The cvss package*

Pierre VIVEGNIS†

Released 2022/12/02

Contents

*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 Environental 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** (fron 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

`\cvssScore` `\cvssScore {⟨CVSS string⟩}`

This is the main macro of this package, responsible for computing the base CVSS 3.1 score of an `{⟨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`

`\cvssScorepretty` `\cvssScorepretty {⟨CVSS string⟩}`

This macro will print a **colored** base CVSS 3.1 score of an `{⟨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

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

`\cvssFrame{Low}`	Low
`\category{9.9}`	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).

```
\RequirePackage{expl3}
\RequirePackage[skins]{tcolorbox}
\tcbuselibrary{xparse}
\RequirePackage{xstring}
\RequirePackage{hyperref}
```

Then, we define the thresholds:

```
% These are the thresholds
\def\scoreLow{0.1}
\def\scoreMed{4.0}
\def\scoreHigh{7.0}
\def\scoreCrit{9.0}
```

And finally the colors for each level (taken from the FIRST CVSS calulator website[1])

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

5.2 Round up function

First we defined the `roundup` function[2] according to the precision mentionned by FIRST (https://www.first.org/cvss/specification-documentAppendixA). .

```
\ExplSyntaxOn
%
\cs_new:Npn \__CVSS_roundup:n #1 {
          \fp_eval:n { ceil(#1,1) }
          \fp_compare:nT { ceil(#1,1)=ceil(#1,0) } {.0}
}
```

5.3 Error messages

We define some error message to help with the troubleshooting

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

[1] Available at https://www.first.org/cvss/calculator/3.1

[2] This 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 by either `N` (None), `A` (Adjacent), `L` (Local) or `P` (Physical).

__CVSS_parseAV

```
\cs_new:Npn \__CVSS_parseAV:n #1
{
     \str_case_e:nnF {#1}
     {
          { N } { 0.85 } % Network
          { A } { 0.62 } % Adjacent
          { L } { 0.55 } % Local
          { P } { 0.2 }  % Physical
     }
     { \msg_error:nnxxx { CVSS } { invalid-option } { parseAV } {#1} {\msg_line_context:} }
}
```

(*End definition for* `\__CVSS_parseAV`.)

5.4.2 Attack Complexity

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

__CVSS_parseAC

```
\cs_new:Npn \__CVSS_parseAC:n #1
{
     \str_case_e:nnF {#1}
     {
          { H } { 0.44 }     % High
          { L } { 0.77 }     % Low

     }
     { \msg_error:nnxxx { CVSS } { invalid-option } { parseAC } {#1} {\msg_line_context:} }
}
```

(*End definition for* `\__CVSS_parseAC`.)

5.4.3 Privileges Required

The value for the Privilged Required metric can only by either `N` (None), `L` (Low) or `H` (High). Hoever since the computation is different wheter 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

```
\cs_new:Npn \__CVSS_parsePRScopeUnchanged:n #1
{
     \str_case_e:nnF {#1}
     {
          { N } { 0.85 } % None
          { L } { 0.62 } % Low
          { H } { 0.27 } % High
```

```
    }
    { \msg_error:nnxxx { CVSS } { invalid-option } { parsePRScopeUnchanged } {#1} {\msg_line
}
```

(*End definition for* `\__CVSS_parsePRScopeUnchanged`.)

`\__CVSS_parsePRScopeChanged`

```
\cs_new:Npn \__CVSS_parsePRScopeChanged:n #1
{
    \str_case_e:nnF {#1}
    {
        { N } { 0.85 } % None
        { L } { 0.68 } % Low
        { H } { 0.50 } % High
    }
    { \msg_error:nnxxx { CVSS } { invalid-option } { parsePRScopeChanged } {#1} {\msg_line_c
}
```

(*End definition for* `\__CVSS_parsePRScopeChanged`.)

`\__CVSS_parsePR`

```
\cs_new:Npn \__CVSS_parsePR:nn #1#2
{
    % #1 Privilege Required
    % #2 Scope
    \str_case_e:nnF {#2}
    {
        { U } { \exp_args:Ne \__CVSS_parsePRScopeUnchanged:n {#1} }
        { C } { \exp_args:Ne \__CVSS_parsePRScopeChanged:n {#1} }
    }
    { \msg_error:nnxxx { CVSS } { invalid-option } { parsePR } {#1} {\msg_line_context:} }
}
```

(*End definition for* `\__CVSS_parsePR`.)

5.4.4 User Interaction

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

`\__CVSS_parseUI`

```
\cs_new:Npn \__CVSS_parseUI:n #1
{
    \str_case_e:nnF {#1}
    {
        { N } { 0.85 } % None
        { R } { 0.62 } % Required
    }
    { \msg_error:nnxxx { CVSS } { invalid-option } { parseUI } {#1} {\msg_line_context:} }
}

```

(*End definition for* `\__CVSS_parseUI`.)

5.4.5 Confidentiality, Integrity and Availability

The value for the Confidentiality, Integrity or Availability metrics can only by 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

```
\cs_new:Npn \__CVSS_parseCIA:n #1
{
     \str_case_e:nnF {#1}
     {
          { H } { 0.56 }
          { L } { 0.22 }
          { N } { 0.00 }
     }
     { \msg_error:nnxxx { CVSS } { invalid-option } { parseCIA } {#1} {\msg_line_context:} }
}
```

(*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 conmputed from the Confidentiality, Availability and Integrity values, as follows

$$ISS = 1 - \Big[(1 - \text{Confidentiality}) \times (1 - \text{Integrity}) \times (1 - \text{Availability})\Big] \tag{1}$$

This equation is then translated into TEXcode :

__CVSS_calcISS

```
\cs_new:Npn \__CVSS_calcISS:nnn #1#2#3
{
     % #1 Confidentiality Impact %High H, Low L, None N
     % #2 Integrity Impact %High H, Low L, None N
     % #3 Availability Impact %High H, Low L, None N
     1 - ( (1 - (\__CVSS_parseCIA:n {#1})) * (1 - (\__CVSS_parseCIA:n {#2})) * (1 - (\__CVSS_
}
```

(*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:

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

This gives the following implementation:

__CVSS_calcImpact

```
\cs_new:Npn \__CVSS_calcImpact:nn #1#2
{
     % #1 = Scope
     % #2 = ISS
     % Scope Unchanged 6.42 × ISS
     % Scope Changed 7.52 × [ISS-0.029] - 3.25 × [ISS-0.02]15
     \str_case_e:nnF {#1}
     {
          { U } { \fp_eval:n { 6.42 * (#2) } } % Scope UNCHANGED
          { C } { \fp_eval:n { 7.52 * ( (#2) - 0.029 ) - 3.25 * ( (#2) - 0.02 )^15 } } % Scop
     }
     { \msg_error:nnxxx { CVSS } { invalid-option } { calcISC } {#1} {\msg_line_context:} }
}%
```

(*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

```
\cs_new:Npn \__CVSS_calcExploitability:nnnnn #1#2#3#4#5
{
     % #1 Attack Vector
     % #2 Attack Complexity
     % #3 Privileges Required
     % #4 User Interaction
     % #5 Scope
     % 8.22 × AttackVector × AttackComplexity × PrivilegeRequired × UserInteraction
     8.22 * (\__CVSS_parseAV:n {#1}) * (\__CVSS_parseAC:n {#2}) * (\__CVSS_parsePR:nn {#3}{#5
}
```

(*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:

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

This gives the following implementation:

__CVSS_cvssBaseScore

```
\cs_new:Npn \__CVSS_cvssBaseScore:nnnnnnnn #1#2#3#4#5#6#7#8 {
     % #1 Attack Vector %Network N, Adjacent A, Local L, Physical P
     % #2 Attack Complexity %Low L, High H
     % #3 Privileges Required %None N, Low L, High H
     % #4 User Interaction %None N, Required R
     % #5 Scope %Unchanged U, Changed C
     % #6 Confidentiality Impact %High H, Low L, None N
     % #7 Integrity Impact %High H, Low L, None N
     % #8 Availability Impact %High H, Low L, None N
     %
     \fp_compare:nTF { \exp_args:Ne \__CVSS_calcImpact:nn {#5}{\exp_args:Ne \__CVSS_calcISS:n
     % IF ISC <=0
     {
          % ISC <=0
          0.0
     }{
          % ISC > 0
          \str_case_e:nnF {#5}
          {
               { U } { % SCOPE UNCHANGED
                    \fp_eval:n { \__CVSS_roundup:n { min( ((\__CVSS_calcImpact:nn {#5}{\__CVS
               }
               { C } { % SCOPE CHANGED
                    \fp_eval:n { \__CVSS_roundup:n { min( (1.08 * ((\__CVSS_calcImpact:nn {#5
               }
          }
          { \msg_error:nnxxx { CVSS } { invalid-option } { parseScope } {#1} {\msg_line_conte
     }%
}
```

(*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

```
\NewExpandableDocumentCommand \cvssScore { m }{%
    % Check that there are 44 chars
    \int_compare:nNnTF { \str_count_ignore_spaces:n {#1} } = {44}{}{
        \msg_error:nnxx{CVSS}{invalid-length}{#1}{\msg_line_context:}
    }
    % Check CVSS: value
    \str_if_eq:eeTF {\str_range:nnn {#1} {1} {5}} {CVSS:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{AV}{\str_range:nnn {#1} {1} {5}}{\msg_line_
    }
    % Check 3.1 value
    \str_if_eq:eeTF {\str_range:nnn {#1} {6} {8}} {3.1}
    {} {
        \msg_error:nnxxx{CVSS}{wrong-version}{3.1}{\str_range:nnn {#1} {6} {8}}{\msg_line_cor
```

```
    }
    % Check 3.1 value
    \str_if_eq:eeTF {\str_range:nnn {#1} {9} {9}} {/}
    {} {
        \msg_error:nnxxx{CVSS}{wrong-version}{/}{\str_range:nnn {#1} {9} {9}}{\msg_line_conte
    }
    % Check AV value
    \str_if_eq:eeTF {\str_range:nnn {#1} {10} {12}} {AV:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{AV}{\str_range:nnn {#1} {10} {12}}{\msg_lin
    }
    % Check AC value
    \str_if_eq:eeTF {\str_range:nnn {#1} {14} {17}} {/AC:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{AC}{\str_range:nnn {#1} {14} {17}}{\msg_lin
    }


    % Check PR value
    \str_if_eq:eeTF {\str_range:nnn {#1} {19} {22}} {/PR:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{PR}{\str_range:nnn {#1} {19} {22}}{\msg_lin
    }

    % Check UI value
    \str_if_eq:eeTF {\str_range:nnn {#1} {24} {27}} {/UI:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{UI}{\str_range:nnn {#1} {24} {27}}{\msg_lin
    }

    % Check S value
    \str_if_eq:eeTF {\str_range:nnn {#1} {29} {31}} {/S:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{S}{\str_range:nnn {#1} {29} {31}}{\msg_line
    }

    % Check I value
    \str_if_eq:eeTF {\str_range:nnn {#1} {33} {35}} {/C:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{C}{\str_range:nnn {#1} {33} {35}}{\msg_line
    }

    % Check I value
    \str_if_eq:eeTF {\str_range:nnn {#1} {37} {39}} {/I:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{I}{\str_range:nnn {#1} {37} {39}}{\msg_line
    }

    % Check A value
    \str_if_eq:eeTF {\str_range:nnn {#1} {41} {43}} {/A:}
    {} {
        \msg_error:nnxxx{CVSS}{invalid-structure}{A}{\str_range:nnn {#1} {41} {43}}{\msg_line
    }

```

```
    \exp_args:Ne \__CVSS_cvssBaseScore:nnnnnnnn
    { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 13 } }
    { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 18 } }
    { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 23 } }
    { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 28 } }
    { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 32 } }
    { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 36 } }
    { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 40 } }
    { \str_use:N \str_item_ignore_spaces:nn { #1 }{ 44 } }
}%
\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.

```
\ExplSyntaxOn
\NewExpandableDocumentCommand \category { m }{%
     \fp_compare:nNnTF {#1}<{\scoreLow}{None}
     {
          \fp_compare:nNnTF{#1}<{\scoreMed}{Low}
          {
               \fp_compare:nNnTF{#1}<{\scoreHigh}{Medium}
               {
                    \fp_compare:nNnTF{#1}<{\scoreCrit}{High}
                    {Critical}
               }%
          }%
     }%
}%
\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`

```
\newcommand{\cvssScorepretty}[1]{%
     \def\CVSScategory{\category{\cvssScore{#1}}}%
     \textcolor{color@cvss@\CVSScategory}{\cvssScore{#1}}%
}%
```

(*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.

```
\newcommand{\cvssLevel}[1]{%
    \def\CVSSscore{\cvssScore{#1}}%
    \category{\CVSSscore}%
}%
```

(*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.

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

(*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:

```
\DeclareTotalTCBox{\cvssFrame}{m}{
    enhanced,nobeforeafter,
    tcbox raise base,
    boxrule=0.4pt,
    top=0mm,bottom=0mm,right=1mm,left=1mm,
    arc=1pt,
    boxsep=2pt,
    colframe=color@cvss@#1,
    colback=tcbcolframe,
    coltext=black,
}{#1}%

\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.

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

(*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 scrore 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.

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

(*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.