BLAZE

INFORMATION SECURITY

Test Dive Conference 2019

BLAZE

Slaying bugs and **improving**
software security through fuzz testing

BLAZE

INTRO

Julio Cesar Fort

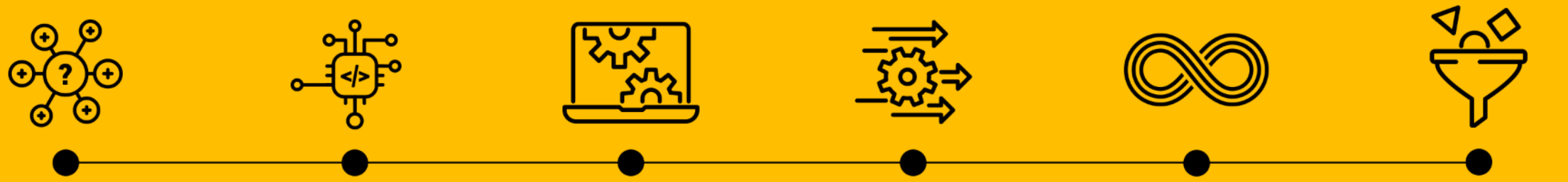
Director of Professional Services
at Blaze Information Security

INTRO

This talk aims to be a **gentle introduction to fuzzing** for software testers and how it can be used to discover issues and improve software robustness and security.

Agenda

BLAZE


Fuzz testing: introduction
Main types of fuzzing
Popular tools
Pre and post-fuzzing
DevSecOps integration
Conclusion

BLAZE

Fuzz testing: introduction

BLAZE

FUZZ TESTING:
INTRODUCTION

BRIEF INTRODUCTION TO FUZZING (1/2):

- **Send inputs automatically to a software**, and monitor its behavior
- **Find inputs that cause:** crashes, hangs, memory leaks, undefined behavior, assert();
- Early research by **professor Barton Miller** in 1988

FUZZ TESTING:
INTRODUCTION

BRIEF INTRODUCTION TO FUZZING (2/2):

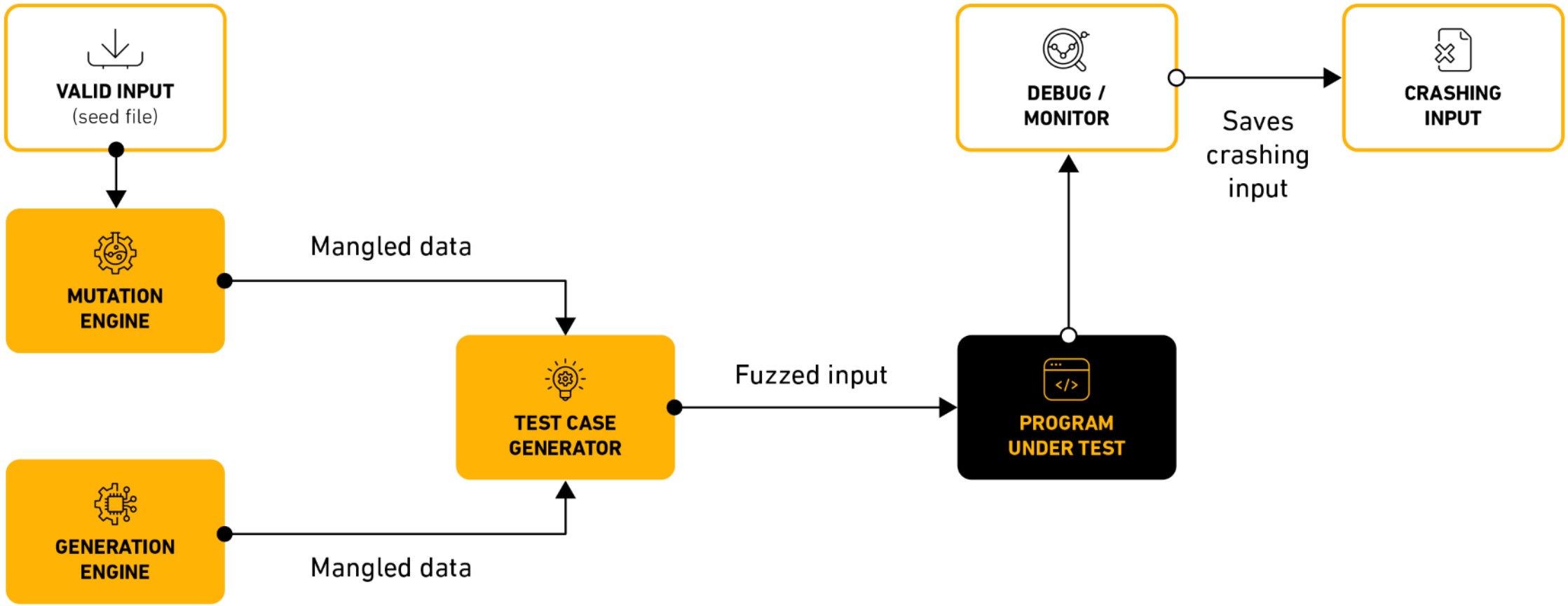
- **Works against software with or without source code**
- An **important element of** a robust **SDLC**
- **Battle-tested:** it can really discover critical bugs

FUZZ TESTING:
INTRODUCTION

FUZZING WORKS BY SENDING INVALID DATA TO A PROGRAM:

- **If data is too valid**, chances are **nothing will happen**
- **If data is too invalid**, it **may get rejected straight away** and **nothing will happen either**
- **Observes** for failures such as hangs, crashes and other unusual behavior

BASIC FUZZING CYCLE



BLAZE

Main types of fuzzing

BLAZE

MAIN TYPES OF FUZZING

MUTATION BASED:

- It requires a set of **seeds** – or **known good inputs**
- **Mangles** the data according to pre-defined **mutation heuristics**

MAIN TYPES OF FUZZING

GENERATION BASED:

- Will be based on the grammar and format of the expected input
- **Test cases created from scratch**

MAIN TYPES OF FUZZING:
INPUT AWARENESS

BLACK BOX (DUMB FUZZING):

- **The fuzzer knows nothing or little to nothing** about the program or its input, applies random mutations or generations
- **Disadvantages:** shallow test cases and less results

MAIN TYPES OF FUZZING:
INPUT AWARENESS

FORMAT-AWARE (“SMART” FUZZING):

- Generation of inputs **based on a pre-defined specification**
- Test cases are more likely to be processed by the program: **better results**
- **It traverses more code,** going deeper into the program’s state

MAIN TYPES OF FUZZING:
OTHER KINDS

IN-MEMORY FUZZING

- Targets a specific function for fuzzing
- Based on the concept of **context snapshots** and **restore points**

MAIN TYPES OF FUZZING:
OTHER KINDS

EVOLUTIONARY (COVERAGE-GUIDED) FUZZING

- Generates or mutates new inputs **based on feedback from the input being fuzzed**
- New inputs are created with **genetic algorithms**
- **Became popular with AFL** (American Fuzzy Lop)

MAIN TYPES OF FUZZING:
WHAT CAN BE FUZZED? (1/2)

Web applications

Browsers

Image editors and document readers

Libraries and APIs

Command line parameters

MAIN TYPES OF FUZZING:
WHAT CAN BE FUZZED? (2/2)

Kernels
system calls
IOCTLs

Network
stacks and
protocols

Server-side
applications

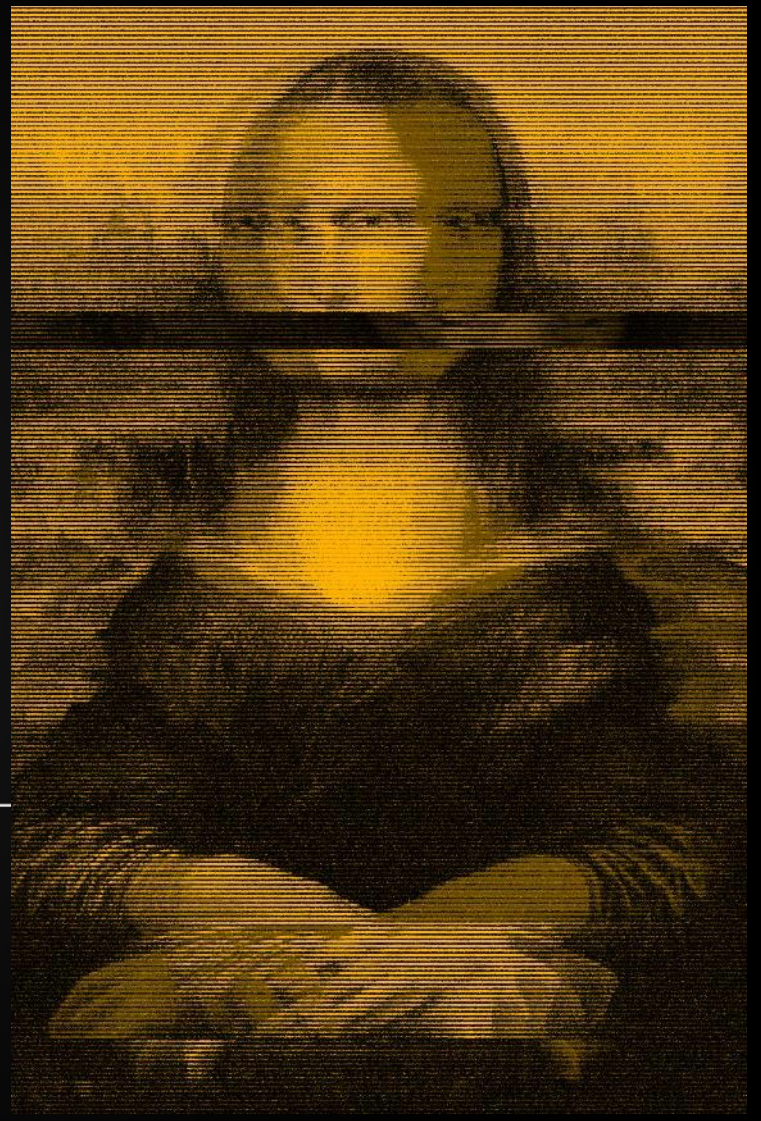
Interpreters
and
engines

...and
much
more!

BLAZE

BLAZE

WHAT THE FUZZ?



“GLITCH ART“

Purposefully causing glitches in images and video to create visual aesthetics

BASIC FUZZING EXAMPLE:
MUTATIONAL

LET'S GLITCH KRAKÓW'S RYNEK GŁÓWNY

Basic fuzzing example:

MUTATIONAL DEMO

BLAZE

Basic fuzzing example:

GENERATIONAL DEMO

BLAZE

MUTATION STRATEGIES AND HEURISTICS

- Bit and byte **flipping**
- Addition of known **corner case inputs** (MAX_INT, negative numbers, very long strings, etc.)
- **Deletion** of random portions of data
- **Swapping** bits and bytes within a file
- ...and **many more!**

Mutation strategies example:

DEMO

BLAZE

Popular tools

BLAZE

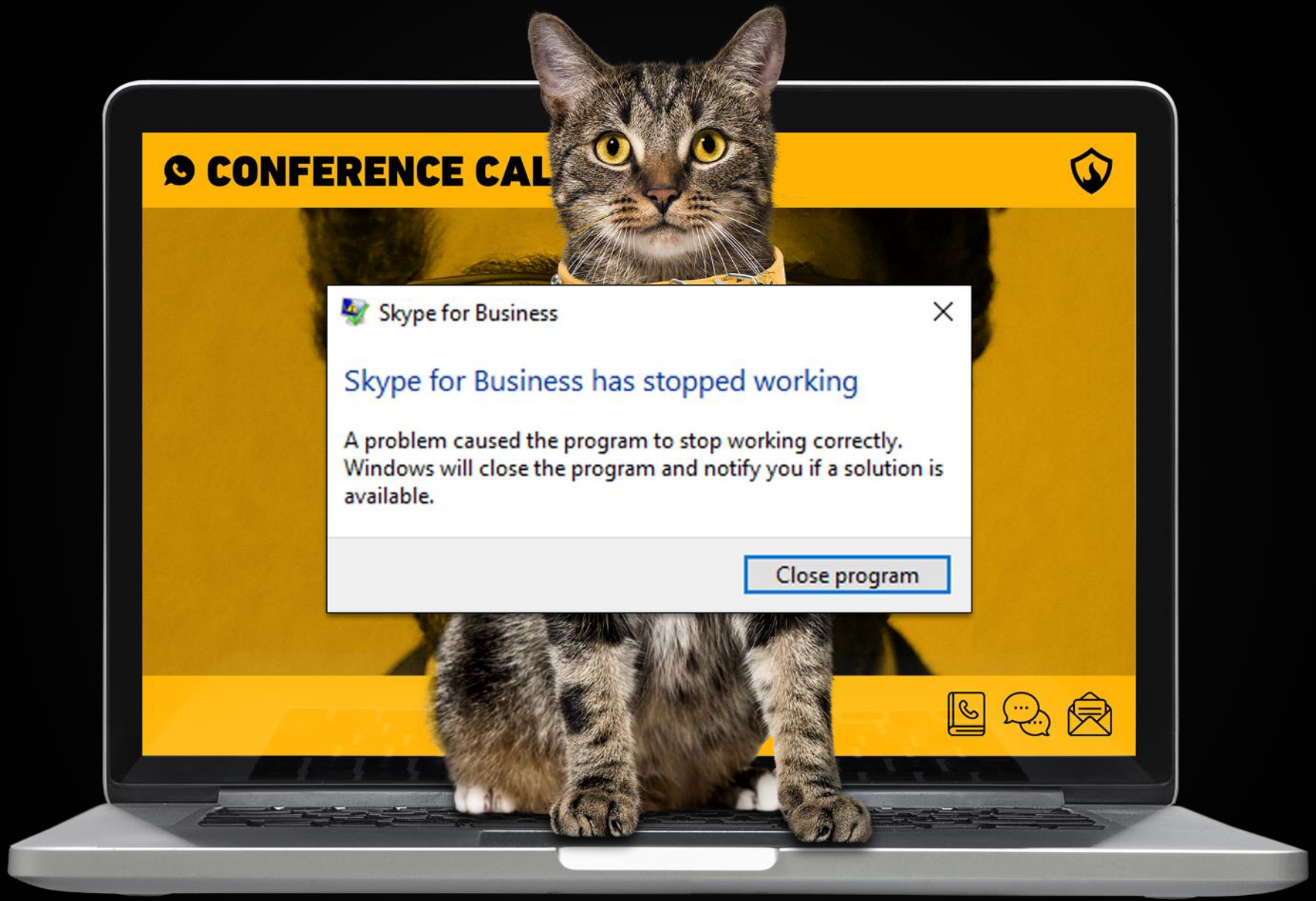
POPULAR TOOLS

- **Mutation-based:** zzuf, radamsa
- **Mutation-based, coverage-guided:** afl, WinAFL, honggfuzz
- **In-memory:** LibFuzzer
- **Generational:** Peach, boofuzz
- Your own?

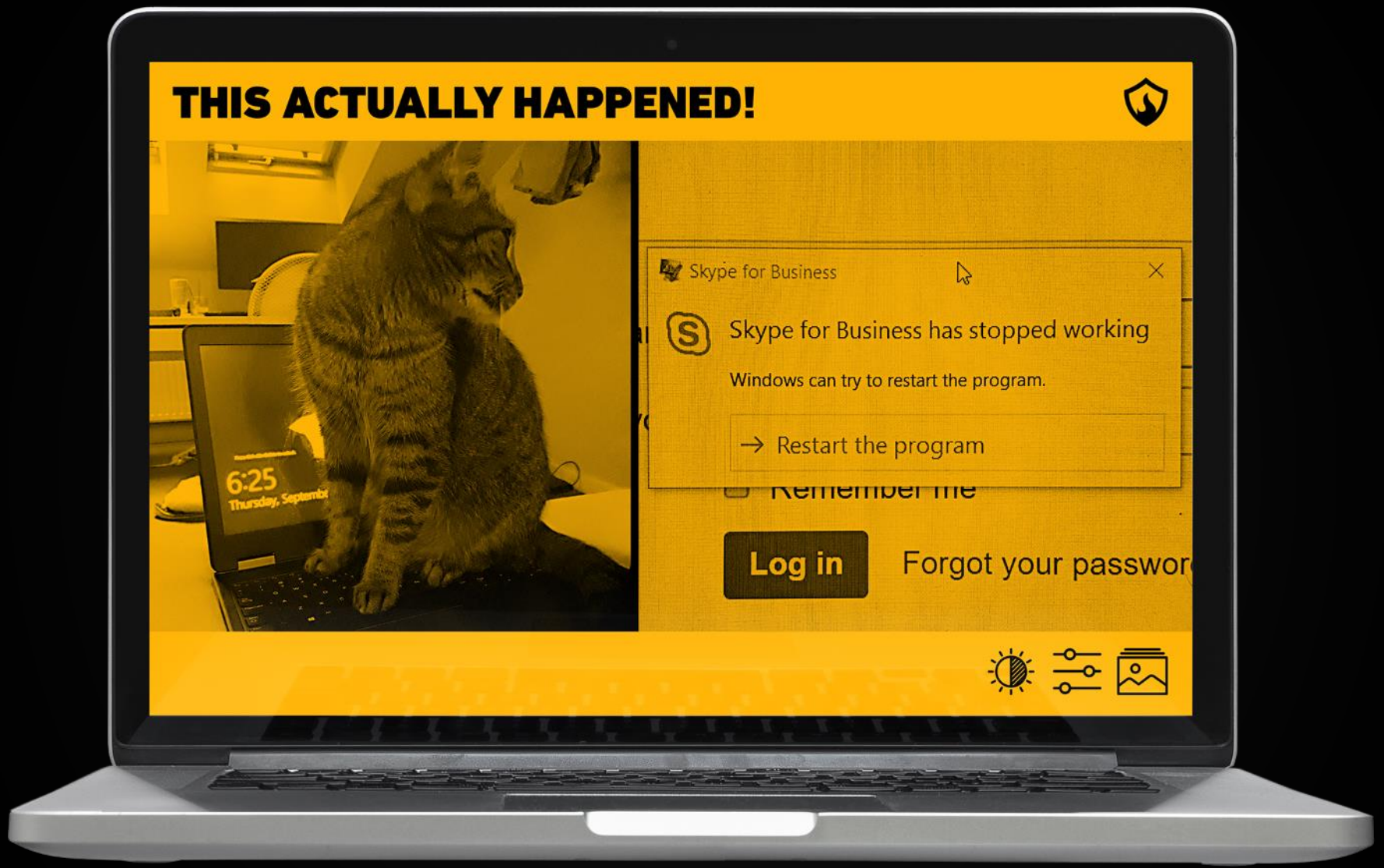
NOT SO
POPULAR TOOLS

BLAZE

BLAZE


CONFERENCE CAL
Skype for Business
Skype for Business has stopped working
A problem caused the program to stop working correctly. Windows will close the program and notify you if a solution is available.
Close program

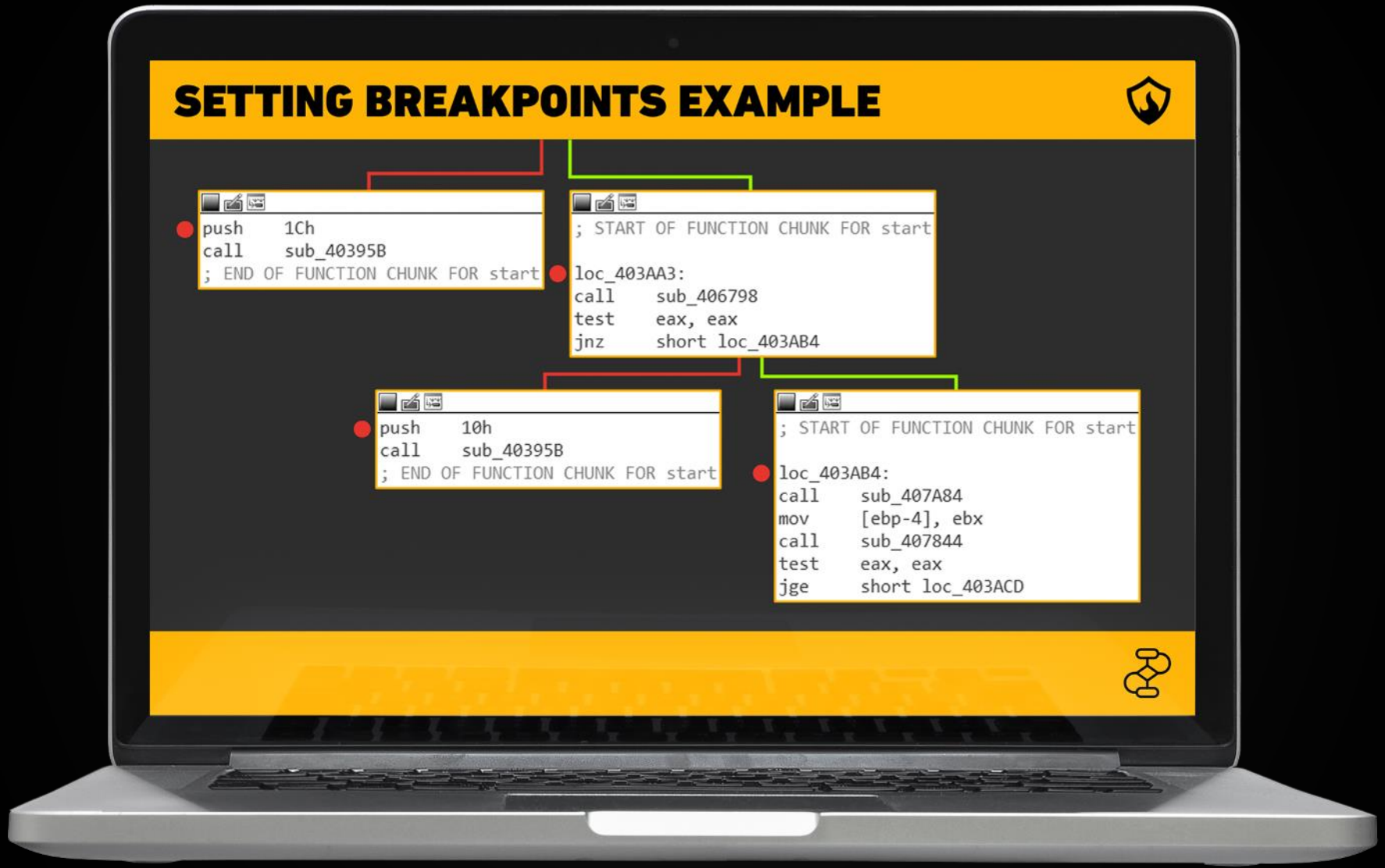
BLAZE


THIS ACTUALLY HAPPENED!
6:25
Skype for Business
Skype for Business has stopped working
Windows can try to restart the program.
→ Restart the program
Log in
Forgot your passwor

BLAZE

WORDS ON CODE COVERAGE

- **Compile-time** instrumentation
- **Binary emulation**
- Breakpoint in **every** basic block
- **Dynamic** binary instrumentation

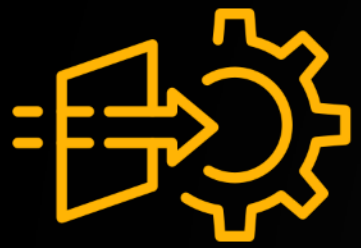

SETTING BREAKPOINTS EXAMPLE
push 1Ch
call sub_40395B
; END OF FUNCTION CHUNK FOR start
; START OF FUNCTION CHUNK FOR start
loc_403AA3:
call sub_406798
test eax, eax
jnz short loc_403AB4
push 10h
call sub_40395B
; END OF FUNCTION CHUNK FOR start
; START OF FUNCTION CHUNK FOR start
loc_403AB4:
call sub_407A84
mov [ebp-4], ebx
call sub_407844
test eax, eax
jge short loc_403ACD

BLAZE

Pre and post-fuzzing

BLAZE

PRE-FUZZING (1/2)



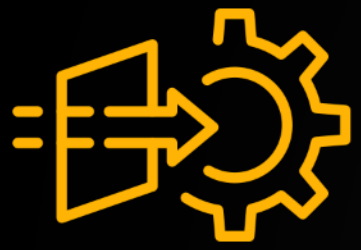
CORPUS

- **A valid set of samples** for a particular file format or protocol
- **Valid cases will be mutated** to create malformed inputs
- Ideally, **a diverse corpus set** that exercises different code paths

PROBLEM: MANY SAMPLES HIT THE SAME CODE PATHS

- **Solution:** Techniques and tools for corpus minimization

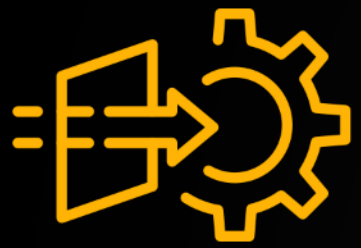
PRE-FUZZING (2/2)



CORPUS MINIMIZATION

- Find the **minimal set** of files that has similar coverage
- Reduce a file to the least possible size, with the same coverage

PRE-FUZZING (3/3)



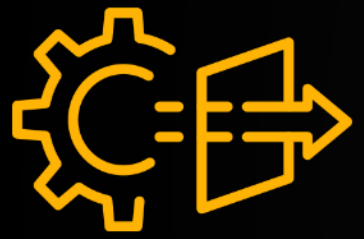
TEST HARNESS

- **How to deliver the fuzzed payload** to the application, a piece of code or library?
- Sometimes we need to **bridge the gap between the fuzzer and the program under test**

PROBLEM: SOMETIMES HARNESSING CAN BE A PAIN

- **Solution:** Just accept it. Fuzzing sometimes can be an ungrateful task

POST-FUZZING



- **Crash case minimization**
- **Crash triaging** for exploitability

DevSecOps
integration

BLAZE

DEVSECOPS INTEGRATION

- **Fuzzing can** (and should) **be integrated as part of the security-oriented DevOps pipeline**
- Existing initiatives such as: **Google's OSS-Fuzz, fuzzbuzz.io** and **fuzzit.dev**
- Or roll out **your own!**

DevSecOps integration:

DEMO

BLAZE

Conclusion

BLAZE

CONCLUSION

- **Fuzzing can be very effective and useful**
- The **entry-level bar** for fuzz testing has been **lowered** down significantly
- **Adding it to your pipeline will greatly improve the robustness of your software!**

REFERENCES

USEFUL REFERENCES TO CHECK OUT

- Fuzzing: Brute Force Vulnerability Discovery (Sutton, Greene, Amini)
- “Baby sitting an army of monkeys” (Charlie Miller)
- Fuzzing entry on Wikipedia
- Aalto University – Software security spring 2018
- The Cisco Secure Development Lifecycle: An Overview
- An Empirical Study of the Reliability of UNIX Utilities (Barton P. Miller)

Questions?

BLAZE

Be secure. Be ahead. Be Blaze.

THANK YOU!

www.blazeinfosec.com

info@blazeinfosec.com

Brazil
Rua Visconde de Jequitinhonha
279. Office 701. Recife

Portugal
Praça Bom Sucesso 131
Península, Office 206, Porto

Poland
Rynek Główny 28
33-332, Kraków

BR: +55 81 3071 7148
PT: +351 222 463 641
PL: +48 792 436 755

BLAZE