ADDO

ALL DAY DEVOPS

NOVEMBER 6, 2019

DJ Schleen

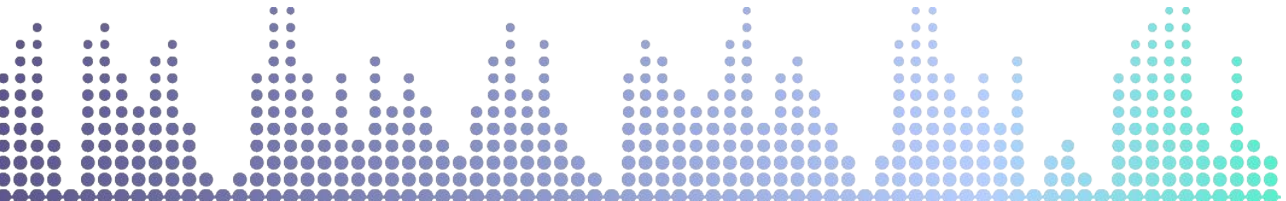
Metric vs. Imperial and why Defect Density Sucks

WARNING

This talk has absolutely nothing to do with Metric or Imperial measurements, nor traditional Defect Density calculations. It's about Security Defect Density and it's absolute uselessness for calculating application security risk. Oh... and gratuitous Deadpool references ahead. DevOps discretion advised.


FEELING VS. FACT

1 VULNERABILITY IN 10000 LINES OF CODE

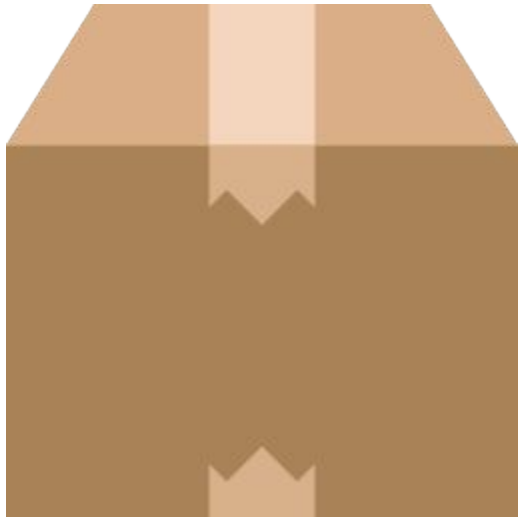


@djschleen

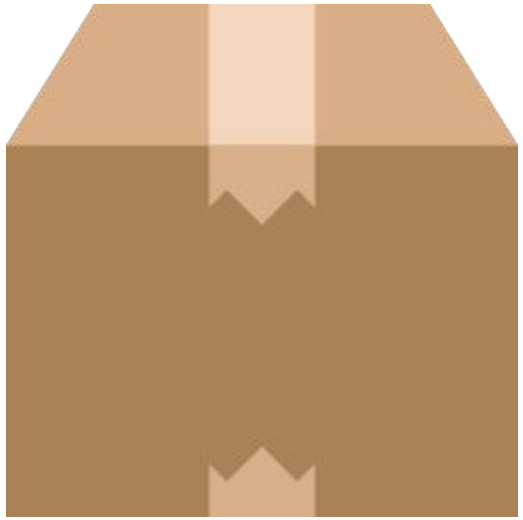
I hate being suspicious about things, but damn that gut feeling is always right.

—uckology

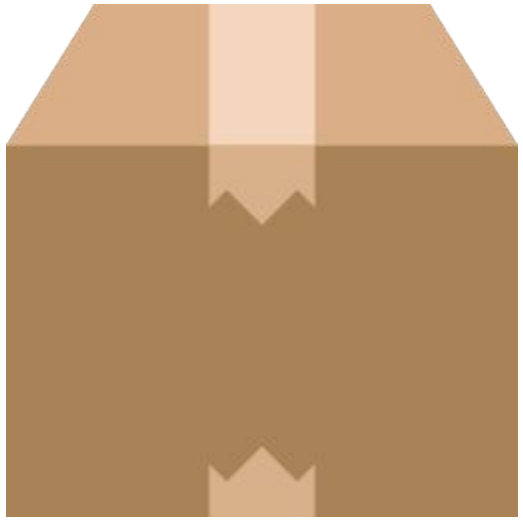
CONTEXT



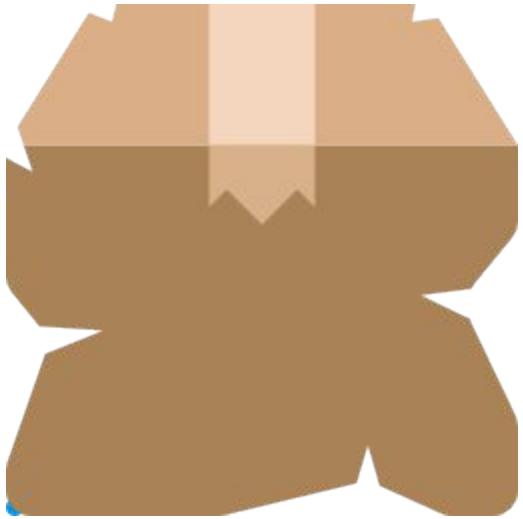
SAST



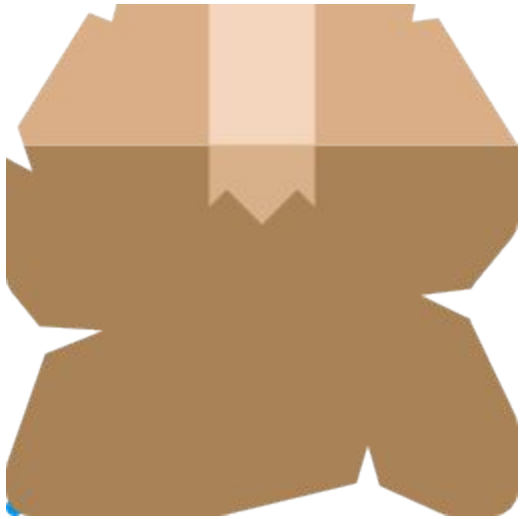
static analysis and security testing



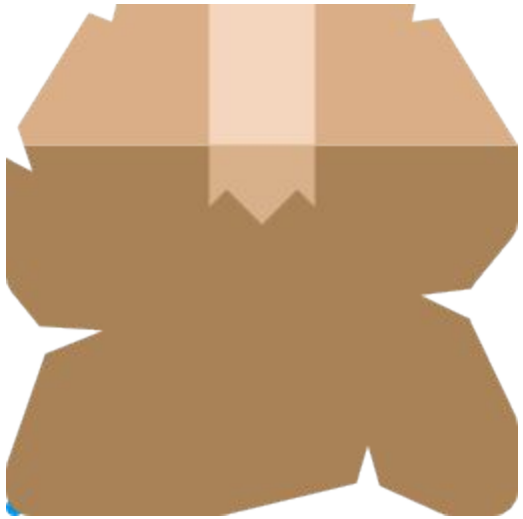
what your team built to put in the box



DAST



dynamic analysis and security testing



pounding the box with a sledgehammer

for the next three days...

CSA

container vulnerability analysis

how the box is packed

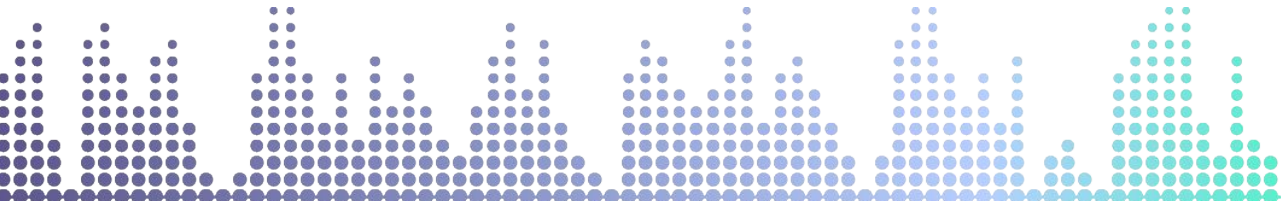
SCA

software composition analysis

packing the box with someone else's stuff


WHAT'S WRONG HERE?

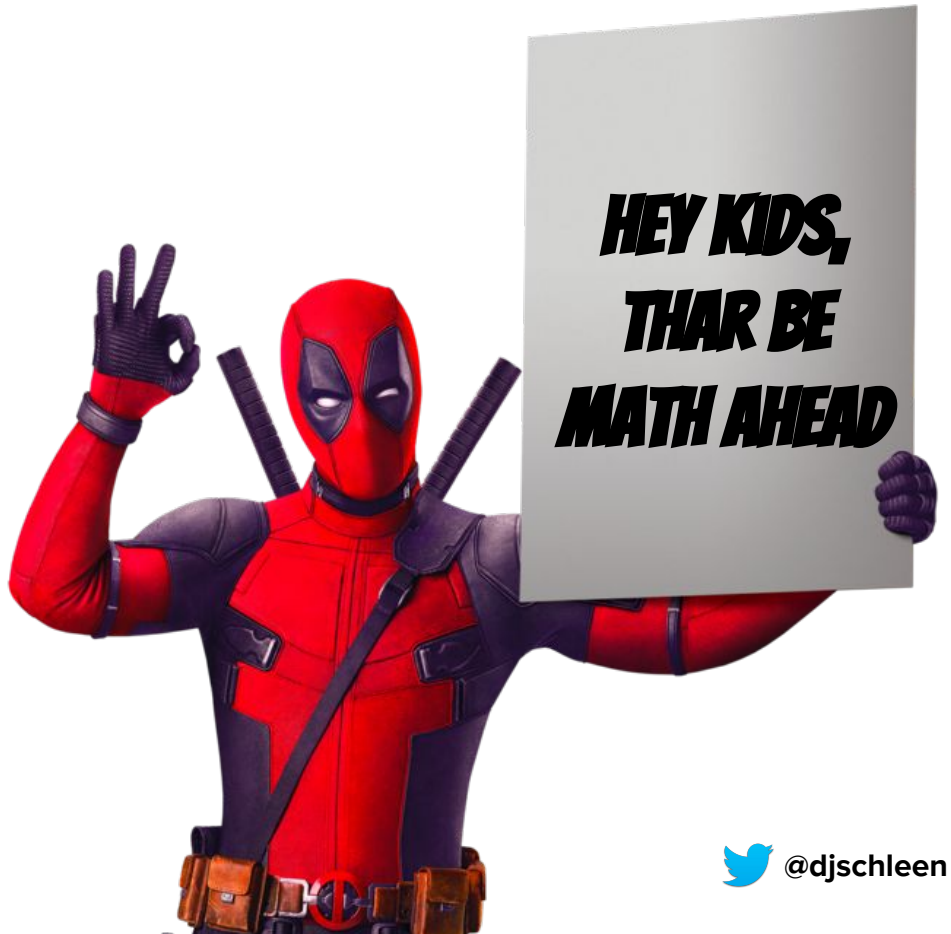
LETS DIG IN...



d Defect Density

v Vulnerabilities

l Lines of Code



$$d = v/l$$

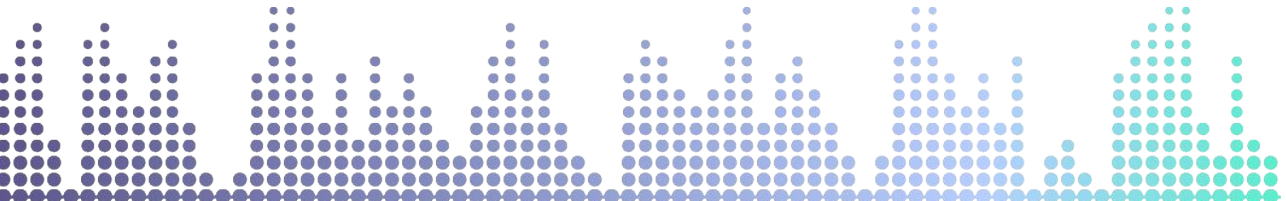
$$0.01 = 1000/100000$$

$$1\%$$

0.01 OR 1%

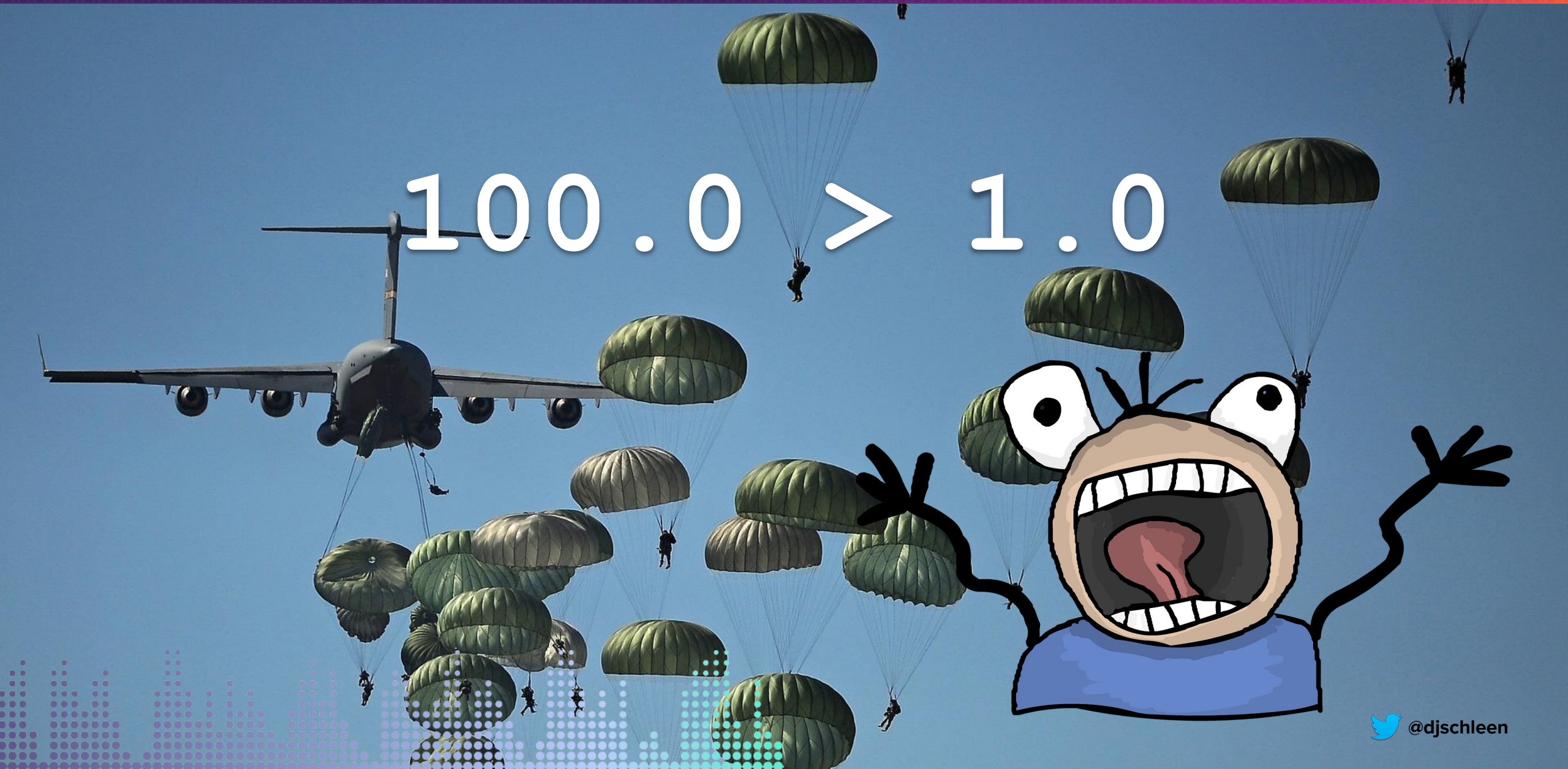
WHATEVER THIS MEANS.

1 VULNERABILITY IN 10000 LINES OF CODE



@djschleen

```
d = (v/1) * 10000
100 = (1000/100000)
      * 10000
```


100.0 > 1.0
@djschleen

```
for i in {0..20000200}
do
 echo [comment] >> [sourcefile]
done
```


0.49


HEY KIDS, THAR BE DRAGONS!

FUNNY BUSINESS!

Concatenate garbage

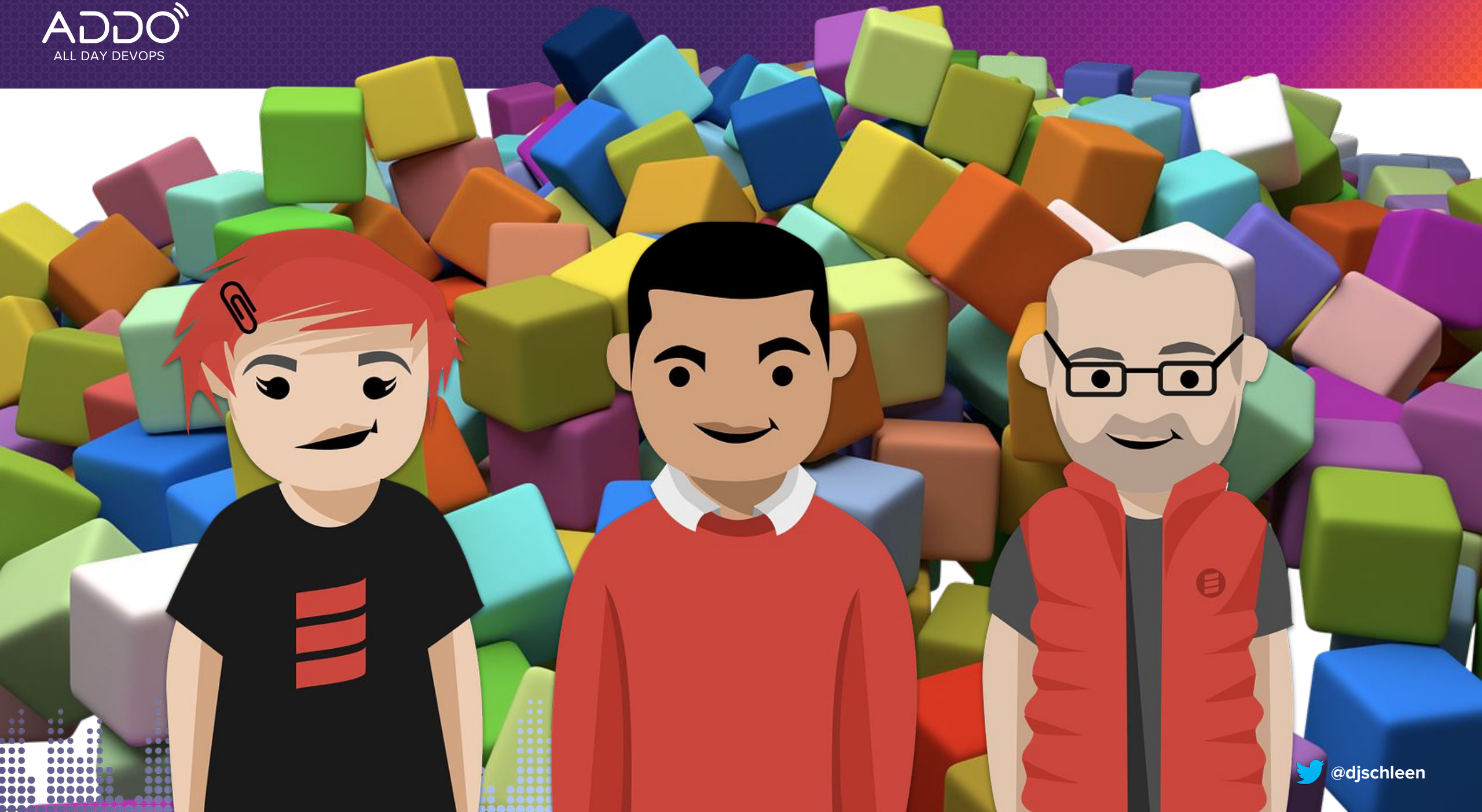
Load up on comments

Keep dead code

Include third party code

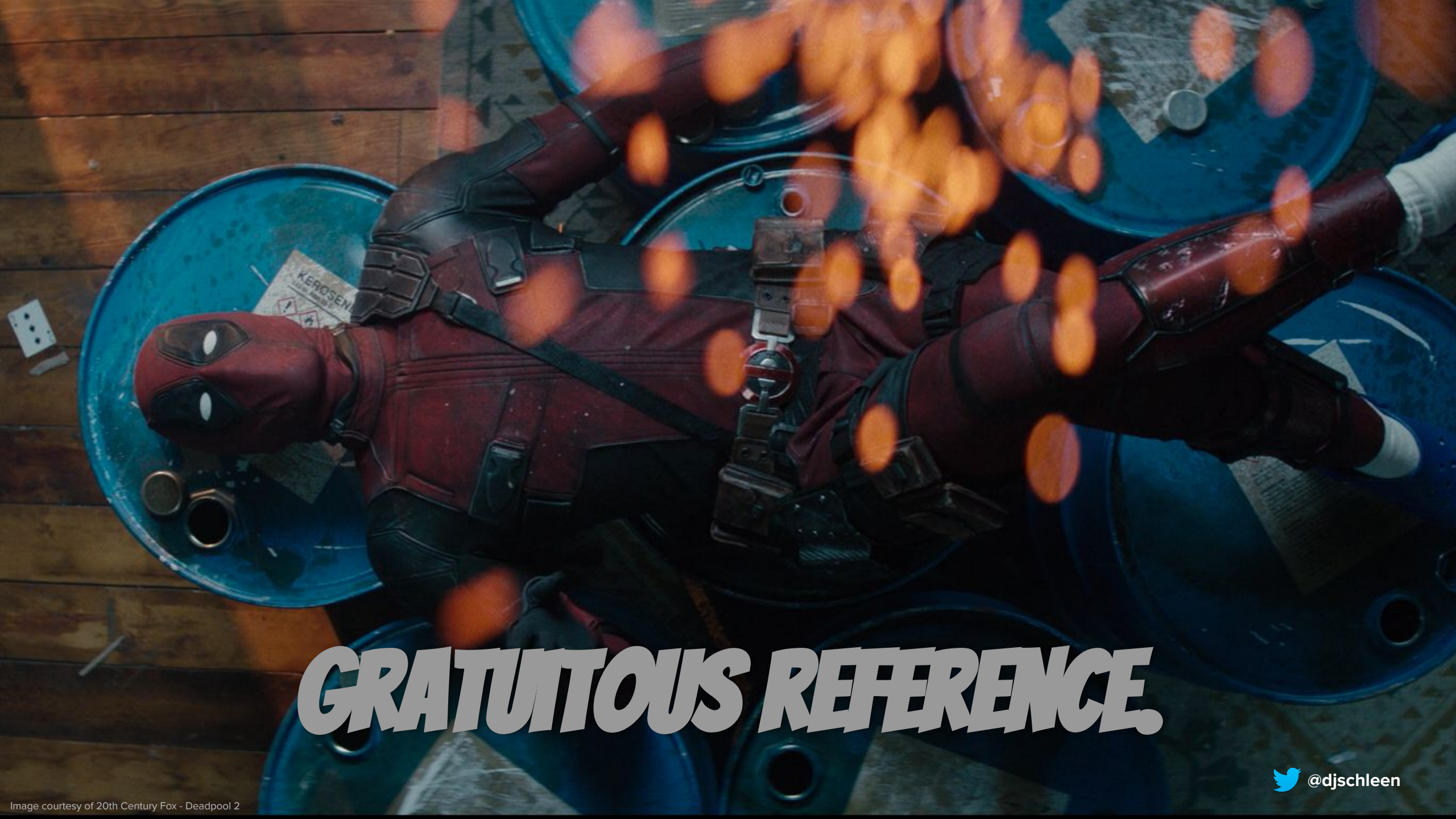
Really... only Highs from SAST? ***REALLY?***

ADDO
ALL DAY DEVOPS



@djschleen

ONLY 3%-15% OF APPLICATION CODE IS DEVELOPED BY YOUR TEAM



GRATUITOUS REFERENCE.

@djschleen

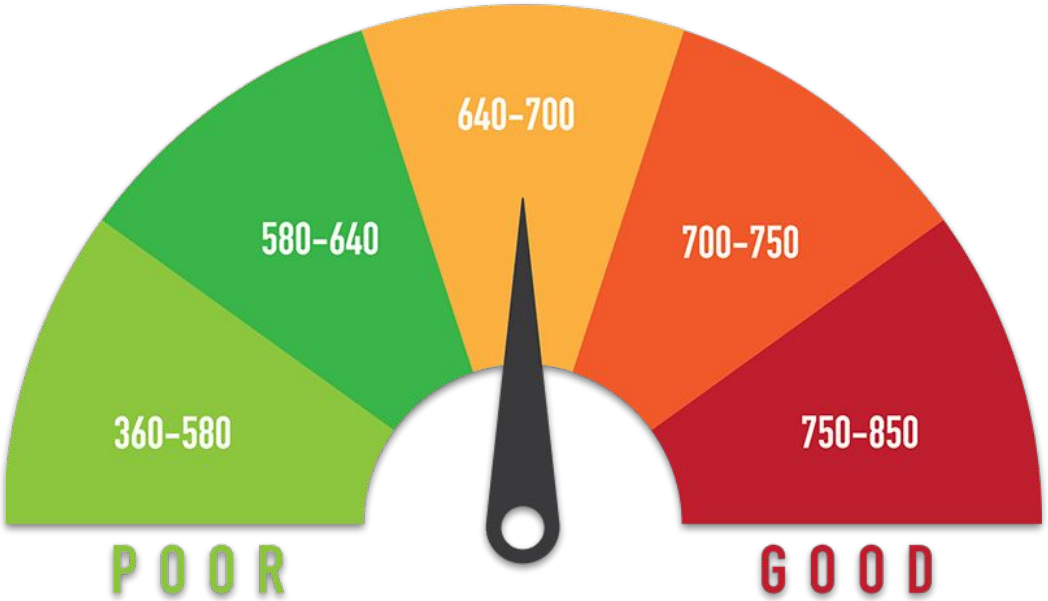
Image courtesy of 20th Century Fox - Deadpool 2


Usefull
Useless

APPLICATION SECURITY HEALTH

If your application had a credit score...

Would you give it a loan with a score of 210?



IMPLEMENTATION GOALS

Process data from any source

Discard duplicate findings

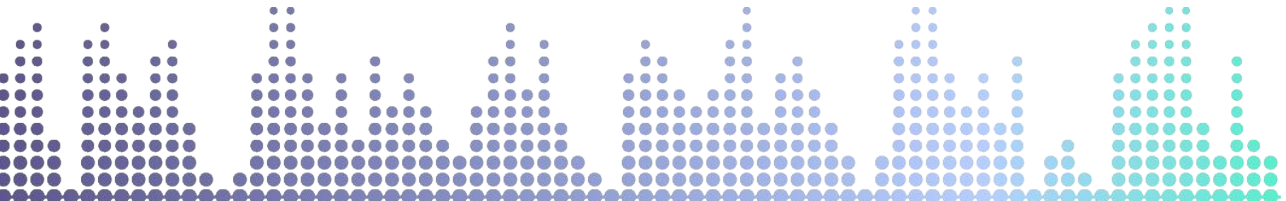
Criticals, Highs, Mediums, and Lows included

Weigh recent vulnerabilities higher

Weigh vulnerabilities by severity

Weigh vulnerabilities by ability to exploit

REMOVE DEPENDENCY ON LINES OF CODE!



```
 ➪ ./ash calculate --identifiers CVE-2019-1563,CVE-2019-1563
570
● 0 17:01   dj@spaceman:~/Code/djschleen/ash [master|Δ10?1]
 ➪ ./ash calculate --identifiers CVE-2019-1563,CVE-2019-1549
535
● 0 17:01   dj@spaceman:~/Code/djschleen/ash [master|Δ10?1]
 ➪ ./ash calculate --identifiers CVE-2019-1563,CVE-2019-1549,CVE-2019-1547
590
```

FRANCIS

DEADPOOL

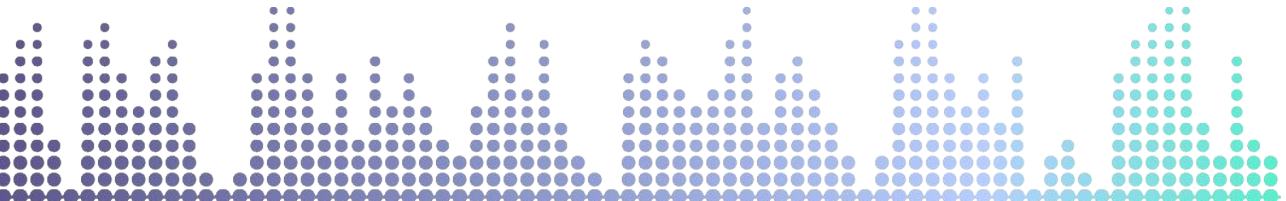
ADDO
ALL DAY DEVOPS

INSTANTLY UNDERSTANDABLE...

UNLIKE YOUR CAT

@djschleen

POC



README.md

38 components 0 1 0

Application Security Health Score (ASH)

*** NOTE: This is a working POC but hasn't been tested at a massive scale ***

I've been looking for a replacement for Security Defect Density that can provide a more precise measurement of the security health of an application. Enter the Application Security Health Score (ASH) - a calculation providing a single number similar to a credit score to describe application security risk.

If you were a bank and your application was applying for a loan, would you give it one with a credit score of 70, or a score of 810?

Contributing

Contributions are definitely encouraged!. The scoring calculation has many TODO's that would be great to implement. Create a pull request and let's get more accurate in scoring

ADDO
ALL DAY DEVOPS

djschleen/**ash** > **master** > cmd / **calculate.go** Blame

FILES SYMBOLS

- calculate.go
- completion.go
- info.go
- info_test.go
- root.go

```
26          },
27      }
28  )
29
30  func init() {
31      calculateCmd.Flags().StringSliceVarP(&cveIDList, "identifiers", "i", []string{}, "")
32      rootCmd.AddCommand(calculateCmd)
33  }
34
35  //Calculate Computes the ASH ash score
36  func Calculate(shortCves []calc.ShortCve) int64 {
37      //TODO: Weight by date
38      //TODO: Weight by impact
39      //TODO: Weight by ease of exploit
40      //TODO: Deduplicate
41
42      sumWeight, avg := 0.0, 0.0
43      for _, v := range shortCves {
44          sumWeight += v.SeverityWeight()
45          avg += v.Cvss * v.SeverityWeight()
46      }
47      avg /= sumWeight
48
49      avg *= -100
50      avg = avg + 1000
51      return int64(math.RoundToEven(avg))
```

@djschleen

ADDO
ALL DAY DEVOPS

djschleen/ash > master > cmd / calculate.go Blame

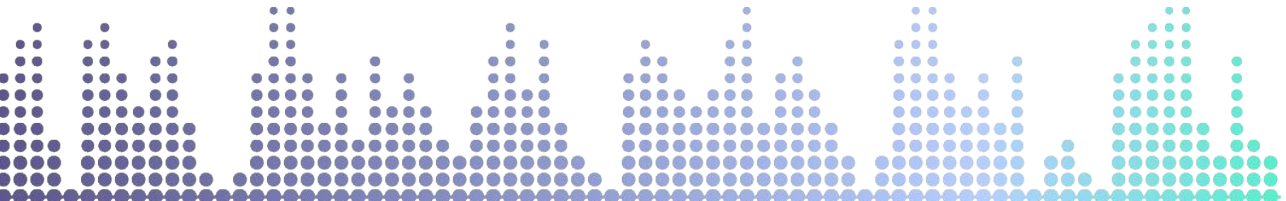
FILES SYMBOLS

- calculate.go
- completion.go
- info.go
- info_test.go
- root.go

```
26          },
27      }
28  )
29
30  func init() {
31      calculateCmd.Flags().StringSliceVarP(&cveIDList, "identifiers", "i", []string{}, "")
32      rootCmd.AddCommand(calculateCmd)
33  }
34
35  //Calculate Computes the ASH ash score
36  func Calculate(shortCves []calc.ShortCve) int64 {
37      //TODO: Weight by date
38      //TODO: Weight by impact
39      //TODO: Weight by ease of exploit
40      //TODO: Deduplicate
41
42      sumWeight, avg := 0.0, 0.0
43      for _, v := range shortCves {
44          sumWeight += v.SeverityWeight()
45          avg += v.Cvss * v.SeverityWeight()
46      }
47      avg /= sumWeight
48
49      avg *= -100
50      avg = avg + 1000
51      return int64(math.RoundToEven(avg))
```

@djschleen

OPEN SOURCE



THANK YOU AND PLEASE CONTRIBUTE!

github.com/djschleen/ash


MAXIMUM
EFFORT!!!

ADDO
ALL DAY DEVOPS

SPONSORS

Sponsorship packages for All Day DevOps are available. If your organization is interested, please contact us for details.

DIAMOND SPONSORS

sonatype

StateFarm

GOLD SPONSORS

vmware

Fannie Mae

NORTHROP GRUMMAN

COMMUNITY ADVOCATES AND VIEWING PARTY SPONSORS

XebiaLabs
Enterprise DevOps

cloudbees

Carnegie Mellon University
Software Engineering Institute

TIGERA

CONTINO

NowSecure

MEDIA SPONSORS

DZone
A DEVADA MEDIA PROPERTY

TechBeacon

DevOps.com

MediaOps

ITSP MAGAZINE

the cyberwire