About Me

$whoami

- Security engineer @ Fb > 2 years
- Security consultant
- I <3 CTFs (LC/BC)
- I <3 server side bugs and automating the detection
- @the_st0rm

- Agenda
  - Setting the scene
  - Securing the codebase
  - Example of rules
  - Static analysis use cases
  - Myth busting
  - Demo! :O

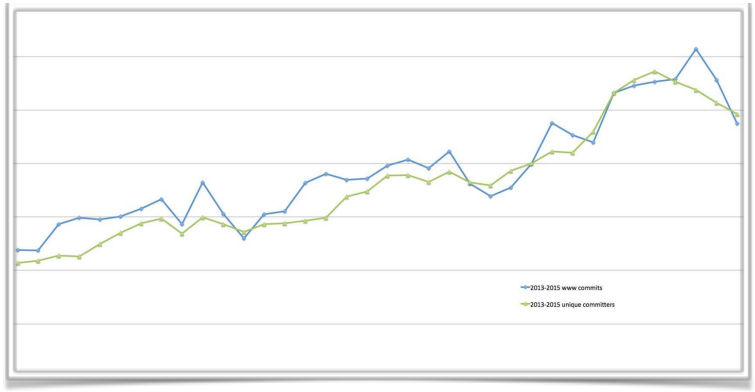
- Engineering @ FB

> 100k

commits per week

www for 2015:

- 609 Pushes
  - 51 weekly pushes
  - 439 daily pushes
  - 94 hotfixes



Android for 2015:

- FB for Android:
  - 34 releases
  - 1 hotfix
- Messenger
  - 39 releases
  - 0 hotfixes

iOS for 2015:

- FB for iOS:
  - 25 releases
  - 5 hotfixes
- Messenger:
  - 27 releases
  - 6 hotfixes

Big Code: Developer Infrastructure at Facebook's Scale

https://www.facebook.com/FacebookforDevelopers/videos/10152800517193553/

- Engineering @ FB

“Nothing at Facebook
is somebody else's problem”

- Securing the codebase
  - Secure frameworks
  - Security reviews
  - Automation (static and dynamic analysis)
  - Whitehat

- Secure frameworks
  - XHP
  - Hack
  - Django
- Limitations
  - Enforcement
  - Depends on the engineer

- Manual security reviews
  - Find cool bugs
- Limitations
  - Time consuming
  - Does not scale
  - Completeness

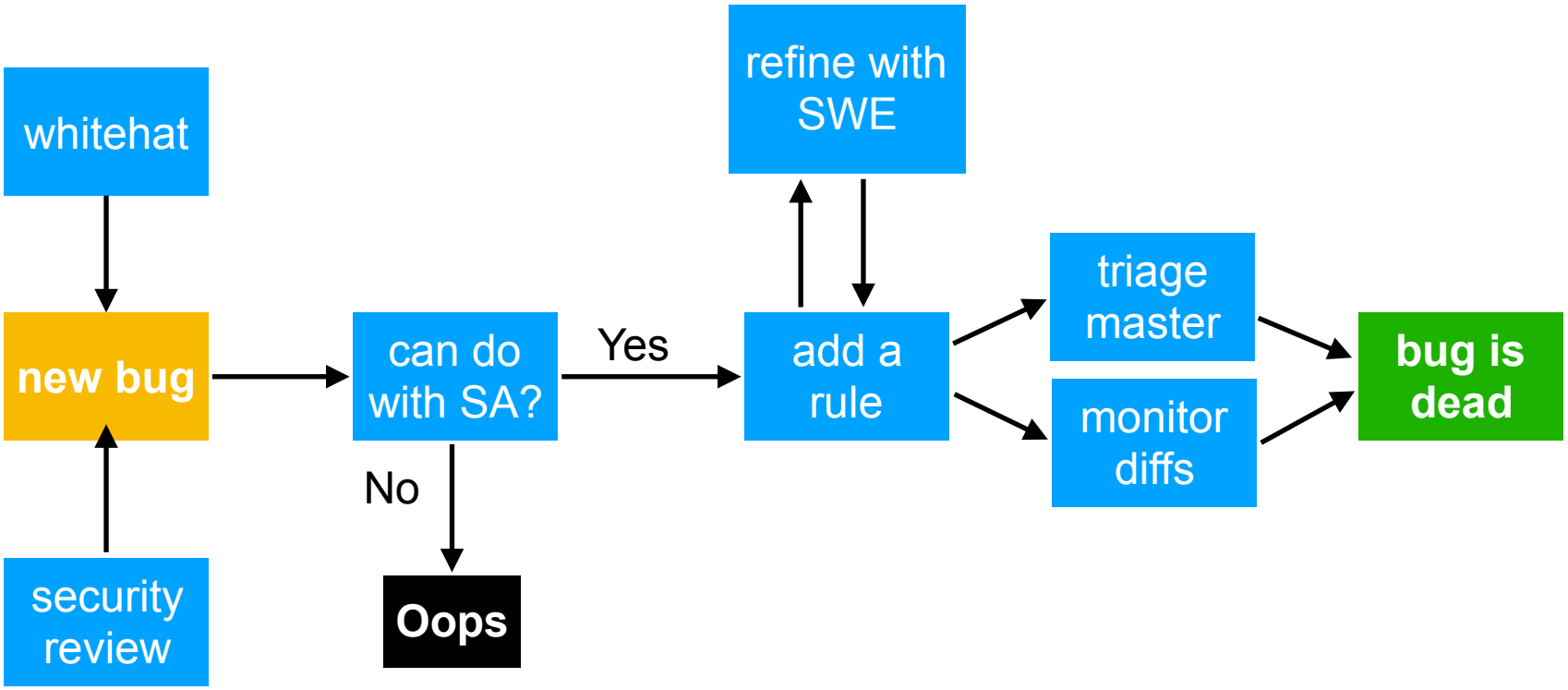
- Automation (Program analysis)
  - Scales
  - Find low hanging fruits
  - And difficult bugs (Fuzzing)
  - Continuous detection [+ prevention]
- Limitations
  - False positives and negatives
  - Difficult to get right

- Whitehat
  - Continuous detection
  - Very unique bugs/talent
- Limitations
  - Test in prod!
  - Expensive for small companies?
  - Signal to noise ratio

Automation (static analysis)

- Automation (static analysis)
  - Scale
    - Tens of millions LoC
    - Thousand commits/day
  - Performance
    - No run-time overhead (e.g fuzzing)
    - Grepping millions of LoC
  - Completeness
  - Proactive vs Reactive

- Static analysis design



- Tips to build good static analysis
  - Coverage
    - Understand the attack surface
    - Define sources
    - Define sinks
  - Simplicity
    - Easy to use
    - Configuring the sources/sinks
    - Adding sanitizers

- Tips to build good static analysis
  - Improving signal
    - Excluding False positives
    - Finding false negatives
  - Feedback to the framework
  - Speed

- Security vulnerabilities we detect
  - We can currently detect more than 20 types of security issues including
    - Higher-order command injection
    - HTTP status codes as privacy oracles
    - Arbitrary file reads/writes
    - Server-side Request Forgery (SSRF)
    - SQL
    - XSS

- Bug detection - Arbitrary file reads/writes
  - Filename going to dangerous function

```
$path = $_FILES['upfile']['name'];
// …
Filesystem::readFile($path);
```

```
$path = $_FILES['upfile']['tmp_name'];
// …
Filesystem::readFile($path);
```

- Bug detection - command injection
- Secure because of high-quality frameworks

```
$t = attacker_controlled();
// … many lines …
execx(“zip %s”, $t);
```

```
$t = attacker_controlled();
// …
execx(“zip a.zip -T '--unzip-command=%s'”, $t);
```

- Commands can execute other commands
  - Static analysis tool can understand format string

--unzip-command cmd

Use command cmd instead of 'unzip -tqq' to test an archive when the -T option is used. On Unix, to use a copy of unzip in the current directory instead of the standard system unzip, could use:

```
zip archive file1 file2 -T -TT "./unzip -tqq"
```

In cmd, {} is replaced by the name of the temporary archive, otherwise the name of the archive is appended to the end of the command. The return code is checked for success (0 on Unix).

- Bug detection - Privacy oracles
  - Static analysis can check
    - action taken under attacker control?
    - action is influenced by privacy check?

```
$group_id = attacker_controlled();
if ($group_id === 100)
  throw HTTP_404();
```

```
$group_id = attacker_controlled();
// load with privacy check
$data = isMember(auth_user(), group_id);
if ($data === null)
  throw HTTP_404();
```

- Use cases
  - Regular analysis
    - Triaged by security engineers
    - Triaged by team owners
  - On-demand analysis
    - Whitehat report
    - Security reviews

- Use cases
  - Diff analysis
    - Analyze base repo
    - Analyze base repo + diff
    - Find new issues
    - High confidence issues => auto comment
    - Mid confidence => Oncall/product team

- Myth busting
  - Does it scale?
    - 20 mins for 10s millions of LoC
  - Is it precise?
    - “Static analyzers are noisy”
  - Is it useful?
    - “They only find trivial errors”

- Analysis dashboard

code	good_signal ↑	false_positive_pct	valid_pct	bad_practice_pct	dont_care_pct
6,053	100.0	0.0%	100.0%	0.0%	0.0%
6,084	100.0	0.0%	100.0%	0.0%	0.0%
6,309	100.0	0.0%	100.0%	0.0%	0.0%
7,000	100.0	0.0%	100.0%	0.0%	0.0%
6,304	91.4	8.6%	34.3%	57.1%	0.0%
6,082	83.3	0.0%	66.7%	16.7%	16.7%
6,026	80.0	20.0%	0.0%	80.0%	0.0%
6,054	78.4	21.6%	18.9%	59.5%	0.0%
6,029	75.0	25.0%	75.0%	0.0%	0.0%
6,052	75.0	12.5%	0.0%	75.0%	12.5%
6,086	60.0	7.5%	40.0%	20.0%	32.5%
6,031	57.1	28.6%	0.0%	57.1%	14.3%
6,059	57.1	14.3%	42.9%	14.3%	28.6%
6,030	50.0	17.6%	14.7%	35.3%	32.4%
6,064	50.0	0.0%	0.0%	50.0%	50.0%
6,017	45.0	10.0%	20.0%	25.0%	45.0%
6,016	41.2	26.5%	26.5%	14.7%	32.4%
6,051	40.0	40.0%	20.0%	20.0%	20.0%
6,310	38.1	14.3%	19.0%	19.0%	47.6%
6,095	35.7	14.3%	19.0%	16.7%	50.0%
6,070	33.3	44.4%	33.3%	0.0%	22.2%
6,089	33.3	66.7%	22.2%	11.1%	0.0%
6,066	21.4	71.4%	21.4%	0.0%	7.1%
6,305	20.0	60.0%	0.0%	20.0%	20.0%
6,087	19.6	47.1%	5.9%	13.7%	33.3%
6,020	18.8	75.0%	6.3%	12.5%	6.3%
6,018	15.4	23.1%	7.7%	7.7%	61.5%
6,062	14.3	85.7%	0.0%	14.3%	0.0%
6,050	3.3	6.7%	0.0%	3.3%	90.0%

- Have you heard about Pyre?
  - Pyre is a fast, scalable type checker for large Python 3 codebases
  - Open source
- Python static analysis?
- Demo?

We are hiring <3

Questions?