



# :/ TOOL TALK /:

EPISODE 2:

# FUZZING

GET THE BUZZ ON FUZZ TESTING IN  
SOFTWARE DEVELOPMENT



View Slides Full-Screen

Ask Question


Enter your question

Submit




Ask Questions Here




Speaker Bio






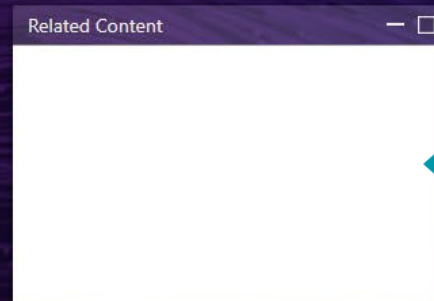
**Matt Keeley**  
Senior Security Consultant  
Bishop Fox



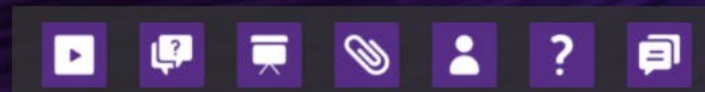


**Joe Sechman**  
AVP Research & Development  
Bishop Fox





Learn More About Our Research



Control Panel for Widgets

# Meet Your Hosts



**Matt Keeley**

SENIOR SECURITY CONSULTANT



**Joe Sechman**

ASSOCIATE VP OF R&D



# TODAY'S AGENDA

- FUZZING BASICS
- HOW FUZZING WORKS
- POPULAR FUZZING TOOLS
- DEMO
- Q & A



# 01 WHAT IS FUZZING?



# FUZZING

---

- » Technique for both security researchers and developers
- » May be a form of two testing types:
  - » Blackbox
    - There is no source code available
    - You will find most security researchers or anyone looking to test the security of a proprietary or closed source project using this method
  - » Whitebox
    - There is source code available
    - Most software developers will be using this on their project since they usually have the code for their own project.

# MAIN GOAL

---

- » Automate the finding of potentially vulnerable sections of a program
- » Use fuzzing to reveal potential bugs through unintended or anomalous behaviors in the application being fuzzed
  - Crashes
  - Infinite loops
  - Otherwise unknown "bad" behaviors
- » Claim the given program is robust enough to perform as intended or to find the bugs in the program so the developer can remediate them

# WHY

---

» Fuzzing can help you find:

- General memory corruptions (heap, stack, etc.)
- Race conditions
- Algorithmic Attacks
- Information Disclosure
- Many more! Much more research on this topic. Constantly evolving.



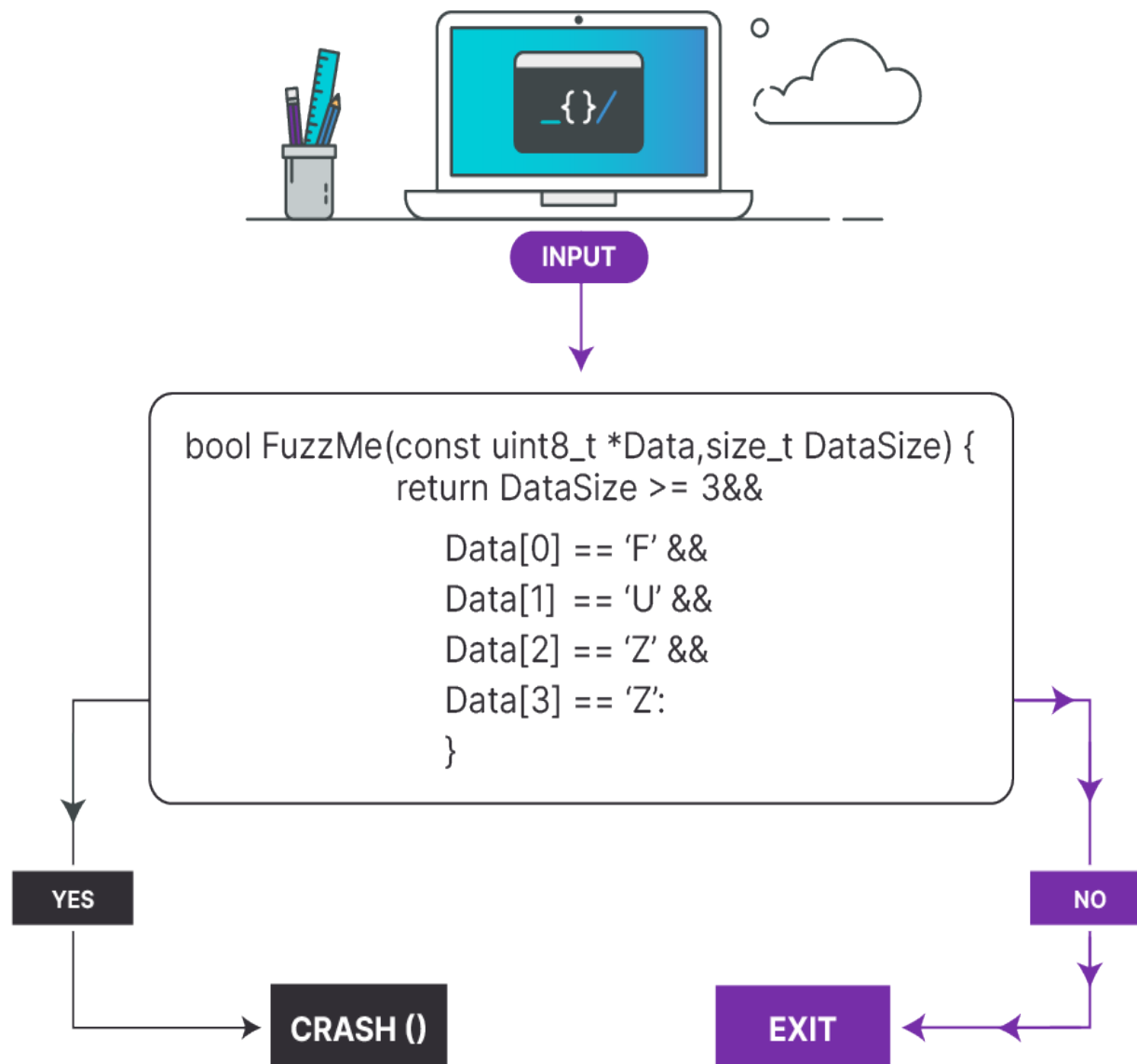


# 02

## HOW DOES IT WORK?

# VISUAL

# TEST



# VISUAL

# FUNK



INPUT

```
bool FuzzMe(const uint8_t *Data, size_t DataSize) {  
    return DataSize >= 3 &&
```

```
    Data[0] == 'F' &&
```

```
    Data[1] == 'U' &&
```

```
    Data[2] == 'Z' &&
```

```
    Data[3] == 'Z':
```

```
}
```

YES

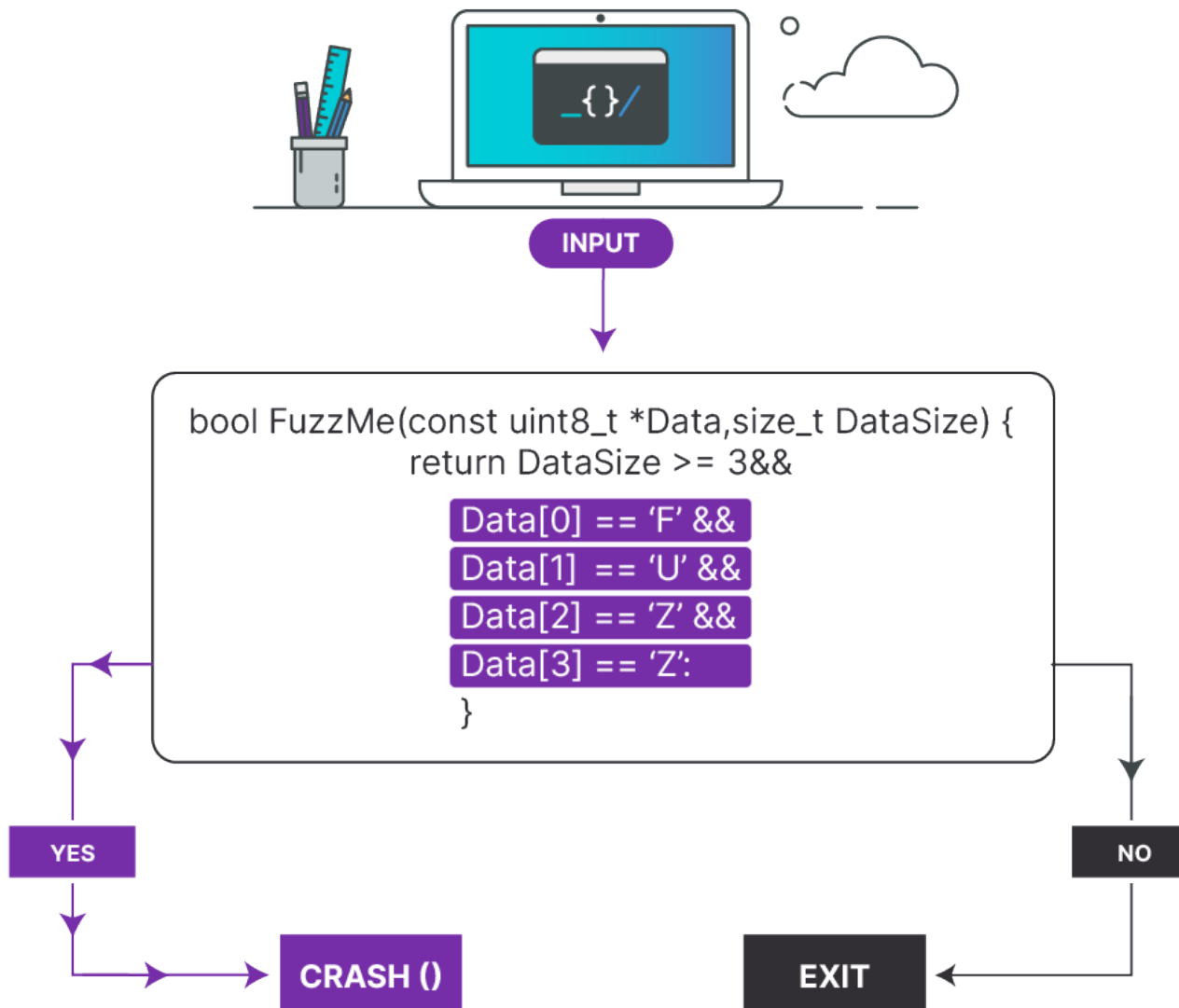
CRASH ()

NO

EXIT

# VISUAL

# FUZZ



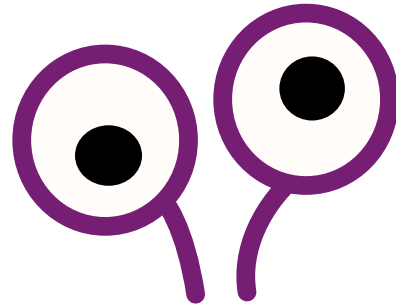
# POPULAR FUZZING TOOLS

---



## AFL++

Brute-force fuzzer coupled with an instrumentation-guided genetic algorithm.



## BooFuzz

Framework written in Python that allows hackers to specify protocol formats and perform fuzzing.



## Fuzzili

Multithreaded fuzzer written for JavaScript applications

# DEMO

---



**YOU'VE GOT QUESTIONS**  
**WE'VE GOT ANSWERS**





# THANK YOU

 [Contact@bishopfox.com](mailto:Contact@bishopfox.com)

 [Twitter.com/BishopFox](https://twitter.com/BishopFox)

 [Linkedin.com/Company/Bishop-Fox](https://linkedin.com/company/Bishop-Fox)

 [GitHub.com/BishopFox](https://github.com/BishopFox)

 [Facebook.com/BishopFoxConsulting](https://facebook.com/BishopFoxConsulting)

 [Reddit r/RedSec/](https://reddit.com/r/RedSec/)

 [Discord redsec](https://discord.com/redsec)

 [www.BishopFox.com](https://www.BishopFox.com)