

DEVOPSDAYS BOGOTÁ 2020

Septiembre 23-25, 2020
Online version

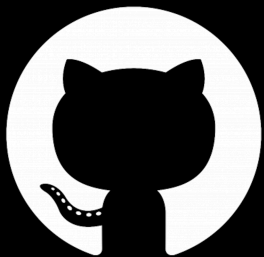


Sasha Rosenbaum

GERENTE DE PRODUCTO EN GITHUB

KEYNOTE SPEAKER





Security is too hard.
It's time for automation!

Sasha Rosenbaum
@DivineOps



Dev
Ops
Architect
Product Manager

Microsoft => GitHub

@DivineOps

And you?

State of security today



More code = more problems

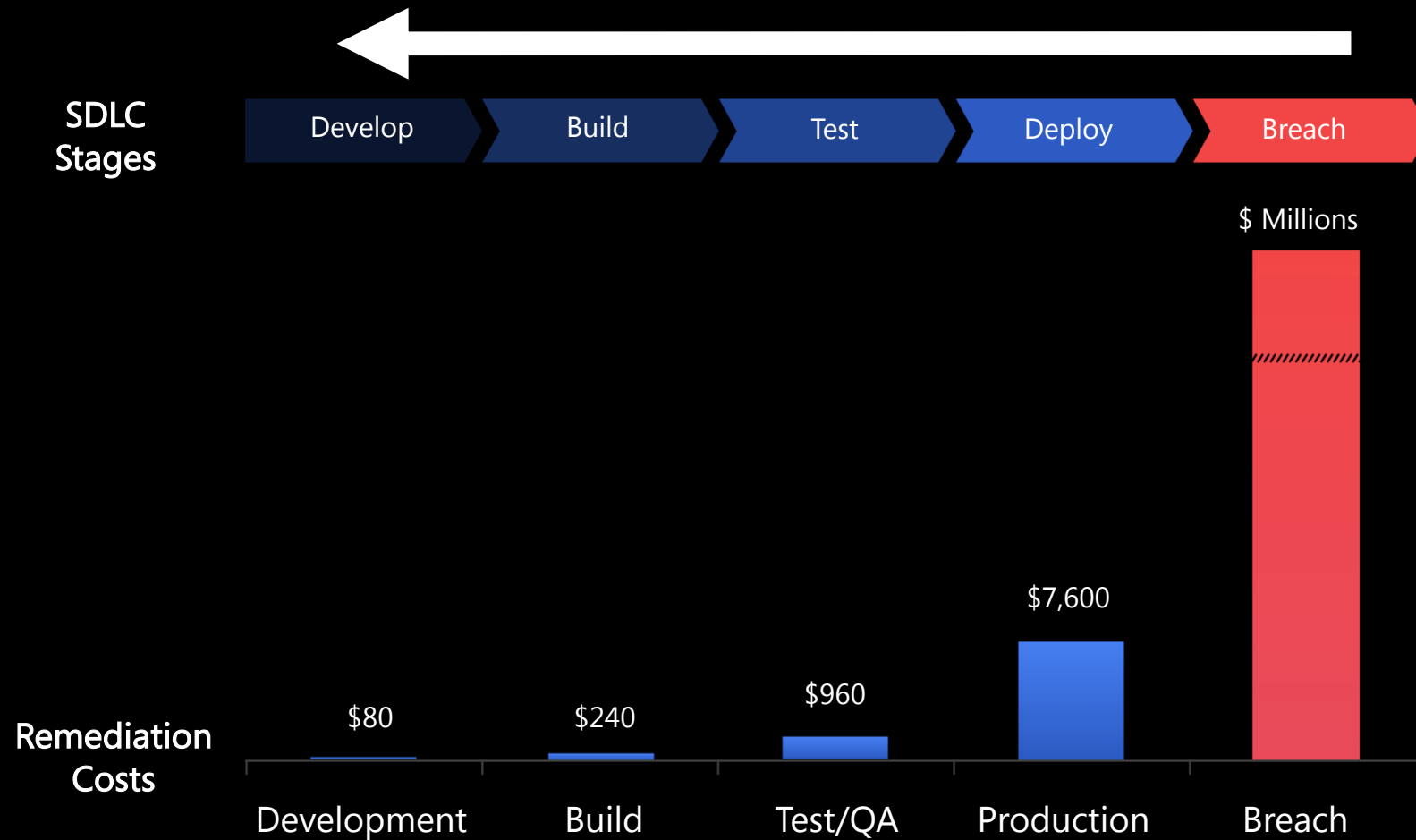


Insecure code causes
breaches

53%

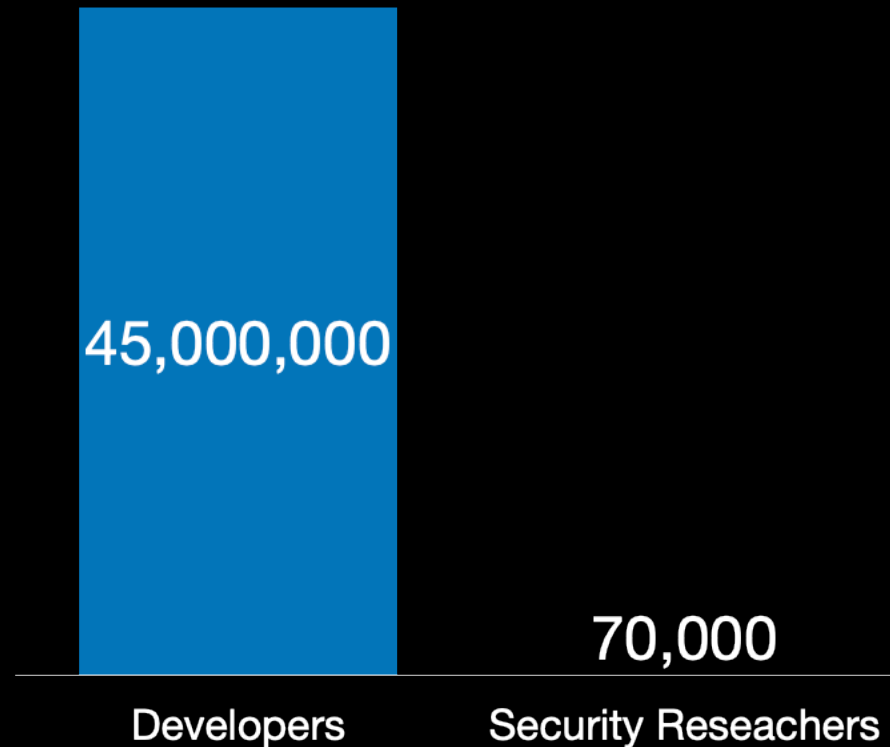
of breaches are caused by
weaknesses in applications

The earlier we remediate, the better!



Security researchers are outnumbered!

570x more developers than security researchers



Assume Breach

There are two types of companies:
those that have been hacked,
and those that don't know they have been hacked



Protect Credentials



The Two Widest Back Doors

- Credential Theft
- Exploiting Known Vulnerabilities

Attackers have changed their playbook...

How do breaches occur?

46% 

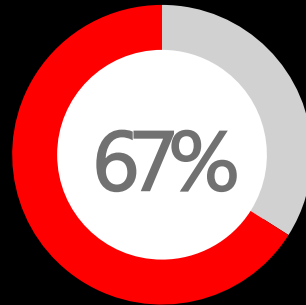
of compromised systems had no malware on them

99% 

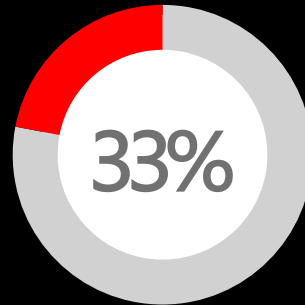
Of the exploited vulnerabilities were compromised more than a year after the CVE was published.



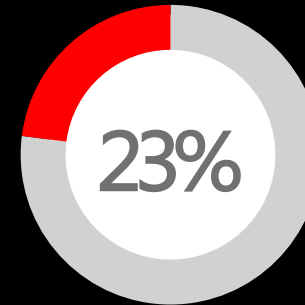
of victims have **up-to-date anti-virus signatures**



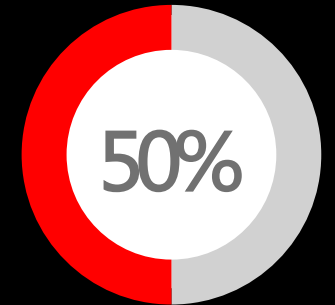
of victims were **notified** by an **external** entity



of victims **discovered** the breach **internally**

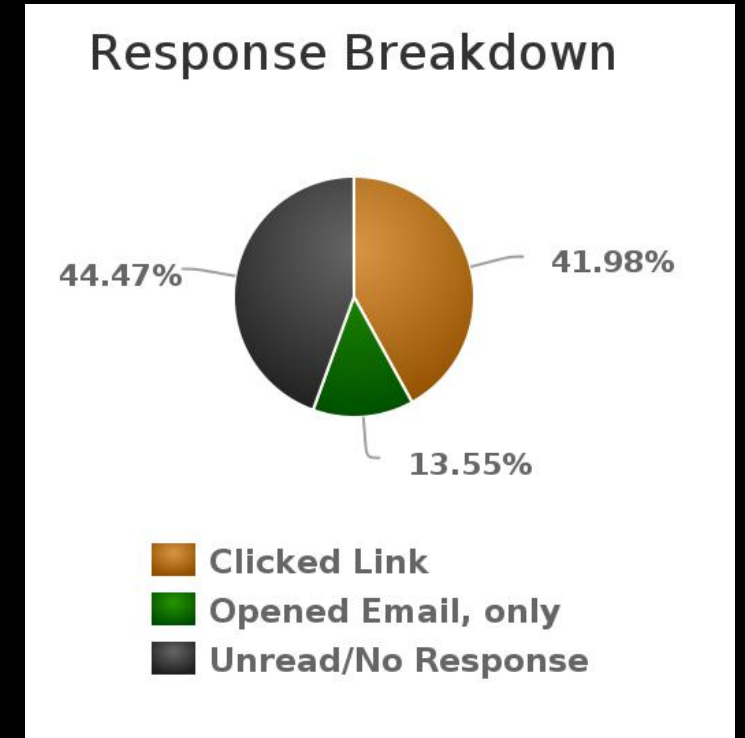
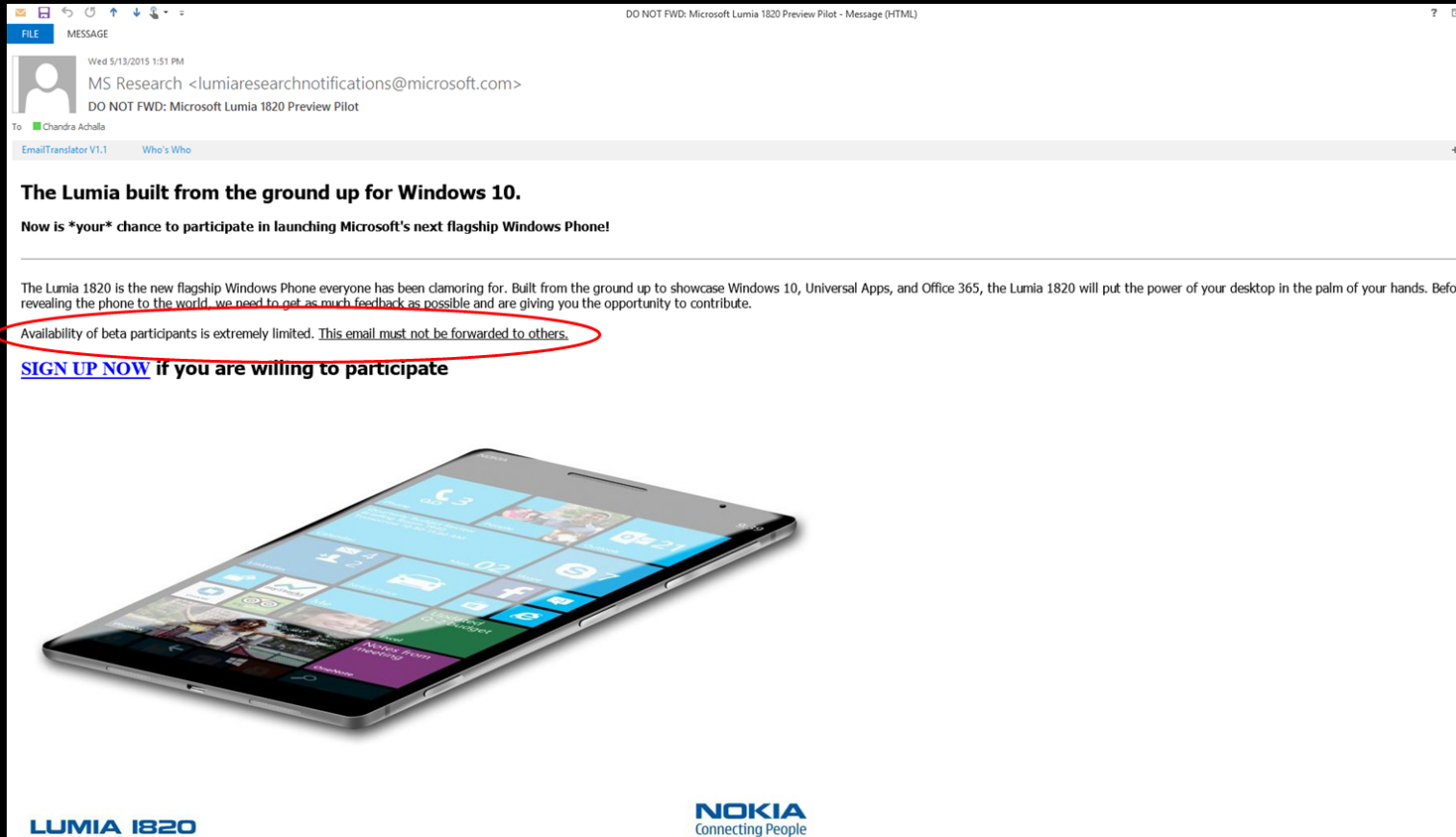


Of recipients **open phishing messages** (11% click on attachments)



Nearly 50% open e-mails and **click on** phishing links **within the first hour**.

Phishing



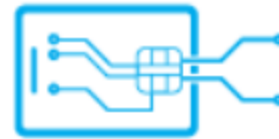
- Total population of 524 people.
- 220 people clicked on signup button. 37 people clicked on both phishing emails
- Only 11 people (2%) reported to as probable phish!

Employee awareness training is not very effective in preventing phishing attacks



Multi-Factor Authentication

Username
someone@example.com
Password



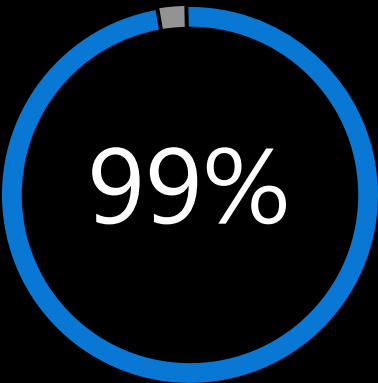
Email protection



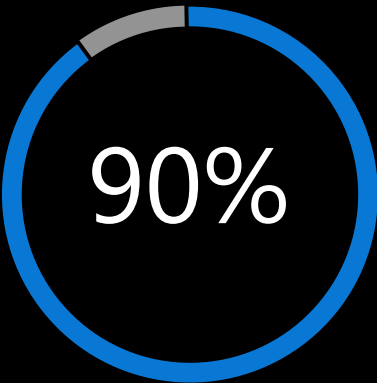
Securing the software supply chain

How much do you rely on open source?

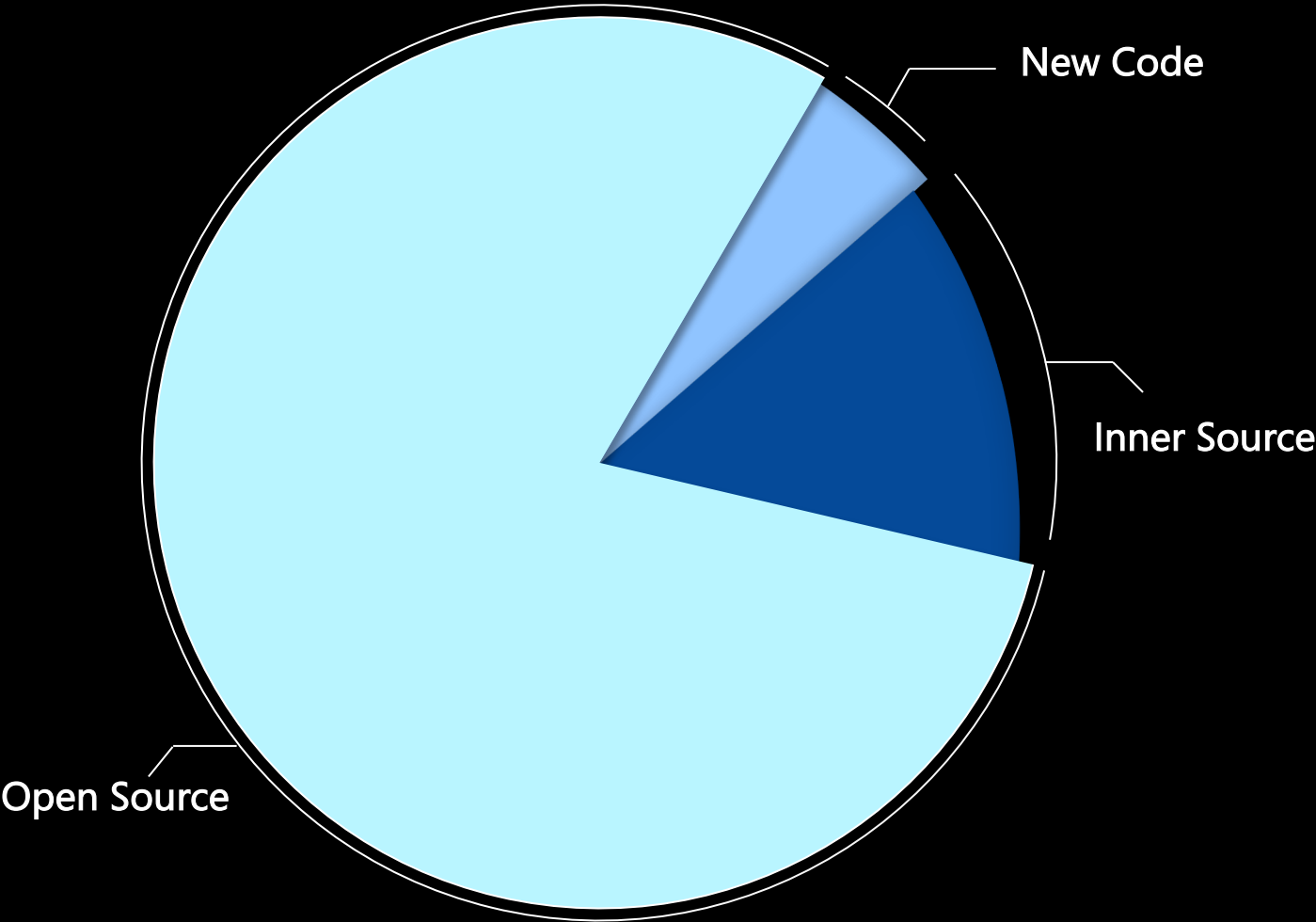
Open source software in the Enterprise



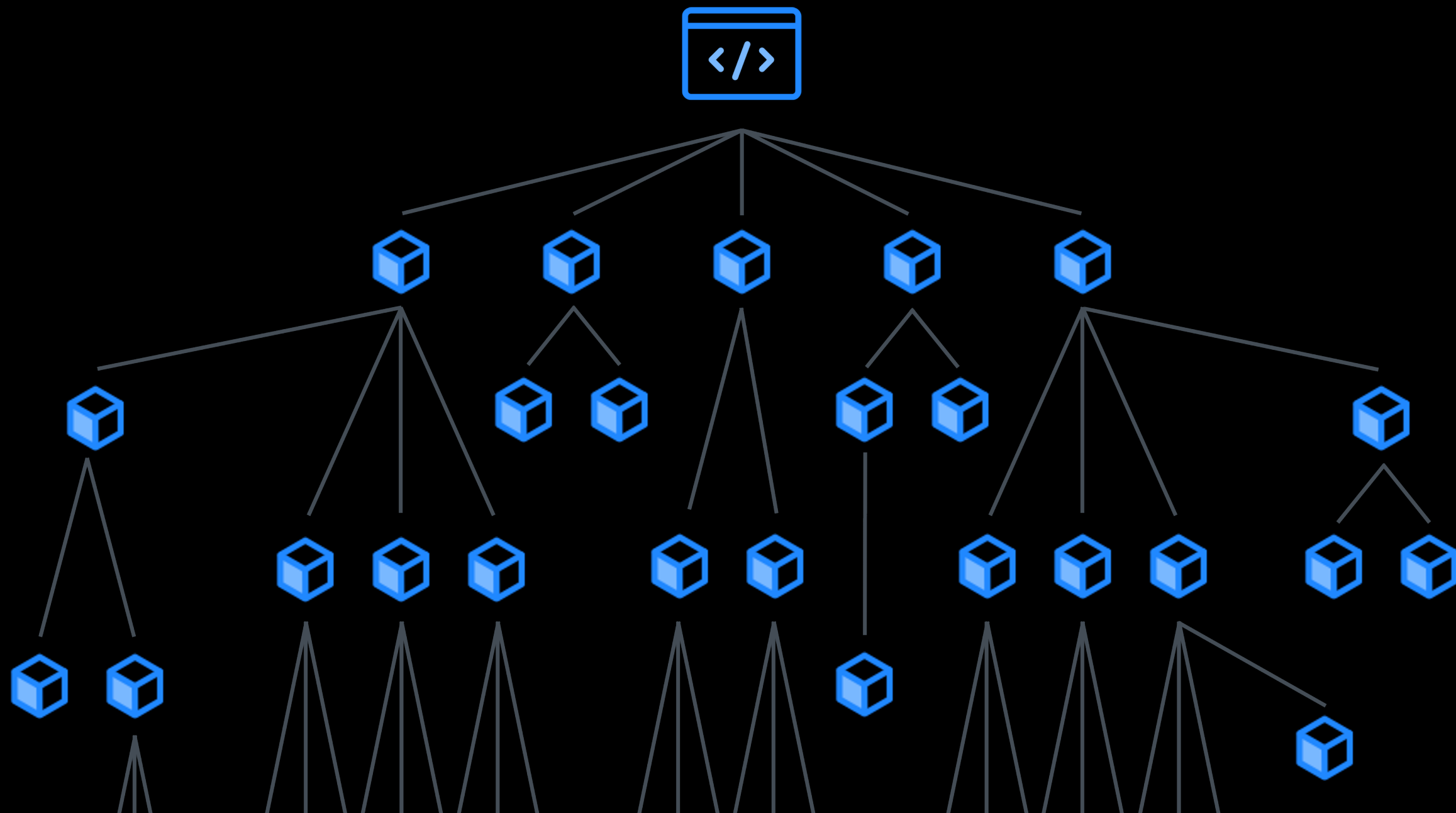
of organizations make extensive use of open source

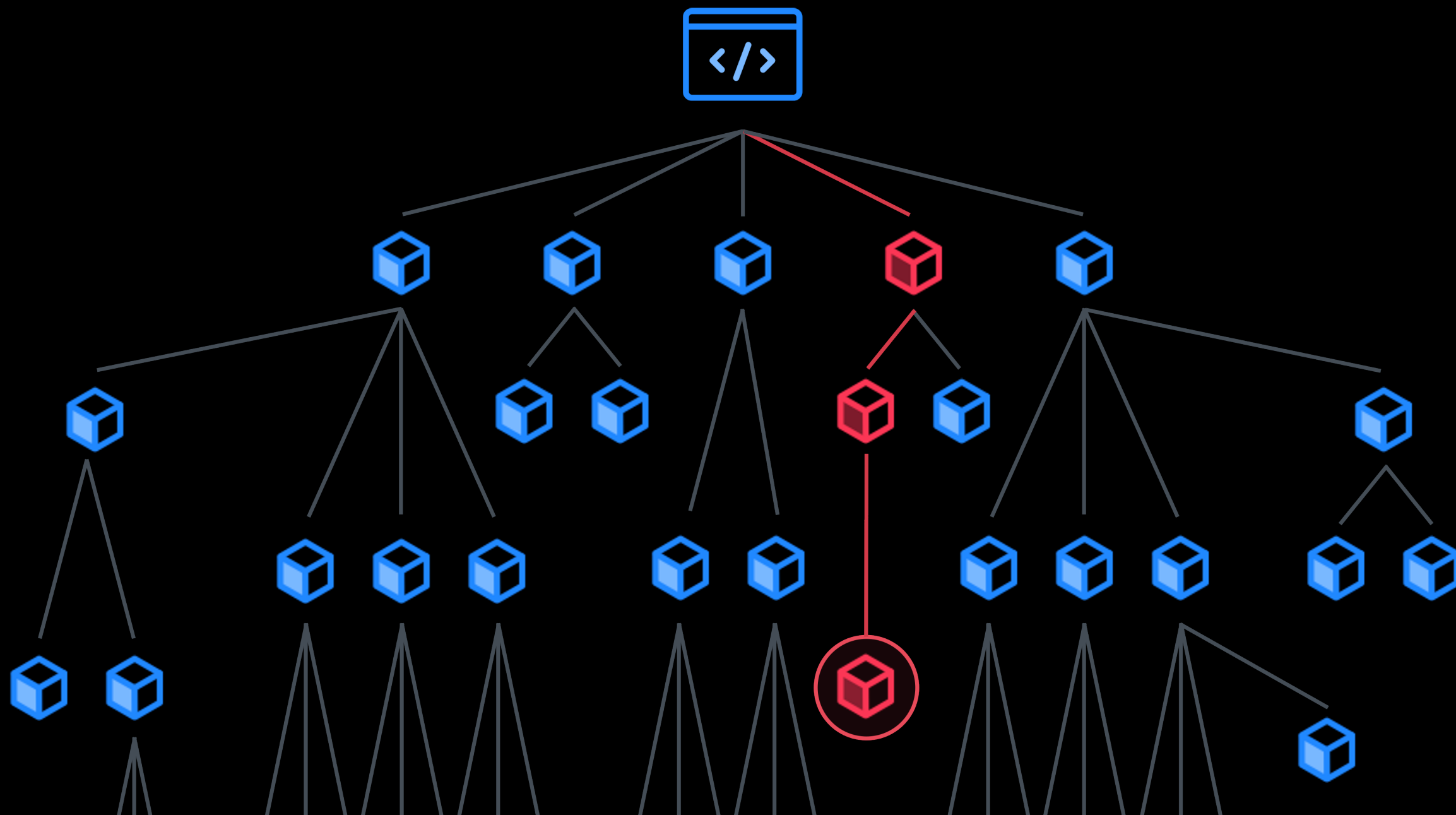


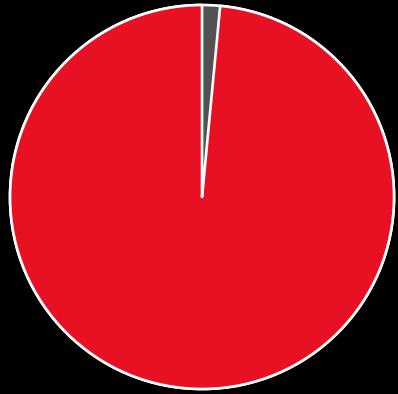
of new application development leverages open source software.



New Application Code







99%

Of the exploited vulnerabilities
were compromised more than a
year after the CVE was published



90% percent of active applications use libraries with a known CVE — 30 percent used a library with a critical CVE. Patching a critical CVE took an average of 34 days.

Automatically upgrade vulnerable dependencies

 This automated pull request fixes a [security vulnerability](#)

moderate severity

Only users with access to Dependabot alerts can see this message. [Learn more about Dependabot security updates](#), [opt out](#), or [give us feedback](#).

 Conversation 0

 Commits 1

 Checks 0

 Files changed 2

+4 -5   



dependabot bot commented on behalf of github on Jun 8

Bumps [hoek](#) from 4.1.1 to 6.1.3.

► Commits

 compatibility unknown

Dependabot will resolve any conflicts with this PR as long as you don't alter it yourself. You can also trigger a rebase manually by commenting `@dependabot rebase`.

► Dependabot commands and options

  Bump hoek from 4.1.1 to 6.1.3 ...

Verified

ec37f56

  dependabot bot added the [dependencies](#) label on Jun 8

Reviewers



Suggestions

 DivineOps

[Request](#)

Still in progress? Convert to draft

Assignees



No one—assign yourself

Labels



[dependencies](#)

Projects



None yet

Milestone

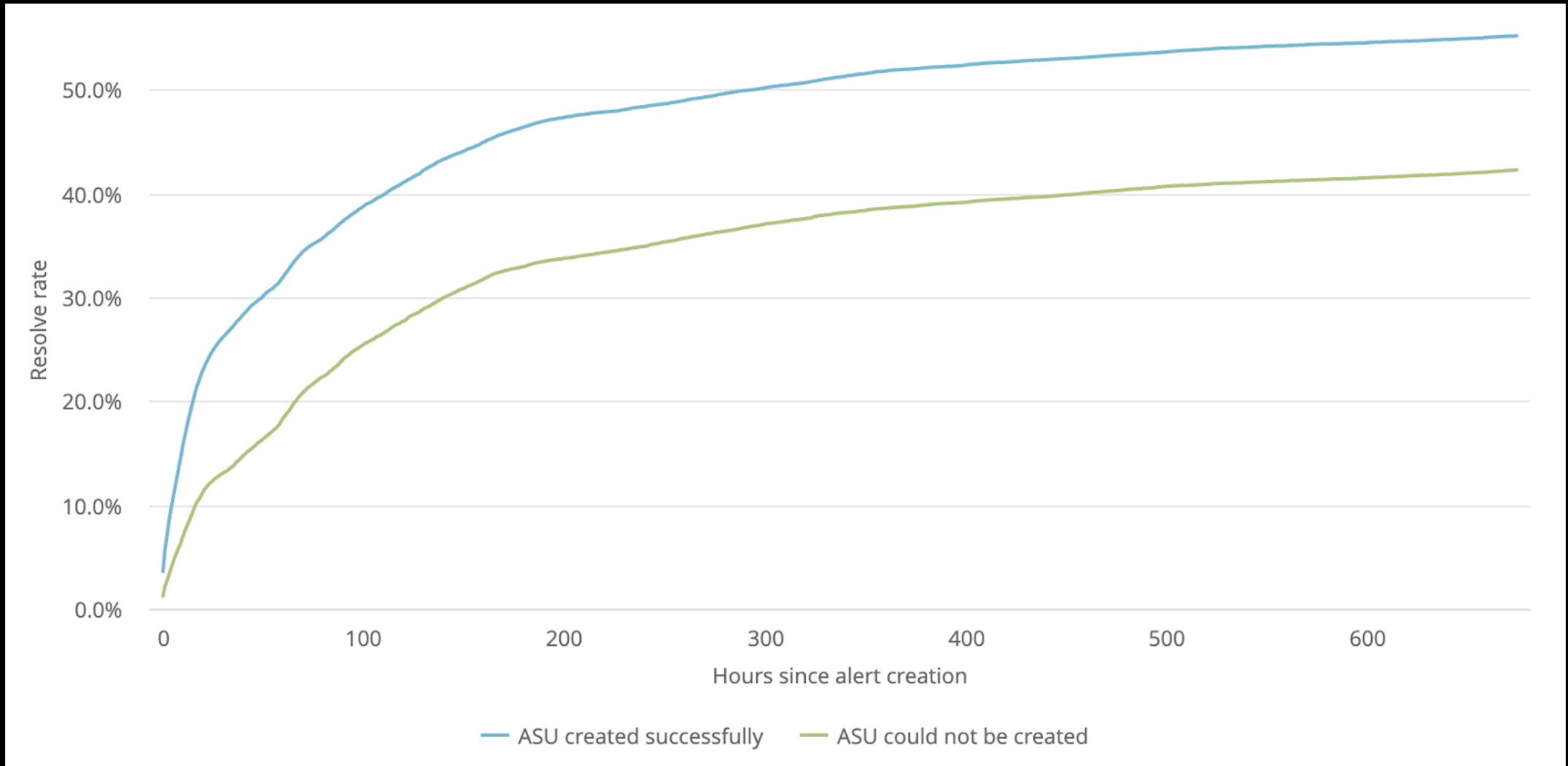


No milestone

90% of my work emails
are dependabot alerts

I'm not sure I would
admit this in public, friend

Dependabot increases the resolve rate and speed



Package Management

- OSS dependencies are scanned for vulnerabilities and kept up to date
- Builds artifacts are managed
- Binary artifacts are accessed via a trusted feed and scanned for vulnerability



Securing you Code

Secret scanning

Code scanning

Database query built from user-controlled sources

4 steps in script.js ▾ ×

Step 1 req.params.category

Source

script.js

```
13
14 // BAD: the category might have SQL special characters in it
15 var query1 = "SELECT ITEM,PRICE FROM PRODUCT WHERE ITEM_CATEGORY='"
16             + req.params.category +
17             "' ORDER BY PRICE";
18
19 pool.query(query1, [], function(err, results) {
```

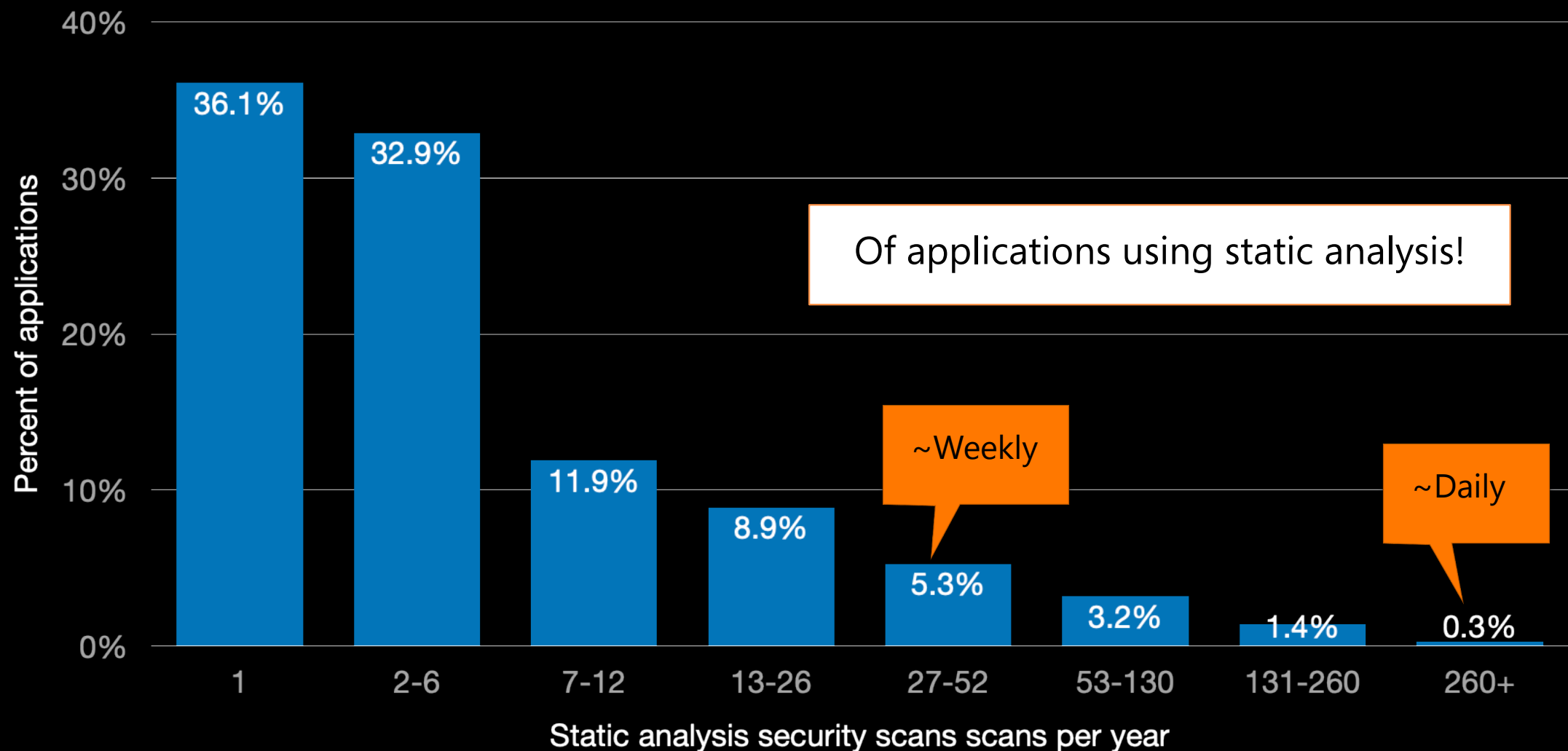
Step 2 "SELECT ... PRICE"

script.js

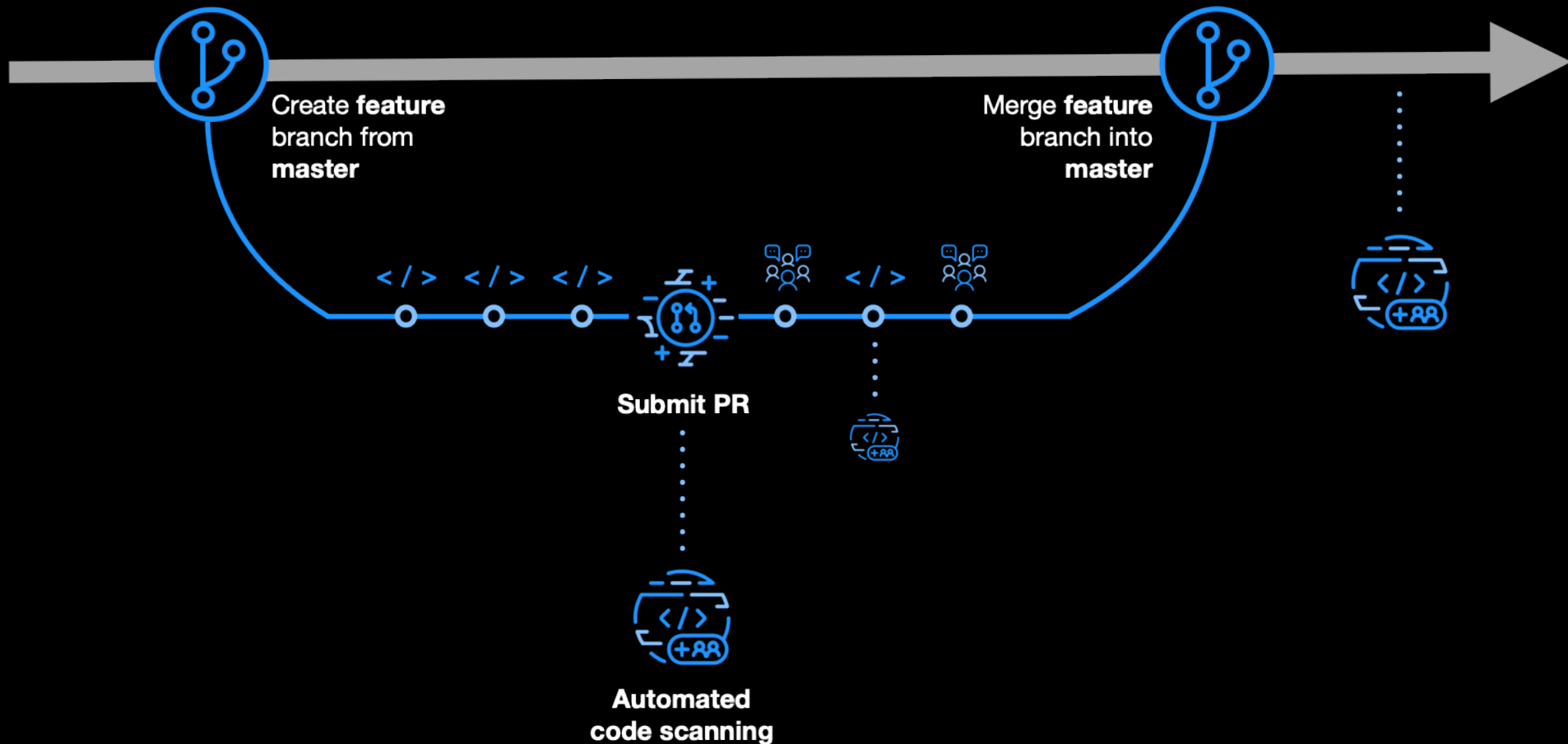
```
12 app.get("/category/:category", function (req, res) {
13
14 // BAD: the category might have SQL special characters in it
15 var query1 = "SELECT ITEM,PRICE FROM PRODUCT WHERE ITEM_CATEGORY='"
16             + req.params.category +
17             "' ORDER BY PRICE";
18
```

Code scanning can help!

Code scanning is still an aspiration



Code scanning is automated code review!



Code scanning



Open

Update script.js #2

Update script.js f21a7ea ▾

× CodeQL

ANNOTATIONS

✗ Check failure on line 30 in script.js



Code scanning

Incorrect suffix check

This suffix check is missing a length comparison to correctly handle lastIndexOf returning -1.

[Show more details](#)

Close ▾

✗ Check failure on line 19 in script.js



Code scanning

Database query built from user-controlled sources

This query depends on [a user-provided value](#).

[Show more details](#)

Show paths

Close ▾

✗ Check failure on line 12 in script.js



Code scanning

Missing rate limiting

This route handler performs [a database access](#), but is not rate-limited.

[Show more details](#)

Automation is not everything



Senior Oops Engineer

@ReinH

I mean yes automation eliminates human errors in the sense that those errors will now be performed by machines

Threat Modeling

Why Threat Model?

A way to identify security issues during design

Developers think about how a product works -
Attackers think about how to abuse a product

Shift the mindset -
Think like an attacker

Threat Model: Pull Request Bypass

① Completion options are shared across users.
② Test it

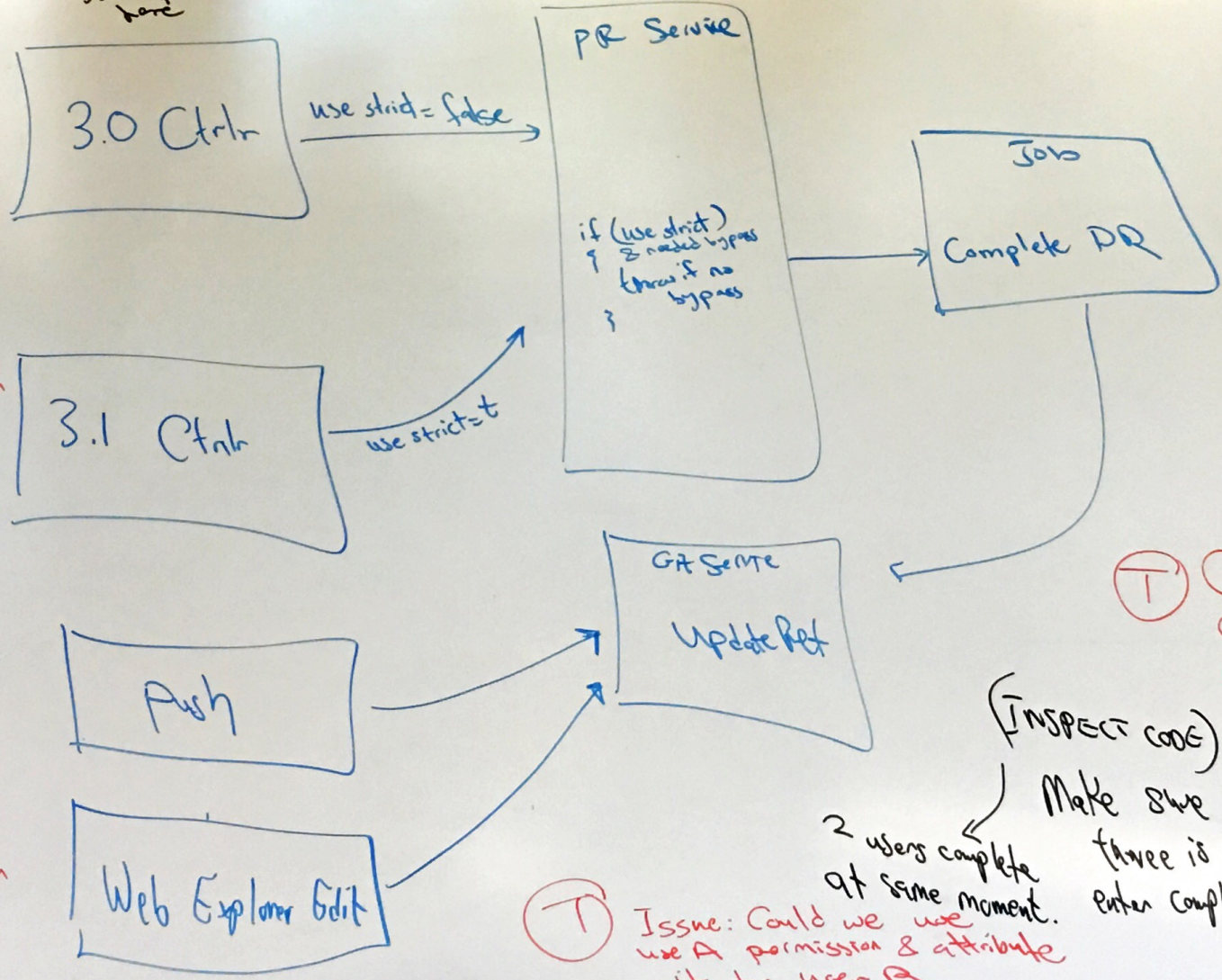
① Elevation - Can someone set the use strict = false & make 3.1 Ctrlr bypass unintentionally.
② almost no damage - Test it
JSAs protect us here

① if:
- have permission
- did not specify bypass
- completed via 3.0 Ctrlr
Does the PR page tell "bypass"
② Yes - test it works

① if:
- User A has permission sets bypass in Completion option
- User B has permission does not want by pass
- User B complete & bypass by accident
② This is a bug - test it

① DOS - What caps do we need?
- message size

Issue #1 - we do not warn if you push from the web



① Can you edit completion options after PR completed

(INSPECT CODE)
Make sure there is only 1 enter completion queue
2 users complete at same moment.

① Issue: Could we use use A permission & attribute it to user B

① Can I push at right time
Skip all v
if push happens
PR is in queue

② Does the job that it is complete the right con

① Do we bypass = t whe you v



SECTOR

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

STATUS
ALPHA
BETA

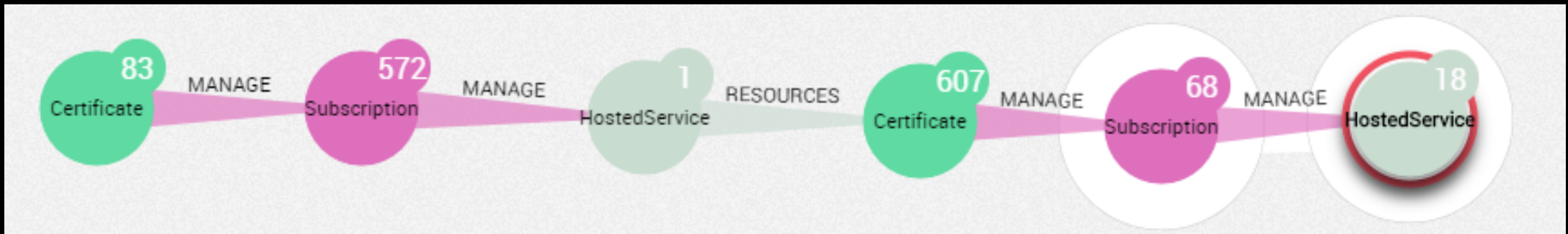
STATUS
ALPHA
BETA



War Games

“Defenders think in lists. Attackers think in graphs. As long as this is true, attackers win”

-- John Lambert (MSTIC)



Security Mindset - Assume Breach

Started with war games to learn attacks and practice response



VS.



- ▶ Initially double-blind test
 - ▶ Over time, eliminated blue team
- Our defenders need to be our defenders

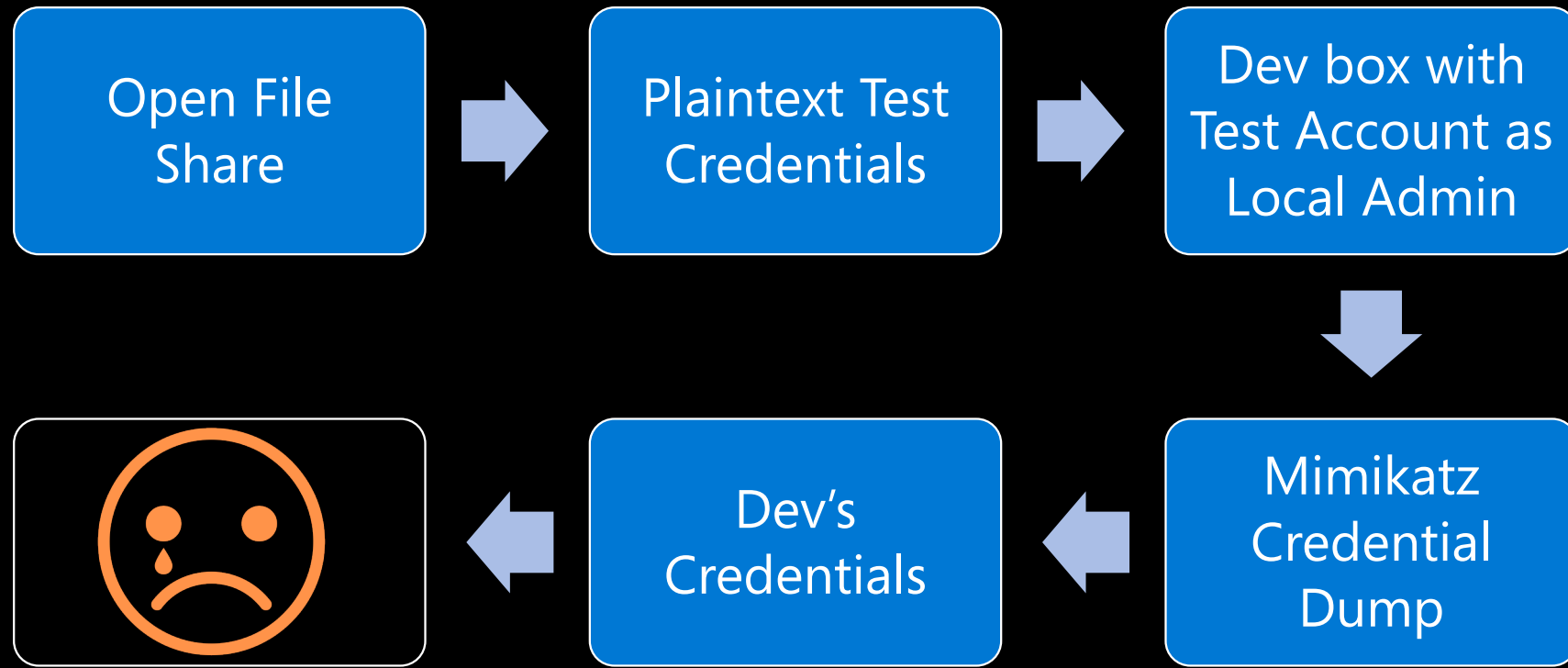
Shifted left to prevent top risks

- ▶ Credential theft
- ▶ Secret leakage
- ▶ OSS vulnerabilities

Block Lateral Movement

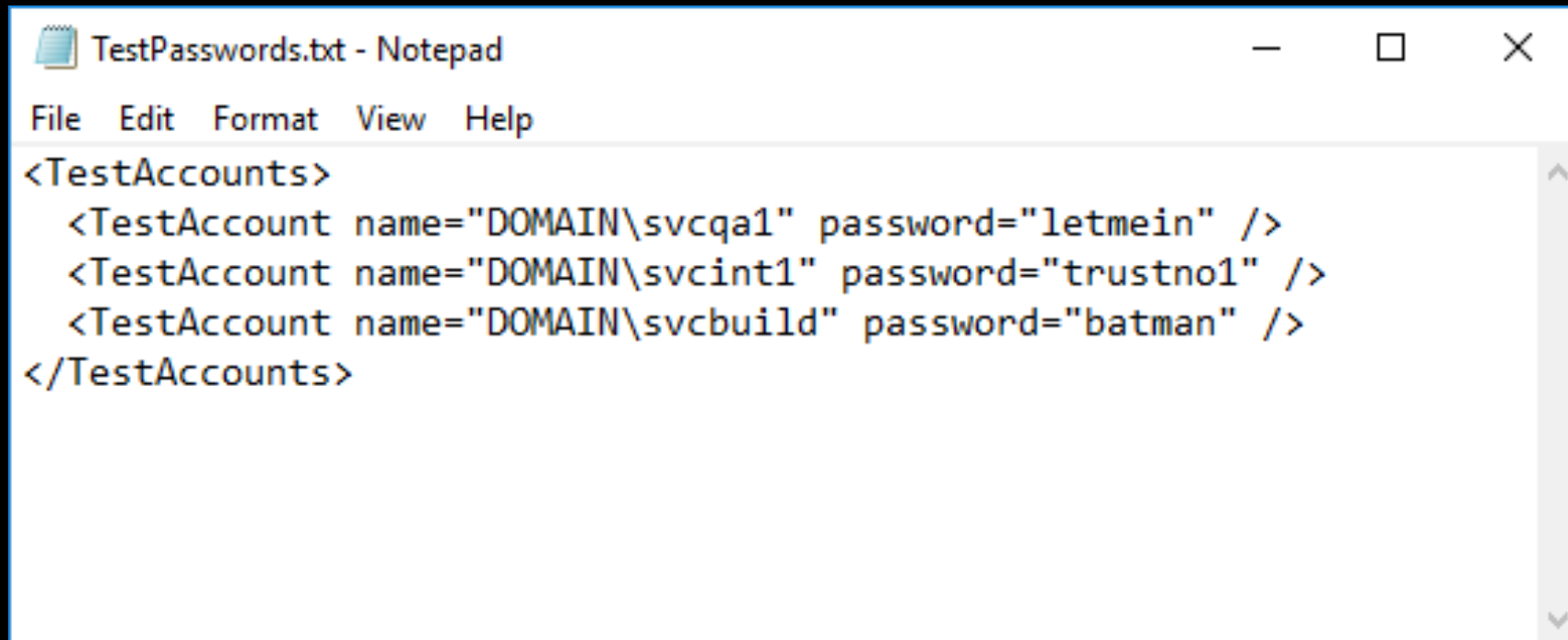
Example: Red Team Attack

What's wrong with this picture?



Another Source of Leak: Credentials in a File

What do plaintext credentials look like?



A screenshot of a Notepad window titled "TestPasswords.txt - Notepad". The window displays XML data containing three test accounts with their names and passwords in plaintext. The XML structure is as follows:

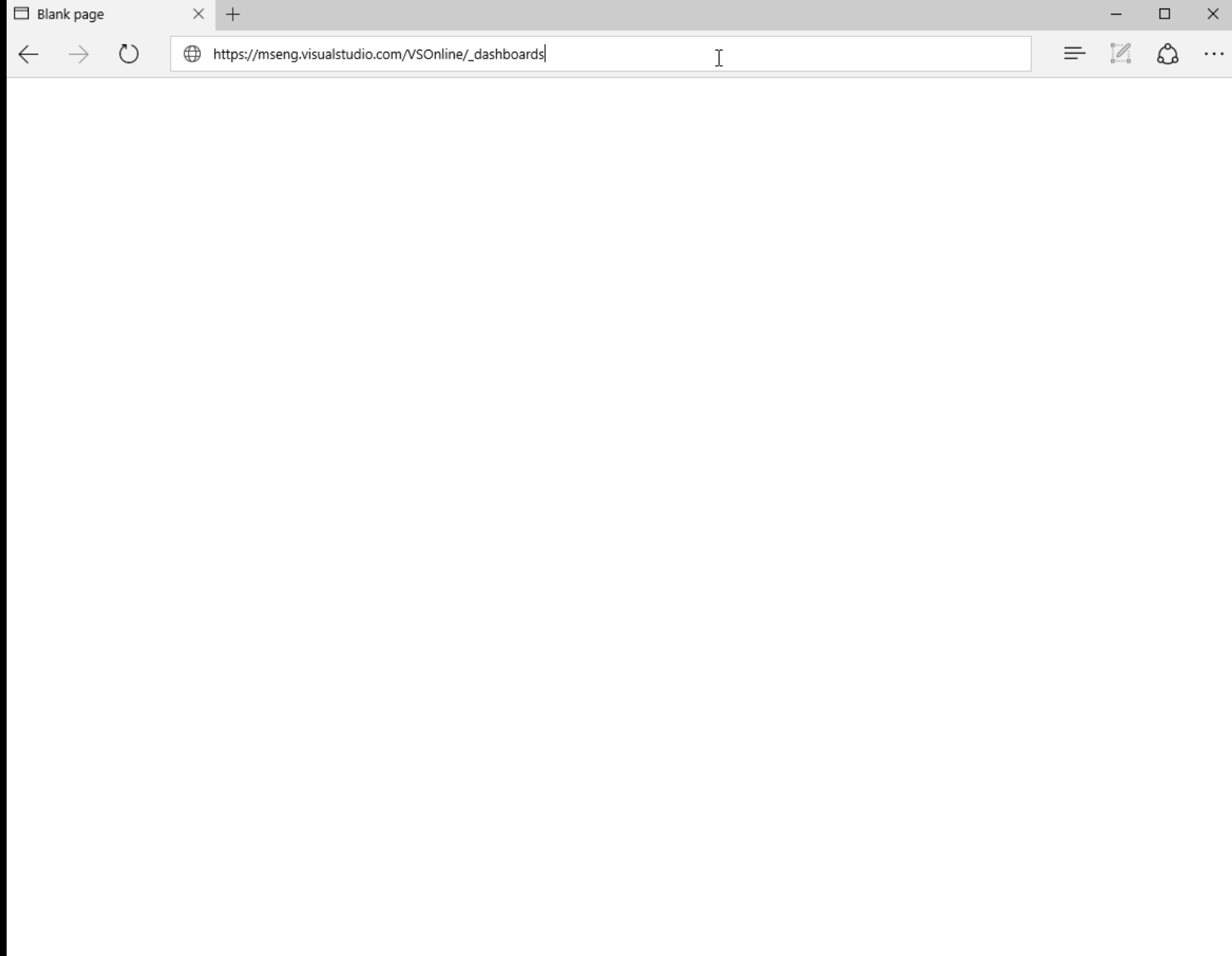
```
<TestAccounts>  
  <TestAccount name="DOMAIN\svcqa1" password="letmein" />  
  <TestAccount name="DOMAIN\svcint1" password="trustno1" />  
  <TestAccount name="DOMAIN\svcbuild" password="batman" />  
</TestAccounts>
```

Every team seems to experience this one at the beginning.

Prove it!

Show

Don't tell



Every time someone viewed the dashboard...

The screenshot shows a web browser window displaying a bug report in the Microsoft Teams interface. The browser's address bar shows the URL `mseng.visualstudio.com/VSONline/_workitems?_a=edit&id=593738&triage=true`. The page header includes 'Team Services / VSONline' and a navigation menu with options like HOME, CODE, WORK, BUILD, TEST, PACKAGE *, RELEASE, and COMPLIANCE *. Below this, there's a sub-menu with Backlogs, Queries, Activities, Estimate, and State Visualizer. The main content area displays the bug details for 'BUG 593738', titled '593738 There's a Persisted XSS Somewhere in VSTS!'. It is currently 'Unassigned' and has 280 comments. A sidebar on the left, labeled 'Query explorer', shows a list of queries, with 'BUG 593738' selected. The bug's state is 'Closed' (indicated by a green dot) and its area is 'VSONline'. The reason is 'Verified' and the iteration is 'VSONline\OneVS\Sprint 102'. At the bottom, there's a list of comments from users: Lori Lamkin, Trevor Gau, Aaron Bjork, Madhu Kavikondala, Aaron Bjork, and Jeff Beehler, all commenting 3 weeks ago. The right side of the bug report shows a '67 of 68' pagination indicator and various action buttons like 'Save', 'Follow', and 'Add Tag'.

Bug 593738: There's a P ×

mseng.visualstudio.com/VSONline/_workitems?_a=edit&id=593738&triage=true

Team Services / VSONline

HOME CODE **WORK** BUILD TEST PACKAGE * RELEASE COMPLIANCE *

Backlogs **Queries** Activities Estimate State Visualizer

> Query explorer

BUG 593738 67 of 68 ↑ ↓

593738 There's a Persisted XSS Somewhere in VSTS!

Unassigned 280 Add Tag Save Follow

State ● Closed Area VSONline

Reason Verified Iteration VSONline\OneVS\Sprint 102 Bug Customer Ask mode Visualizations

LL **Lori Lamkin** commented 3 weeks ago +1!

Trevor Gau commented 3 weeks ago +1!

Aaron Bjork commented 3 weeks ago +1!

MK **Madhu Kavikondala** commented 3 weeks ago +1!

Aaron Bjork commented 3 weeks ago +1!

Jeff Beehler commented 3 weeks ago

Protect Against Lateral Movement

- Assume layers before yours will be breached
- Never assume an internal service is unimportant
- Never assume a service is secure because it is internal

No Standing Permissions

- No standing access to production
- JIT (just in time) tokens only
- Secure Workstations only
- Infrastructure refresh

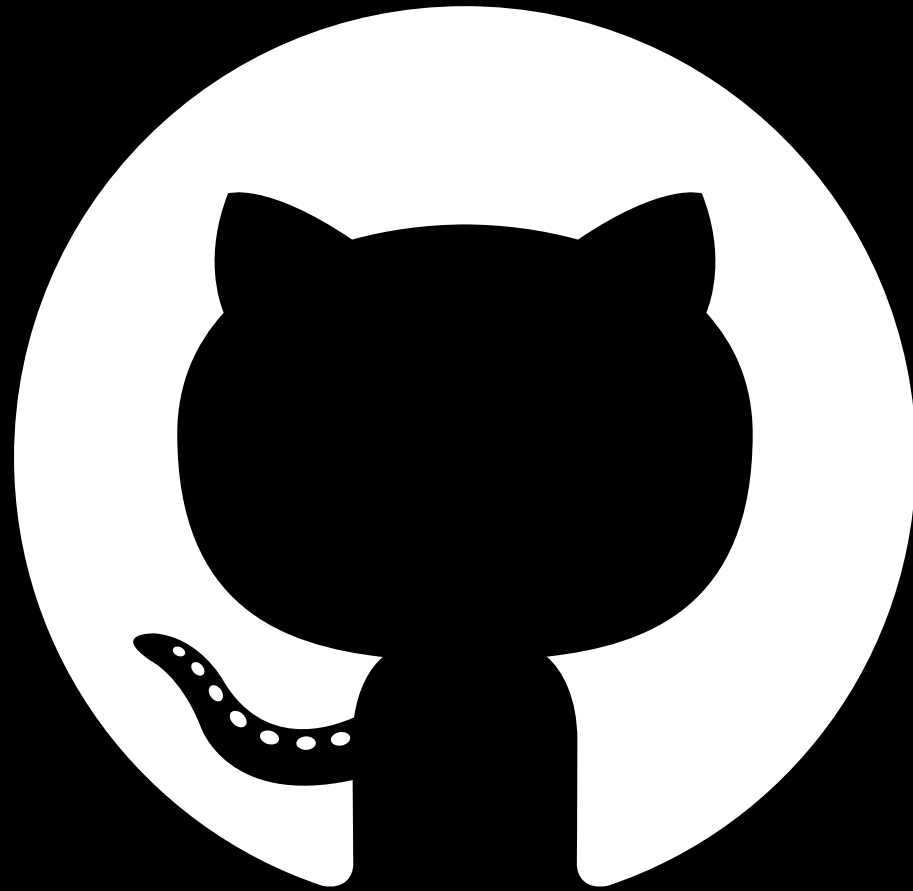
Internal CTFs

Capture the Flag events



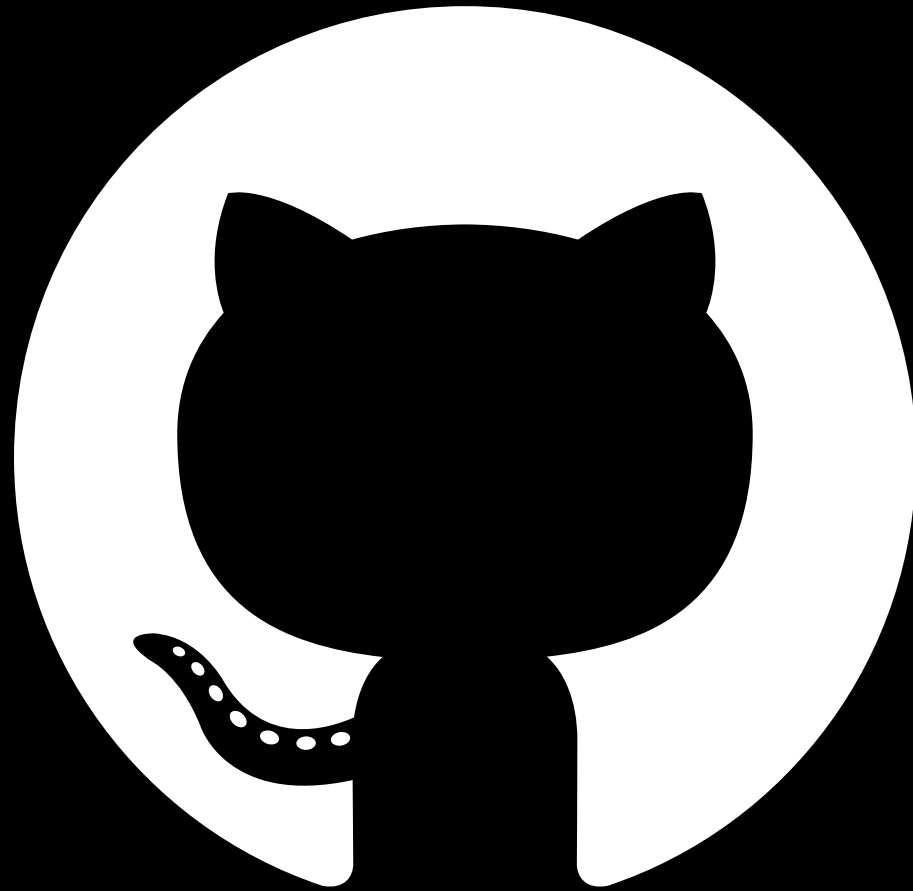
Stay safe!

Thank you!



@DivineOps

Thank you!



@DivineOps