Move fast and secure things
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

Big Code: Developer Infrastructure at Facebook’s Scale
https://www.facebook.com/FacebookforDevelopers/videos/10152800517193553/
• Engineering @ FB

DONE IS BETTER THAN PERFECT

FAIL HARDER

STAY FOCUSED & KEEP SHIPPING
“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

- whitehat
- new bug
- can do with SA?
- Yes
- add a rule
- refine with SWE
- triage master
- monitor diffs
- bug is dead
- No
- Oops
- security review
• 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

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

```php
$path = $_FILES['upfile']['tmp_name'];
// ...
Filesystem::readFile($path);
```
• Bug detection - command injection

• Secure because of high-quality frameworks

$t = \text{attacker\_controlled}();$

// ... many lines ...
execx(“zip %s”, $t);

$t = \text{attacker\_controlled}();$

// ... 
execx(“zip a.zip -T '--unzip\_command=\%s'”, $t);

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

\textit{--unzip\_command cmd}

This section would show how to detect a specific command injection when the \textit{--unzip\_command} option is used. On Unix, to use a copy of unzip in the current directory instead of the standard system unzip, could use:

\texttt{zip archive file1 file2 -T -TT "/unzip \texttt{-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

<table>
<thead>
<tr>
<th>code</th>
<th>good_signal</th>
<th>false_positive_pct</th>
<th>valid_pct</th>
<th>bad_practice_pct</th>
<th>dont_care_pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>6,053</td>
<td>100.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,084</td>
<td>100.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,309</td>
<td>100.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>7,000</td>
<td>100.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,304</td>
<td>91.4%</td>
<td>8.6%</td>
<td>34.3%</td>
<td>57.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,082</td>
<td>83.3%</td>
<td>0.0%</td>
<td>66.7%</td>
<td>16.7%</td>
<td>16.7%</td>
</tr>
<tr>
<td>6,085</td>
<td>80.0%</td>
<td>20.0%</td>
<td>0.0%</td>
<td>80.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,054</td>
<td>78.4%</td>
<td>21.6%</td>
<td>18.9%</td>
<td>58.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,029</td>
<td>75.0%</td>
<td>25.0%</td>
<td>75.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,052</td>
<td>76.0%</td>
<td>12.6%</td>
<td>0.0%</td>
<td>76.0%</td>
<td>12.5%</td>
</tr>
<tr>
<td>6,086</td>
<td>60.0%</td>
<td>7.5%</td>
<td>40.0%</td>
<td>20.0%</td>
<td>32.5%</td>
</tr>
<tr>
<td>6,031</td>
<td>57.1%</td>
<td>28.6%</td>
<td>0.0%</td>
<td>57.1%</td>
<td>14.3%</td>
</tr>
<tr>
<td>6,099</td>
<td>57.1%</td>
<td>14.3%</td>
<td>42.9%</td>
<td>14.3%</td>
<td>28.6%</td>
</tr>
<tr>
<td>6,000</td>
<td>50.0%</td>
<td>17.8%</td>
<td>14.7%</td>
<td>35.3%</td>
<td>32.4%</td>
</tr>
<tr>
<td>6,044</td>
<td>50.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>50.0%</td>
<td>50.0%</td>
</tr>
<tr>
<td>6,017</td>
<td>45.0%</td>
<td>10.0%</td>
<td>20.0%</td>
<td>25.0%</td>
<td>45.0%</td>
</tr>
<tr>
<td>6,016</td>
<td>41.2%</td>
<td>26.5%</td>
<td>26.5%</td>
<td>14.7%</td>
<td>32.4%</td>
</tr>
<tr>
<td>6,001</td>
<td>40.0%</td>
<td>40.0%</td>
<td>20.0%</td>
<td>20.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td>6,310</td>
<td>38.1%</td>
<td>14.3%</td>
<td>19.0%</td>
<td>19.0%</td>
<td>47.6%</td>
</tr>
<tr>
<td>6,095</td>
<td>35.7%</td>
<td>14.3%</td>
<td>19.0%</td>
<td>16.7%</td>
<td>50.0%</td>
</tr>
<tr>
<td>6,070</td>
<td>33.3%</td>
<td>44.4%</td>
<td>33.9%</td>
<td>0.0%</td>
<td>22.2%</td>
</tr>
<tr>
<td>6,089</td>
<td>33.3%</td>
<td>66.7%</td>
<td>22.2%</td>
<td>11.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,066</td>
<td>21.4%</td>
<td>71.4%</td>
<td>21.4%</td>
<td>0.0%</td>
<td>7.1%</td>
</tr>
<tr>
<td>6,305</td>
<td>20.0%</td>
<td>60.0%</td>
<td>0.0%</td>
<td>20.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td>6,087</td>
<td>19.6%</td>
<td>47.1%</td>
<td>5.9%</td>
<td>13.7%</td>
<td>33.3%</td>
</tr>
<tr>
<td>6,020</td>
<td>18.8%</td>
<td>75.0%</td>
<td>6.3%</td>
<td>12.5%</td>
<td>6.3%</td>
</tr>
<tr>
<td>6,018</td>
<td>15.4%</td>
<td>23.1%</td>
<td>7.7%</td>
<td>7.7%</td>
<td>61.5%</td>
</tr>
<tr>
<td>6,062</td>
<td>14.3%</td>
<td>85.7%</td>
<td>0.0%</td>
<td>14.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td>6,050</td>
<td>3.3%</td>
<td>6.7%</td>
<td>0.0%</td>
<td>3.3%</td>
<td>90.0%</td>
</tr>
</tbody>
</table>
• 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?