android Bootcamp 2016

Proactive bug finding

Android Attack Team

Thursday January 21, 2016

Agenda

Why do we exist?

What do we do?

What’s next?

Why do we exist?

Android Attack Team

Find bugs and vulnerabilities before other people do.

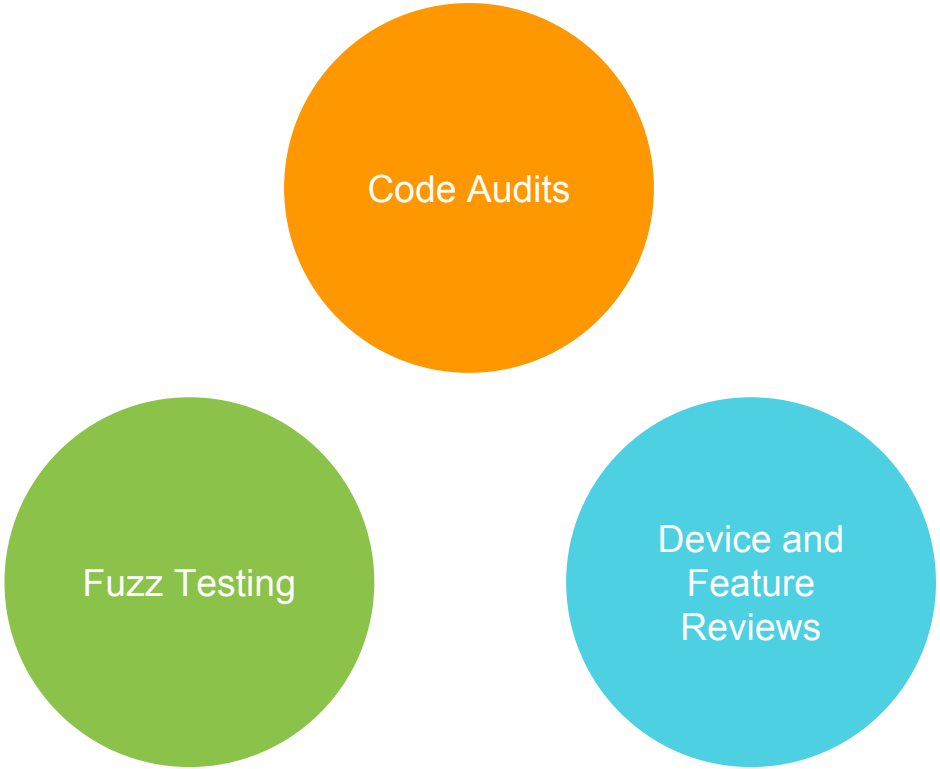
Analyze root cause, kill entire classes of bugs where possible.

Raise the cost of finding bugs and writing Android exploits.

Image from: http://game-icons.net

What do we do?

Our approach



Fuzz testing

Fuzz testing - why?

Scales well. After a fuzzer is up and running, it can be left for months without interaction to look for bugs.

Complements code reviews. Can reveal bugs that manual audits miss.

Raises the cost of exploits. External actors also fuzz, and fixing bugs we find makes finding new ones harder.

Image from: https://pixabay.com/

Fuzz testing - strategy

Map security relevant attack surfaces.

Identify attack vectors that reach those surfaces.

Use available tools to fuzz them.

Develop new tools to improve attack vector coverage.

Measure code coverage for each vector.

Fuzz testing - attack surfaces

- Media Codecs
- App IPC
- System Services
- Local pipes/sockets
- Kernel (device drivers, syscalls, debugfs/procfs)
- Open ports
- Writable files
- ...

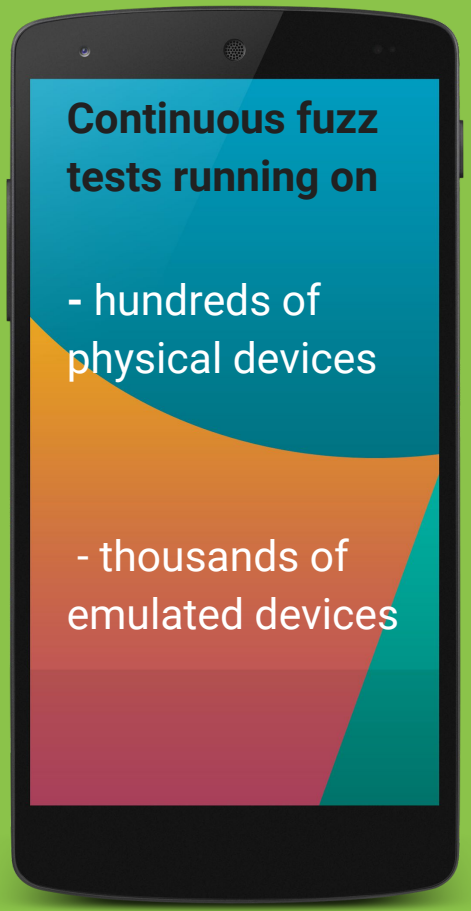
Fuzz testing - Cluster Fuzz

Scalable fuzzing framework.

Continuous fuzz tests on both physical and virtual devices.

Error reports with stack trace, logcat, etc, when it finds a bug.

Automatically checks for regressions, notifies when a bug is fixed, etc.



Code audits

Code audits

Manually audit security critical components of Android.

Unearthed critical bugs and helped component vulnerability analysis.

Image from: https://upload.wikimedia.org/

Device and feature reviews

Feature reviews

Review new features (platform and Google Play Services) before they ship.

Root cause analysis to identify common mistakes.

Prioritize critical, remotely-accessible components/features.

Recommend changes, exploit mitigations to developers.

Device reviews

ASDDB or Android Snapshot Device Database. Two components:

- **Tool** to collect a snapshot of various information from a device.
  - File, device node permissions
  - List of files and processes
  - Exposed system services and components, etc.
- **Centralized database** (accessible via a cloud front end) of snapshots from different model-Android version combinations.

Device reviews

Security reports are generated by comparing two or more snapshots and analyzing common and unique issues discovered across them.

- World-writable files and directories
- Accessible device nodes
- Open sockets
- Changed API surface, etc.

Easy to thoroughly vet a reference snapshot, and use it to compare multiple new snapshots quickly (and accurately).

What next?

Plans for 2016

Improve

- Increase fuzz testing coverage
- More code audits
- Enhance ASDDB and existing fuzzers
- Static analysis

Add

- Develop additional fuzzers
- Security pre-submit checks
- And more!

Image from: http://software-topics.com/

THANK YOU