

The `cvss` package*

Pierre VIVEGNIS†

Released 2022/12/02

Contents

1	Introduction	3
2	Acknowledgements	3
3	Usage	3
3.1	Direct Macros	3
3.2	Indirect Macros	4
4	Examples	5
4.1	Direct Form	5
4.2	Imbricated Form	5
4.3	Test Computations	6
5	Implementation	7
5.1	Initial set up	7
5.2	Round up function	7
5.3	Error messages	7
5.4	CVSS metrics parsing	7
5.4.1	Attack Vector	8
5.4.2	Attack Complexity	8
5.4.3	Privileges Required	8
5.4.4	User Interaction	9
5.4.5	Confidentiality, Integrity and Availability	10
5.5	CVSS computation	10
5.5.1	Impact Sub Score (ISS)	10
5.5.2	Impact	10
5.5.3	Exploitability	11
5.5.4	CVSS Base Score	11
5.5.5	CVSS Base Score	12
5.6	CVSS levels	14
5.7	Fancy prints	15
5.7.1	Framed CVSS Level	15
5.7.2	Full CVSS display	16

*This file describes version v1.1, last revised 2022/12/02.

†E-mail: pierre@vivegnis.be

1 Introduction

The `cvss` package allows the user to compute CVSS 3.1 base scores and use them in documents. The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the characteristics and severity of software vulnerabilities. CVSS consists of three metric groups: Base, Temporal, and Environmental.

This packages only deal with Base score. Temporal and Environmental scores will be part of a future release.

More information can be found at <https://www.first.org/cvss/specification-document>.

2 Acknowledgements

I want to thank Alexander Lill who first created a `cvss` project in \LaTeX (available at <https://github.com/AlexanderLill/cvss3tex>).

3 Usage

The goal of this package is to compute the CVSS base score for an input CVSS vector, and to give the user macro to output it in 3 different forms

- The CVSS **score** (from 0.0 to 10)
- the **level** (None, Info, Low, Medium, High or Critical)
- the **colored level**
- the **tag** which is a colored frame around the level

All macros are expandable, which makes them usable in any context.

The macros of this packages are divided in 2 categories:

- **direct macros** : that will take as input the CVSS base score and give you the result
- **indirect macros** : that are intermediary, in the way that they only compute a form based on the precedent one.

3.1 Direct Macros

<u><code>\cvssScore</code></u>	<code>\cvssScore {\textit{CVSS string}}</code>
--------------------------------	--

This is the main macro of this package, responsible for computing the base CVSS 3.1 score of an $\{\textit{input vector}\}$ (without CVSS3.1/). The output of this macro is a floating point CVSS score, for example 5.4.

`\cvssScore{CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:H/I:L/A:N}`

This will output the following CVSS base score: 5.3

<u><code>\cvssScorepretty</code></u>	<code>\cvssScorepretty {\textit{CVSS string}}</code>
--------------------------------------	--

This macro will print a **colored** base CVSS 3.1 score of an $\{\textit{input vector}\}$ (without CVSS3.1/). The output of this macro is a floating point CVSS score.

`\cvssScorepretty{CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:H/I:L/A:N}`

This will output the following CVSS score: 4.8

`\cvssLevel`

`\cvssLevel {<CVSS string>}`

This macro will output the CVSS level from an `{<input vector>}` (without CVSS3.1/), for example Info.

`\cvssLevel{CVSS:3.1/AV:A/AC:H/PR:H/UI:R/S:U/C:H/I:L/A:N}`

This will output the following CVSS level: Medium

`\cvssLevelpretty`

`\cvssLevelpretty {<CVSS string>}`

This macro will output the **colored** CVSS level from an `{<input vector>}` (without CVSS3.1/).

`\cvssLevelpretty{CVSS:3.1/AV:A/AC:H/PR:H/UI:R/S:U/C:L/I:L/A:N}`

This will output the following CVSS level: Low

`\cvssTag`

`\cvssTag {<CVSS string>}`

This macro will output a colored tag with the CVSS level inside, from an `{<input vector>}` (without CVSS3.1/).

`\cvssTag{CVSS:3.1/AV:A/AC:H/PR:H/UI:R/S:U/C:N/I:N/A:N}`

This will output the following CVSS level: None

`\cvssPrint`

`\cvssPrint {<CVSS string>}`

This macro will print all details of a CVSS string: colored level, score, and hyperlink to FIRST calculator, from an `{<input vector>}` (without CVSS3.1/).

`\cvssPrint{CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H}`

This will output the following CVSS level:

Critical 10 [CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H](#)

3.2 Indirect Macros

`\category`

`\category {<CVSS score>}`

This macro will output the CVSS category (None, Info, Low, Medium, High or Critical) based on the input CVSS vector passed as argument `{<numerical score>}`. The mandatory argument is a floating point CVSS score, for example 5.4.

`\category{9.9}`

This will output the following scope: Critical.

`\cvssFrame`

`\cvssFrame {<CVSS score>}`

This macro will output a CVSS tag based on a CVSS **level** passed as argument. The mandatory argument must be one of the defined CVSS levels (None, Info, Low, Medium, High or Critical), for example **Info**.

`\cvssFrame{High}`

This will output the following tag: **High**.

4 Examples

4.1 Direct Form

<code>\cvssScore{CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:H/I:L/A:N}</code>	5.3
<code>\cvssLevel{CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:H/I:L/A:N}</code>	Medium
<code>\cvssLevelpretty{CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H}</code>	High
<code>\cvssTag{CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H}</code>	Critical

We can thus embed this in text lines like this:

`\textbf{\cvssLevel{CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:H/I:L/A:N}}-level`

Which will be rendered like this : *the vuln has a **Medium**-level and we can output it inline.*

4.2 Imbricated Form

<code>\cvssFrame{Low}</code>	Low
<code>\category{9.9}</code>	Critical

We can even combine them:

`\category{\cvssScore{CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:H/I:L/A:N}}`

And this outputs: Medium

`\cvssFrame{\category{\cvssScore{CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:H/I:L/A:N}}}`

And the result is: **Medium**

4.3 Test Computations

Should be 7.3: \cvssScore{CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L}

Should be 8.3: \cvssScore{CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:L}

Should be 9.9: \cvssScore{CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:H}

Should be 9.9: \cvssScore{CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L}

Should be 7.2: \cvssScore{CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N}

Should be 7.1: \cvssScore{CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:L}

Should be 5.8: \cvssScore{CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:C/C:L/I:L/A:L}

Should be 5.5: \cvssScore{CVSS:3.1/AV:A/AC:H/PR:L/UI:N/S:C/C:L/I:L/A:L}

Should be 5.1: \cvssScore{CVSS:3.1/AV:A/AC:H/PR:L/UI:R/S:C/C:L/I:L/A:L}

Should be 4.3: \cvssScore{CVSS:3.1/AV:A/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:L}

Should be 2.4: \cvssScore{CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:U/C:L/I:N/A:N}

Should be 0.0: \cvssScore{CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:N/A:N}

And the results of the computations:

Should be 7.3: 7.3

Should be 8.3: 8.3

Should be 9.9: 9.9

Should be 9.9: 9.9

Should be 7.2: 7.2

Should be 7.1: 7.1

Should be 5.8: 5.8

Should be 5.5: 5.5

Should be 5.1: 5.1

Should be 4.3: 4.3

Should be 2.4: 2.4

Should be 0.0: 0.0

5 Implementation

5.1 Initial set up

Load the essential support (expl3, tcolorbox, xstring and hyperref).

```
1 \RequirePackage{expl3}
2 \RequirePackage[skins]{tcolorbox}
3 \tcbuselibrary{xparse}
4 \RequirePackage{xstring}
5 \RequirePackage{hyperref}
```

Then, we define the thresholds:

```
6 % These are the thresholds
7 \def\scoreLow{0.1}
8 \def\scoreMed{4.0}
9 \def\scoreHigh{7.0}
10 \def\scoreCrit{9.0}
```

And finally the colors for each level (taken from the FIRST CVSS calculator website¹)

```
11 \definecolor{color@cvss@None}{RGB}{83, 170, 51}
12 \definecolor{color@cvss@Low}{RGB}{255, 203, 13}
13 \definecolor{color@cvss@Medium}{RGB}{249, 160, 9}
14 \definecolor{color@cvss@High}{RGB}{223, 61, 3}
15 \definecolor{color@cvss@Critical}{RGB}{204, 5, 0}
```

5.2 Round up function

First we defined the roundup function² according to the precision mentioned by FIRST (<https://www.first.org/cvss/specification-documentAppendixA>). .

```
16 \ExplSyntaxOn
17 %
18 \cs_new:Npn \__CVSS_roundup:n #1 {
19     \fp_eval:n { ceil(#1,1) }
20     \fp_compare:nT { ceil(#1,1)=ceil(#1,0) } { .0 }
21 }
```

5.3 Error messages

We define some error message to help with the troubleshooting

```
22 \msg_new:nnn { CVSS } { invalid-option } { Value~'#2'~invalid~for~#1~#3.}
23 \msg_new:nnn { CVSS } { invalid-structure } { CVSS~metric~#1~is~not~correct~(#2)~#3.}
24 \msg_new:nnn { CVSS } { invalid-length } { CVSS~vector~"#1"~is~badly~formatted~#2.}
25 \msg_new:nnn { CVSS } { wrong-version } { Wrong~CVSS~version~(#2)~#3.}
```

5.4 CVSS metrics parsing

Then we can define the numerical values for each of the CVSS metric (Attack Vector, Attack Complexity, ...). This is done by checking the string value of the argument, and outputting the correpsondant value. For each function, a error message is thrown if the value is not one acceptable for that metric.

¹Available at <https://www.first.org/cvss/calculator/3.1>

²This function was inspired by the following posts: <https://tex.stackexchange.com/a/615358/28926>

5.4.1 Attack Vector

The value for the Attack Vector can only be either N (None), A (Adjacent), L (Local) or P (Physical).

`__CVSS_parseAV`

```
26 \cs_new:Npn __CVSS_parseAV:n #1
27 {
28     \str_case_e:nnF {#1}
29     {
30         { N } { 0.85 } % Network
31         { A } { 0.62 } % Adjacent
32         { L } { 0.55 } % Local
33         { P } { 0.2 } % Physical
34     }
35     { \msg_error:nnxxx { CVSS } { invalid-option } { parseAV } {#1} {\msg_line_context:} }
36 }
```

(End definition for __CVSS_parseAV.)

5.4.2 Attack Complexity

The value for the Attack Complexity metric can only be either L (Low) or H (High).

`__CVSS_parseAC`

```
37 \cs_new:Npn __CVSS_parseAC:n #1
38 {
39     \str_case_e:nnF {#1}
40     {
41         { H } { 0.44 } % High
42         { L } { 0.77 } % Low
43     }
44     { \msg_error:nnxxx { CVSS } { invalid-option } { parseAC } {#1} {\msg_line_context:} }
45 }
46 }
```

(End definition for __CVSS_parseAC.)

5.4.3 Privileges Required

The value for the Privileged Required metric can only be either N (None), L (Low) or H (High). However since the computation is different whether the Scope is changed or not, we've defined 2 functions.

3 Internal macros are thus used, one per choice (Scope unchanged and Scope change), plus the function to choose which one to take into account.

`__CVSS_parsePRScopeUnchanged`

```
47 \cs_new:Npn __CVSS_parsePRScopeUnchanged:n #1
48 {
49     \str_case_e:nnF {#1}
50     {
51         { N } { 0.85 } % None
52         { L } { 0.62 } % Low
53         { H } { 0.27 } % High
```



```

54     }
55     { \msg_error:nnxxx { CVSS } { invalid-option } { parsePRScopeUnchanged } {#1} {\msg_line_
56 }

```

(End definition for _CVSS_parsePRScopeUnchanged.)

_CVSS_parsePRScopeChanged

```

57 \cs_new:Npn \_CVSS_parsePRScopeChanged:n #1
58 {
59     \str_case_e:nnF {#1}
60     {
61         { N } { 0.85 } % None
62         { L } { 0.68 } % Low
63         { H } { 0.50 } % High
64     }
65     { \msg_error:nnxxx { CVSS } { invalid-option } { parsePRScopeChanged } {#1} {\msg_line_c
66 }

```

(End definition for _CVSS_parsePRScopeChanged.)

_CVSS_parsePR

```

67 \cs_new:Npn \_CVSS_parsePR:nn #1#2
68 {
69     % #1 Privilege Required
70     % #2 Scope
71     \str_case_e:nnF {#2}
72     {
73         { U } { \exp_args:Ne \_CVSS_parsePRScopeUnchanged:n {#1} }
74         { C } { \exp_args:Ne \_CVSS_parsePRScopeChanged:n {#1} }
75     }
76     { \msg_error:nnxxx { CVSS } { invalid-option } { parsePR } {#1} {\msg_line_context:} }
77 }

```

(End definition for _CVSS_parsePR.)

5.4.4 User Interaction

The value for the User Interaction metric can only be either N (None) or R (Required).

_CVSS_parseUI

```

78 \cs_new:Npn \_CVSS_parseUI:n #1
79 {
80     \str_case_e:nnF {#1}
81     {
82         { N } { 0.85 } % None
83         { R } { 0.62 } % Required
84     }
85     { \msg_error:nnxxx { CVSS } { invalid-option } { parseUI } {#1} {\msg_line_context:} }
86 }
87

```

(End definition for _CVSS_parseUI.)

5.4.5 Confidentiality, Integrity and Availability

The value for the Confidentiality, Integrity or Availability metrics can only be either N (None), L (Low) or H (High). Since the values are the same for the 3 metrics, we've grouped them together.

`__CVSS_parseCIA`

```

88
89 \cs_new:Npn \__CVSS_parseCIA:n #1
90 {
91     \str_case:e:nnF {#1}
92     {
93         { H } { 0.56 }
94         { L } { 0.22 }
95         { N } { 0.00 }
96     }
97     { \msg_error:nnxxx { CVSS } { invalid-option } { parseCIA } {#1} {\msg_line_context:} }
98 }

```

(End definition for `__CVSS_parseCIA`.)

5.5 CVSS computation

5.5.1 Impact Sub Score (ISS)

The value for the Impact Sub-Score (ISS) is computed from the Confidentiality, Availability and Integrity values, as follows

$$ISS = 1 - \left[(1 - \text{Confidentiality}) \times (1 - \text{Integrity}) \times (1 - \text{Availability}) \right] \quad (1)$$

This equation is then translated into \TeX code :

`__CVSS_calcISS`

```

99 \cs_new:Npn \__CVSS_calcISS:nnn #1#2#3
100 {
101     % #1 Confidentiality Impact %High H, Low L, None N
102     % #2 Integrity Impact %High H, Low L, None N
103     % #3 Availability Impact %High H, Low L, None N
104     1 - ( (1 - (\__CVSS_parseCIA:n {#1})) * (1 - (\__CVSS_parseCIA:n {#2})) * (1 - (\__CVSS_parseCIA:n {#3})) )
105 }

```

(End definition for `__CVSS_calcISS`.)

5.5.2 Impact

The calculations for the impact depends whether the scope is changed or not, and will be computed differently:

$$\text{Impact} \rightarrow \begin{cases} \text{Scope Unchanged} & 6.42 \times ISS \\ \text{Scope Changed} & 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15} \end{cases} \quad (2)$$

This gives the following implementation:

_CVSS_calcImpact

```

106 \cs_new:Npn \_CVSS_calcImpact:nn #1#2
107 {
108     % #1 = Scope
109     % #2 = ISS
110     % Scope Unchanged 6.42 × ISS
111     % Scope Changed 7.52 × [ISS-0.029] - 3.25 × [ISS-0.02]15
112     \str_case:e:nnF {#1}
113     {
114         { U } { \fp_eval:n { 6.42 * (#2) } } } % Scope UNCHANGED
115         { C } { \fp_eval:n { 7.52 * ( (#2) - 0.029 ) - 3.25 * ( (#2) - 0.02 )15 } } % Scope CHANGED
116     }
117     { \msg_error:nnxxx { CVSS } { invalid-option } { calcISC } {#1} {\msg_line_context:} }
118 }%

```

(End definition for _CVSS_calcImpact.)

5.5.3 Exploitability

The equation to compute the exploitability is the following:

$$8.22 \times \text{AttackVector} \times \text{AttackComplexity} \times \text{PrivilegesRequired} \times \text{UserInteraction} \quad (3)$$

This gives the following implementation:

_CVSS_calcExploitability

```

119 \cs_new:Npn \_CVSS_calcExploitability:nnnnn #1#2#3#4#5
120 {
121     % #1 Attack Vector
122     % #2 Attack Complexity
123     % #3 Privileges Required
124     % #4 User Interaction
125     % #5 Scope
126     % 8.22 × AttackVector × AttackComplexity × PrivilegeRequired × UserInteraction
127     8.22 * (\_CVSS_parseAV:n {#1}) * (\_CVSS_parseAC:n {#2}) * (\_CVSS_parsePR:nn {#3}{#4}) * (\_CVSS_parseUI:nn {#4}{#5})
128 }

```

(End definition for _CVSS_calcExploitability.)

5.5.4 CVSS Base Score

Now that all the pre-requisites are calculated, we can compute the CVSS base score as follows:

$$\text{Base Score} = \begin{cases} 0 & \text{if Impact} \geq 0 \\ \text{Roundup}\left(\min\left[\left(\text{Impact} + \text{Exploitability}\right), 10\right]\right) & \text{if Scope is Unchanged} \\ \text{Roundup}\left(\min\left[1.08 \times \left(\text{Impact} + \text{Exploitability}\right), 10\right]\right) & \text{if Scope is changed} \end{cases} \quad (4)$$

This gives the following implementation:

`__CVSS_cvssBaseScore`

```
129 \cs_new:Npn \__CVSS_cvssBaseScore:nnnnnnnn #1#2#3#4#5#6#7#8 {
130     % #1 Attack Vector %Network N, Adjacent A, Local L, Physical P
131     % #2 Attack Complexity %Low L, High H
132     % #3 Privileges Required %None N, Low L, High H
133     % #4 User Interaction %None N, Required R
134     % #5 Scope %Unchanged U, Changed C
135     % #6 Confidentiality Impact %High H, Low L, None N
136     % #7 Integrity Impact %High H, Low L, None N
137     % #8 Availability Impact %High H, Low L, None N
138     %
139     \fp_compare:nTF { \exp_args:Ne \__CVSS_calcImpact:nn {#5}{\exp_args:Ne \__CVSS_calcISS:n
140     % IF ISC <=0
141     {
142         % ISC <=0
143         0.0
144     }{
145         % ISC > 0
146         \str_case_e:nnF {#5}
147         {
148             { U } { % SCOPE UNCHANGED
149                 \fp_eval:n { \__CVSS_roundup:n { min( (\__CVSS_calcImpact:nn {#5}{\__CVSS
150             }
151             { C } { % SCOPE CHANGED
152                 \fp_eval:n { \__CVSS_roundup:n { min( (1.08 * (\__CVSS_calcImpact:nn {#5}{\__CVSS
153             }
154         }
155         { \msg_error:nnxxx { CVSS } { invalid-option } { parseScope } {#1} {\msg_line_cont
156     }%
157 }
```

(End definition for `__CVSS_cvssBaseScore`.)

5.5.5 CVSS Base Score

Now we can use a macro to check the validity of the CVSS string and **finally** call `__CVSS_cvssBaseScore` internally. This is the most important macro of this whole package, and is expandable.

`\cvssScore`

```
158 \NewExpandableDocumentCommand \cvssScore { m }{%
159     % Check that there are 44 chars
160     \int_compare:nNnTF { \str_count_ignore_spaces:n {#1} } = {44}{%{
161         \msg_error:nnxx{CVSS}{invalid-length}{#1}{\msg_line_context:}
162     }
163     % Check CVSS: value
164     \str_if_eq:eeTF {\str_range:nnn {#1} {1} {5}} {CVSS:}
165     {} {
166         \msg_error:nnxxx{CVSS}{invalid-structure}{AV}{\str_range:nnn {#1} {1} {5}}{\msg_line_
167     }
168     % Check 3.1 value
169     \str_if_eq:eeTF {\str_range:nnn {#1} {6} {8}} {3.1}
170     {} {
171         \msg_error:nnxxx{CVSS}{wrong-version}{3.1}{\str_range:nnn {#1} {6} {8}}{\msg_line_co
```

```

172 }
173 % Check 3.1 value
174 \str_if_eq:eeTF {\str_range:nnn {#1} {9} {9}} {/}
175 {} {
176     \msg_error:nnxxx{CVSS}{wrong-version}{/}{\str_range:nnn {#1} {9} {9}}{\msg_line_conten
177 }
178 % Check AV value
179 \str_if_eq:eeTF {\str_range:nnn {#1} {10} {12}} {AV:}
180 {} {
181     \msg_error:nnxxx{CVSS}{invalid-structure}{AV}{\str_range:nnn {#1} {10} {12}}{\msg_line
182 }
183 % Check AC value
184 \str_if_eq:eeTF {\str_range:nnn {#1} {14} {17}} {/AC:}
185 {} {
186     \msg_error:nnxxx{CVSS}{invalid-structure}{AC}{\str_range:nnn {#1} {14} {17}}{\msg_line
187 }
188
189
190 % Check PR value
191 \str_if_eq:eeTF {\str_range:nnn {#1} {19} {22}} {/PR:}
192 {} {
193     \msg_error:nnxxx{CVSS}{invalid-structure}{PR}{\str_range:nnn {#1} {19} {22}}{\msg_line
194 }
195
196 % Check UI value
197 \str_if_eq:eeTF {\str_range:nnn {#1} {24} {27}} {/UI:}
198 {} {
199     \msg_error:nnxxx{CVSS}{invalid-structure}{UI}{\str_range:nnn {#1} {24} {27}}{\msg_line
200 }
201
202 % Check S value
203 \str_if_eq:eeTF {\str_range:nnn {#1} {29} {31}} {/S:}
204 {} {
205     \msg_error:nnxxx{CVSS}{invalid-structure}{S}{\str_range:nnn {#1} {29} {31}}{\msg_line
206 }
207
208 % Check I value
209 \str_if_eq:eeTF {\str_range:nnn {#1} {33} {35}} {/C:}
210 {} {
211     \msg_error:nnxxx{CVSS}{invalid-structure}{C}{\str_range:nnn {#1} {33} {35}}{\msg_line
212 }
213
214 % Check I value
215 \str_if_eq:eeTF {\str_range:nnn {#1} {37} {39}} {/I:}
216 {} {
217     \msg_error:nnxxx{CVSS}{invalid-structure}{I}{\str_range:nnn {#1} {37} {39}}{\msg_line
218 }
219
220 % Check A value
221 \str_if_eq:eeTF {\str_range:nnn {#1} {41} {43}} {/A:}
222 {} {
223     \msg_error:nnxxx{CVSS}{invalid-structure}{A}{\str_range:nnn {#1} {41} {43}}{\msg_line
224 }
225

```

```

226 \exp_args:Ne \__CVSS_cvssBaseScore:nnnnnnnn
227 { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 13 } }
228 { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 18 } }
229 { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 23 } }
230 { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 28 } }
231 { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 32 } }
232 { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 36 } }
233 { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 40 } }
234 { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 44 } }
235 }%
236 \ExplSyntaxOff

```

(End definition for `\cvssScore`. This function is documented on page 3.)

5.6 CVSS levels

Since we can compute the numerical score of a given CVSS string, we can now get the classification of a CVSS vector using the FIRST terminology :

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

Then we can build our switch case to assign a level to the numerical CVSS score

\category This macro will output a CVSS level based on the numerical CVSS score.

```

237 \ExplSyntaxOn
238 \NewExpandableDocumentCommand \category { m }{%
239   \fp_compare:nNnTF {#1}<{\scoreLow}{None}
240   {
241     \fp_compare:nNnTF{#1}<{\scoreMed}{Low}
242     {
243       \fp_compare:nNnTF{#1}<{\scoreHigh}{Medium}
244       {
245         \fp_compare:nNnTF{#1}<{\scoreCrit}{High}
246         {Critical}
247       }%
248     }%
249   }%
250 }%
251 \ExplSyntaxOff

```

We can even have a colored version of the score.

This macro will output the **colored** CVSS level based on the CVSS vector.

\cvssScorepretty

```

252 \newcommand{\cvssScorepretty}[1]{%
253   \def\CVSScategory{\category{\cvssScore{#1}}}%
254   \textcolor{color@cvss@\CVSScategory}{\cvssScore{#1}}%
255 }%

```

(End definition for `\category` and `\cvssScorepretty`. These functions are documented on page 4.)
 We have also built a macro that will output the CVSS level based on the CVSS string, that combines `\cvssScore` and `\category`:

`\cvssLevel` This macro will output a CVSS level based on the numerical CVSS score.

```
256 \newcommand{\cvssLevel}[1]{%
257     \def\CVSSscore{\cvssScore{#1}}%
258     \category{\CVSSscore}%
259 }%
```

(End definition for `\cvssLevel`. This function is documented on page 4.)
 And we can even have a colored version of this level.

`\cvssLevelpretty` This macro will output the **colored** CVSS level based on the numerical CVSS score.

```
260 \newcommand{\cvssLevelpretty}[1]{%
261     \def\CVSScategory{\category{\cvssScore{#1}}}%
262     \textcolor{color@cvss@\CVSScategory}{\CVSScategory}%
263 }%
```

(End definition for `\cvssLevelpretty`. This function is documented on page 4.)

5.7 Fancy prints

5.7.1 Framed CVSS Level

For nice display of the CVSS score we created also tags, that can be used to highlight the CVSS score.

`\cvssFrame` First, we define `cvssFrame`, a type of `tcolorbox` we are going to use:

```
264 \DeclareTotalTCBox{\cvssFrame}{m}{%
265     enhanced,nobeforeafter,
266     tcbox raise base,
267     boxrule=0.4pt,
268     top=0mm,bottom=0mm,right=1mm,left=1mm,
269     arc=1pt,
270     boxsep=2pt,
271     colframe=color@cvss@#1,
272     colback=tcbcolframe,
273     coltext=black,
274 }{#1}%
275
276 \MakeRobust\cvssFrame
```

(End definition for `\cvssFrame`. This function is documented on page 5.)
 Then we can call this box in conjunction with `cvssScore`.

`\cvssTag` This macro will output the **colored** CVSS level based on the numerical CVSS score.

```
277 \newcommand{\cvssTag}[1]{%
278     \def\CVSSscore{\cvssScore{#1}}%
279     \cvssFrame{\category{\CVSSscore}}%
280 }%
```

(End definition for `\cvssTag`. This function is documented on page 4.)

5.7.2 Full CVSS display

We can even have a nice all-in display of the category, the score and a hyperlink to the FIRST calculator using a combination of all the functions we've defined:

\cvssPrint v1.12022/11/30Full CVSS vector as input is now supported This macro will output the **colored** CVSS level based on the numerical CVSS score.

```
281 \newcommand{\cvssPrint}[1]{%  
282     \def\CVSSscore{\cvssScore{#1}}  
283     \cvssFrame{\category{\CVSSscore}} \quad \CVSSscore \quad %  
284     \href{https://www.first.org/cvss/calculator/3.1\###1}{#1}  
285 }%
```

(End definition for \cvssPrint. This function is documented on page 4.)

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
\#	284
C	
calcExploitability internal commands:	
_CVSS_calcExploitability	119
calcImpact internal commands:	
_CVSS_calcImpact	106
calcISS internal commands:	
_CVSS_calcISS	99
\category	4, 237, 258, 261, 279, 283
cs commands:	
\cs_new:Npn	18, 26, 37, 47, 57, 67, 78, 89, 99, 106, 119, 129
CVSS internal commands:	
_CVSS_calcExploitability:nnnnn	119, 149, 152
_CVSS_calcImpact:nn	106, 139, 149, 152
_CVSS_calcISS:nnn	99, 139, 149, 152
_CVSS_cvssBaseScore:nnnnnnnn	129, 226
_CVSS_parseAC:n	37, 127
_CVSS_parseAV:n	26, 127
_CVSS_parseCIA:n	89, 104
_CVSS_parsePR:nn	67, 127
_CVSS_parsePRScopeChanged:n	57, 74
_CVSS_parsePRScopeUnchanged:n	47, 73
_CVSS_parseUI:n	78, 127
_CVSS_roundup:n	18, 149, 152
cvssBaseScore internal commands:	
_CVSS_cvssBaseScore	129
\CVSScategory	253, 254, 261, 262
\cvssFrame	5, 264, 279, 283
\cvssLevel	4, 256
\cvssLevelpretty	4, 260
\cvssPrint	4, 281
\CVSSscore	257, 258, 278, 279, 282, 283
\cvssScore	3, 158, 253, 254, 257, 261, 278, 282
\cvssScorepretty	3, 252
\cvssTag	4, 277
D	
\DeclareTotalTCBox	264
\def	7, 8, 9, 10, 253, 257, 261, 278, 282
\definecolor	11, 12, 13, 14, 15
E	
exp commands:	
\exp_args:Ne	73, 74, 139, 226
\ExplSyntaxOff	236, 251
\ExplSyntaxOn	16, 237
F	
fp commands:	
\fp_compare:nNnTF	239, 241, 243, 245
\fp_compare:nTF	20, 139
\fp_eval:n	19, 114, 115, 149, 152
H	
\href	284
I	
int commands:	
\int_compare:nNnTF	160
M	
\MakeRobust	276
msg commands:	
\msg_error:nnnn	161
\msg_error:nnnnn	35, 45, 55, 65, 76, 85, 97, 117, 155, 166, 171, 176, 181, 186, 193, 199, 205, 211, 217, 223
\msg_line_context:	35, 45, 55, 65, 76, 85, 97, 117, 155, 161, 166, 171, 176, 181, 186, 193, 199, 205, 211, 217, 223
\msg_new:nnn	22, 23, 24, 25
N	
\newcommand	252, 256, 260, 277, 281
\NewExpandableDocumentCommand	158, 238
P	
parseAC internal commands:	
_CVSS_parseAC	37
parseAV internal commands:	
_CVSS_parseAV	26
parseCIA internal commands:	
_CVSS_parseCIA	88
parsePR internal commands:	
_CVSS_parsePR	67
parsePRScopeChanged internal commands:	
_CVSS_parsePRScopeChanged	57
parsePRScopeUnchanged internal commands:	
_CVSS_parsePRScopeUnchanged	47

parseUI internal commands:		
<code>__CVSS_parseUI</code>	78	
Q		
<code>\quad</code>	283	
R		
<code>\RequirePackage</code>	1, 2, 4, 5	
S		
<code>\scoreCrit</code>	10, 245	
<code>\scoreHigh</code>	9, 243	
<code>\scoreLow</code>	7, 239	
<code>\scoreMed</code>	8, 241	
str commands:		
<code>\str_case_e:nnTF</code>	28, 39, 49, 59, 71, 80, 91, 112, 146	
<code>\str_count_ignore_spaces:n</code>	160	
<code>\str_if_eq:nnTF</code> ...	164, 169, 174, 179, 184, 191, 197, 203, 209, 215, 221	
<code>\str_item_ignore_spaces:nn</code>	227, 228, 229, 230, 231, 232, 233, 234	
<code>\str_range:nnn</code>	164, 166, 169, 171, 174, 176, 179, 181, 184, 186, 191, 193, 197, 199, 203, 205, 209, 211, 215, 217, 221, 223	
<code>\str_use:N</code>	227, 228, 229, 230, 231, 232, 233, 234	
T		
<code>\tcbuselibrary</code>	3	
<code>\textcolor</code>	254, 262	