

# Usability Issues of Modern Fuzzers

Matthew Smith

Usable Security and Privacy Lab, Universität Bonn, Fraunhofer FKIE,





# Outline

- Chapter 1: Usable Security Introduction
- Chapter 2: Fuzzing vs Static Analysis Usability Study

@m42smith





# Chapter 1

## Usable Security & Privacy

@m42smith

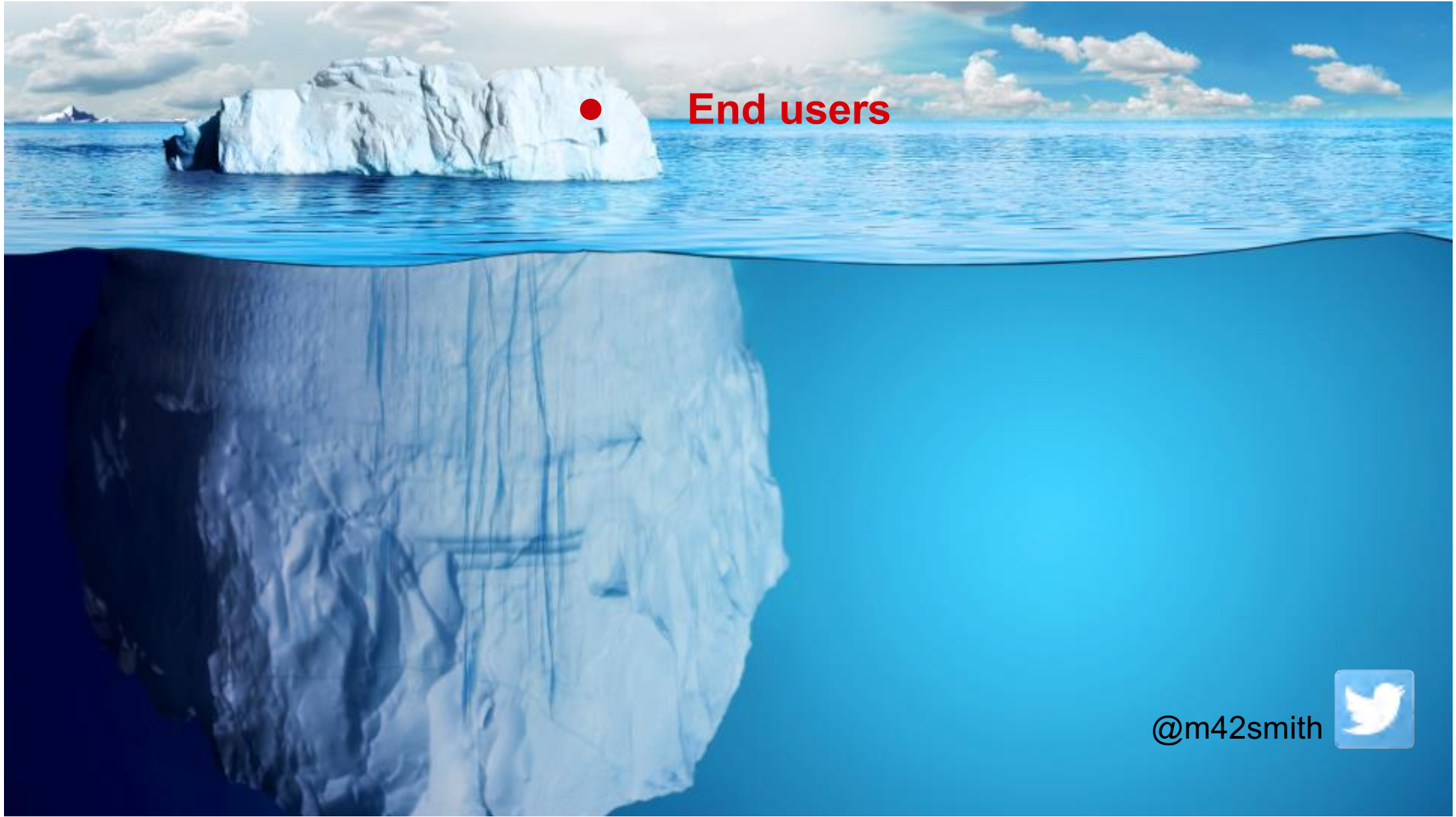




# The goal of Usable Security Research is to make security easy!

@m42smith





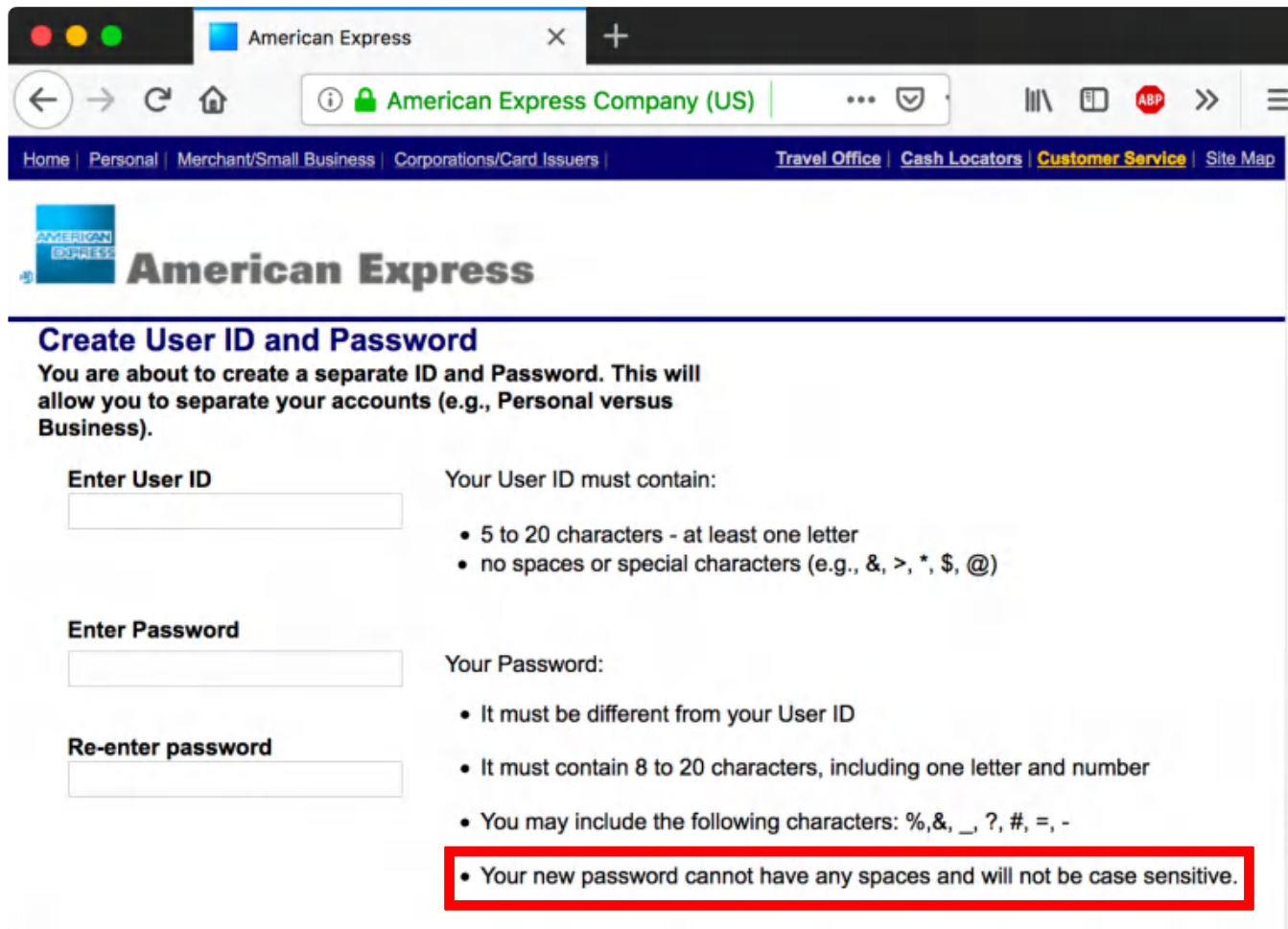
● End users

@m42smith





# Administrator Issues: Terrible Password Policies



**Create User ID and Password**  
You are about to create a separate ID and Password. This will allow you to separate your accounts (e.g., Personal versus Business).

**Enter User ID**

Your User ID must contain:

- 5 to 20 characters - at least one letter
- no spaces or special characters (e.g., &, >, \*, \$, @)

**Enter Password**

Your Password:

- It must be different from your User ID
- It must contain 8 to 20 characters, including one letter and number
- You may include the following characters: %, &, \_, ?, #, =, -
- Your new password cannot have any spaces and will not be case sensitive.

**Re-enter password**



@m42smith





## Developer Issues: Large scale disasters

**THE VERGE**

TECH ▾

SCIENCE ▾

CULTURE ▾

CARS ▾

REVIEWS ▾

LONGFORM

VIDEO

MORE ▾

US & WORLD / TECH / CYBERSECURITY

# Yahoo says all 3 billion user accounts were impacted by 2013 security breach

by [Natt Garun](#) | [@nattgarun](#) | Oct 3, 2017, 5:07pm EDT

25.01.2019 12:51 Uhr | Security

## Neue Passwort-Leaks: Insgesamt 2,2 Milliarden Accounts betroffen

Nach der Passwort-Sammlung Collection #1 kursieren nun auch die riesigen Collections #2-5 im Netz. So überprüfen Sie, ob Ihre Accounts betroffen sind.



@m42smith











Trust me! I know what I'm doing!



@m42smith





# Developers are not the enemy!

Green and Smith'16

@m42smith





## Developer User Study Lab





# Chapter 2

## Fuzzing vs Static Analysis







We had planed a SAST & DAST evaluation, but...

 coverity™

american fuzzy lop



 FORTIFY

honggfuzz

 CHECKMARX

libFuzzer



## SAST companies don't seem to want any scrutiny...

"Customer will not disclose to any third party any comparison of the results of operation of xxx's products with other products. "

Software Evaluation License Agreement

"Education Institute will not disclose to any third party any comparison of the results of operation of xxx' Licensed Products with other products, except as expressly permitted by this Agreement."

ACADEMIC END USER SOFTWARE LICENSE AND MAINTENANCE  
AGREEMENT

@m42smith





## Usability comparison



clang static analyser

<https://clang-analyzer.llvm.org/>



libFuzzer

<https://llvm.org/docs/LibFuzzer.html>



## Designing Static Tasks

- Selected popular OS projects
  - ran CLANG static analyser
- Selected two projects for the study
  - jq – JSON parser
  - Tesseract – OCR software
- We assumed reports to be FP
  - Added bugs for TP
- To defeat diff analysis:
  - used older version
  - removed version number

Program	Clang analyzer reports
Tesseract	476
protobuf 3.9.x	92
protobuf 3.8.x	121
util-linux	142
simple-obfs	15
cmatrix	3
vlc	219
wine	4746
netdata	32
darknet	73
libnice	3
obs-studio	456
jq	4
FFmpeg	639
yuzu	339
spdlog	0
simdjson	2

Table 1: Overview of github projects and reports of clang static analyzer





## Easy Static Task

- Selected jq JSON parser
  - 4 False Positives
  - Added 1 bug
    - found using default CLANG options
- Steps needed
  - scan-build ./configure
  - scan-build ./make
  - scan-view path/to/folder
  - examine the report manually
    - 5 items

@m42smith





## Hard Static Task

- Selected Tesseract OCR
  - 476 False Positives using default CLANG options
  - 658 False Positives with additional options
  - Added 2 bugs
    - default CLANG options
    - additional options
- Steps needed
  - Scan-build ./configure
  - Scan-build ./make
  - Scan-view path/to/folder
  - examine the output manually
    - 477 – 660 items

@m42smith





## Designing Fuzzing Tasks

- How to get an overview of different difficulty levels?
  - Interviewed fuzzing and pen-testing experts at Fraunhofer FKIE and Code Intelligence
- Selected
  - yaml-cpp as the easy project
  - surricata as the hard project

@m42smith





## Easy Fuzzing Task

- `yaml-cpp`
  - Small program
  - Small and obvious interface as entry point
  - Instrumentation and sanitizers not necessary
  - Build process does not need to be modified
  - Easy fuzz-target
  - Bug triggers very quickly

@m42smith







## Hard Fuzzing Task

- Surricata
  - Large program
  - Less clear where fuzzer needs to enter
  - Instrumentation and sanitizers are necessary
  - Build process needs to be modified
  - More complex fuzz-target
  - Bug triggers after longer time

@m42smith





## Study setup

- 32 students from the masters course Usable Security and Privacy
- Recruited based on a self-assessment questionnaire
  - At least 4 out of 7 in C/C++ skill ranking
  - At least 3 out of 5 Linux skill ranking
- 20 hours of work required
  - journal of activities
  - bug report
- 2 tasks per participant
  - easy static vs easy fuzzing
  - hard static vs hard fuzzing
- 11% bonus for exam
  - could write an essay instead





## Results

Task	started	aborted	submitted	success
Static-easy	12			
Static-hard	11			

Task	started	aborted	submitted	success
Fuzzing-easy	16			
Fuzzing-hard	11			

@m42smith





## Open Usability Problems

- CLANG static analyser
  - flood of false positives is a killer
  - participants used tool correctly but it was no help to them
- libFuzzer
  - high skill requirement
  - problems:
    - getting an overview of what needs to be done
    - selecting compilation parameters / adapting build process
    - writing fuzz-targets
  - documentation needs serious work
  - participants did not get to use tool properly

@m42smith

