling and analysis of systems, with focus on security and data analytics. Contact him at masbicudo@gmail.com.

Mateus Schulz Nogueira is currently pursuing his M.Sc. in informatics at the Federal University of Rio de Janeiro, Rio de Janeiro, 21941-590, Brazil. His research interests include modeling and analysis of systems, with a focus on machine learning. Contact him at msznogueira@gmail.com.

Leandro Pfleger de Aguiar is a principal key expert on the topic of industrial control systems security and security services lifecycle at Siemens Technology, Princeton, New Jersey, USA. Aguiar received an M.Sc. in industrial data mining from the Federal University of Minas Gerais, MG, Brazil. As part of his role at Siemens, he supports business units across various industry verticals like digital industries, healthcare, mobility, and energy, developing and enhancing security capabilities of products and services. He holds several patents in this field. Contact him at leandro.pfleger@siemens.com.

Daniel Sadoc Menasché is an associate professor in the Department of Computer Science at the Federal University of Rio de Janeiro, Rio de Janeiro, 21941-590, Brazil. His research interests include modeling, analysis, security and performance evaluation of computer systems. Menasché received a Ph.D. in computer science from the University of Massachusetts, Amherst, in 2011. He was coauthor of papers that were awarded best paper awards at IEEE Globecom 2007, Association for Computing Machinery CoNext 2009, IEEE Infocom 2013. and the 2015 International Conference on Global Software Engineering. Contact him at sadoc@dcc.ufrj.br.

Digital Object Identifier 10.1109/MSEC.2021.3087875