OOSC · SOFTWARE SUPPLY-CHAIN SECURITY

# Can You Trust the Commit?

What the development process reveals when the code reveals nothing

Shreya Shree

Two papers · ICSE-SEIP 2021 · CODASPY 2023

THE CASE

# The PHP backdoor was hard to identify from code alone — but the development process held clues

IDENTITY
impersonated

SIGNING KEY
compromised

MESSAGE
“typo fix”

ATTACK
spread across two commits

**Pushed at 06:00 — the previous 20 commits: all after 18:00**

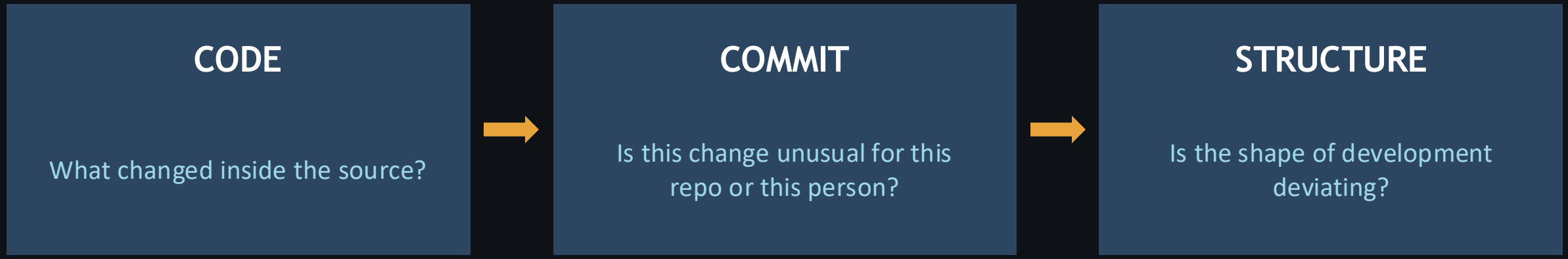
Identity failed. Static analysis would not flag this code. The development history still carried a signal.

**Can the development process become a security signal?**

php/php-src commit c730aa2 · March 2021

THE FRAME

# An attack leaves different traces depending on where you look




This is not about replacing code analysis. Two papers, published two years apart, explore different units of observation.

**The unit of observation is the design decision.**

Gonzalez et al. 2021 · Ganz et al. 2023

# Anomalicious never reads the code: it uses only commit logs and repository metadata

- Outlier change properties
- Sensitive files
- File history
- Rejected pull requests
- Contributor trust

→

**FLAGGED?**
**only when SEVERAL factors violate rules at once**

No single factor means malicious. The model is rule-based, so every alert explains itself to the maintainer.

# Anomaly is contextual: the same commit is an outlier or unremarkable

The detector measures two baselines:

the repository's mean change properties, and each contributor's own.

Default threshold: two standard deviations from the mean — and configurable.

## REPOSITORY

Typical ~20 lines
This commit 2,000
→ OUTLIER

## CONTRIBUTOR

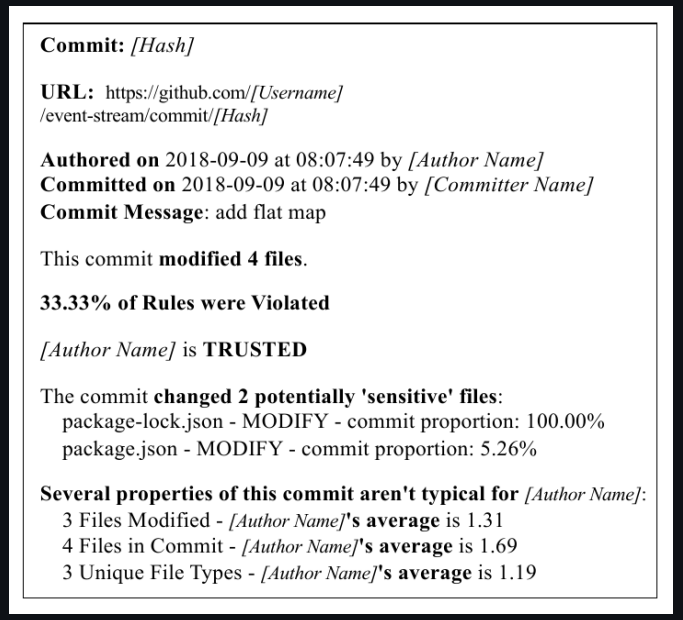
Typical ~1,500 lines
This commit 2,000
→ UNREMARKABLE

*Illustrative figures*

**A large commit can be strange for the repository and entirely normal for its author.**

Source: Anomalicious §III-A — repository-level and contributor-level means

# In event-stream the trust signal stayed silent — two other factors fired



**Commit:** *[Hash]*

**URL:** https://github.com/*[Username]*
/event-stream/commit/*[Hash]*

**Authored on** 2018-09-09 at 08:07:49 by *[Author Name]*
**Committed on** 2018-09-09 at 08:07:49 by *[Committer Name]*
**Commit Message**: add flat map

This commit **modified 4 files**.

**33.33% of Rules were Violated**

*[Author Name]* is **TRUSTED**

The commit **changed 2 potentially 'sensitive' files**:
package-lock.json - MODIFY - commit proportion: 100.00%
package.json - MODIFY - commit proportion: 5.26%

**Several properties of this commit aren't typical for** *[Author Name]*:
3 Files Modified - *[Author Name]***'s average** is 1.31
4 Files in Commit - *[Author Name]***'s average** is 1.69
3 Unique File Types - *[Author Name]***'s average** is 1.19

**WHY IT WAS STILL CAUGHT**

"Author is TRUSTED" — the trust factor never fired. The attacker was a real contributor with elevated privileges.

What fired instead — together:
SENSITIVE FILES package.json
OUTLIER PROPERTIES 3 files vs an average of 1.31
→ 33.33% of rules violated

**One factor alone would have missed it. This is what "no single factor" means in practice.**

Source: Gonzalez et al. 2021, Figure 3 — example commit report from event-stream

PART I · THE EVIDENCE

# Behavioral signals found the malicious commit in 8 of 15 repositories

**15**

malware-infected repositories

**8 / 15**

repositories where the malicious commit was found — 53.33%

**<1%**

of commits flagged — but in a separate 100-repository study

**Seven were missed — and that is the interesting half.**

Source: Gonzalez et al. 2021, Table III (15 malware-infected repos) and §V-A (100 popular repos)

# Four ways a commit-level detector loses sight of the attack

## 1 · ORDINARY FILES

The change never touches a build or config file.

## 2 · TRUSTED CONTRIBUTOR

Of the seven commits missed: “all authors were trusted.”

## 3 · MULTIPLE COMMITS

No single commit looks suspicious on its own.

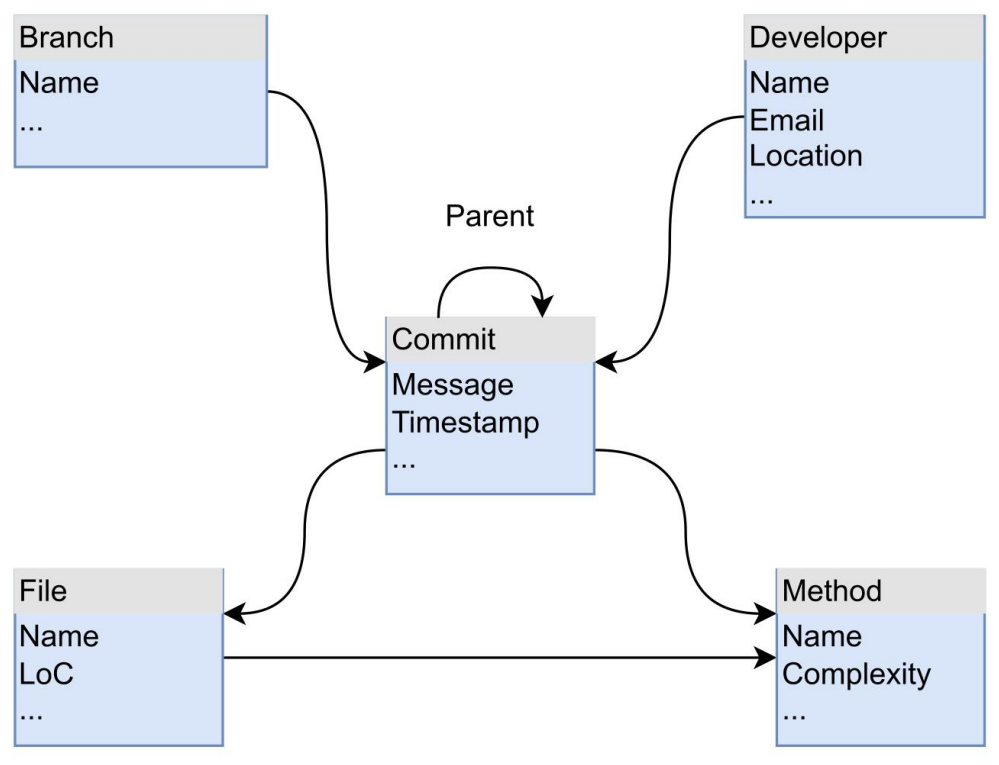
## 4 · ADAPTIVE ATTACKER

Paper two models an attacker who learns the detector and evades it.

**A commit may be too small a unit of observation.**

Evidence: Gonzalez et al. 2021 §V-B3 (undetected commits) · Ganz et al. 2023 §5.3 Robustness against Adversaries

# The second paper moves the unit of observation to the whole repository




Commits, developers, files, methods and branches — one heterogeneous graph, with time on the edges.

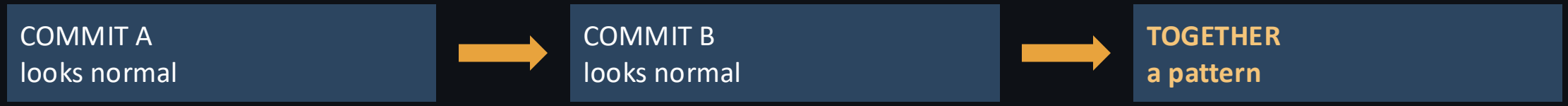
Graph neural networks learn normal collaboration; a one-class learner flags what deviates.

**Not: is this commit unusual. But: is the structure of development deviating?**

Figure: Ganz et al. 2023, Figure 2 — Model of a collaboration graph · SAP · Bielefeld · TU Berlin

# Some attacks do not exist inside any one commit — only across them

> “Hand-written rules are limited by the prior knowledge of the practitioners who produce them.
>
> Ganz et al., CODASPY 2023 — on the rule-based approach




**Paper one encodes suspicious behavior as rules. Paper two learns a representation of normal behavior.**

Verbatim — CODASPY 2023, §2.1 “Large variance”, citing Gonzalez et al.

# Head to head on the same 19 real backdoors: 15 found versus 5

**5 / 19**

Anomalicious, as reimplemented by paper two

**15 / 19**

Collaboration graphs — the 78.9% headline

**— BUT DO NOT COMPARE 53.33% WITH 78.9% —**

Those two headline rates come from different corpora and different targets. This table is the valid comparison: one testing corpus of 19 repositories, 6,656 commits, both detectors run on it. Note the honest caveat — paper two's false-positive rate is lower per commit (0.70% vs 3.96%) but HIGHER per project (10.75% vs 9.77%).

Source: Ganz et al. 2023, Table 3 — detection performance on the testing corpus

THE SYNTHESIS

# What you can see depends on what you choose to observe

| VIEW | WHAT IT REVEALS | WHAT IT MISSES |
|---|---|---|
| **Code** | What changed | Context and intent |
| **Commit + history** | Is this unusual for this repo or person | Attacks spread across commits |
| **Relationships over time** | Is collaboration structure deviating | An attacker whose behavior matches established patterns |

## AND THE BOUNDARY IS NOT CLEAN

Paper two's graph carries code-derived metrics — maintainability deltas, token counts, cyclomatic complexity. Even the process view reaches back toward the code.

Anomaly is not maliciousness.

# If an attacker knows what “normal” looks like, can abnormality still be a useful signal?

**Yes — but never alone. Behavioral signals can help decide where a human should look.**

REFERENCES

# Both papers, the released code, and the incident commit

## PAPER ONE

Gonzalez, Zimmermann, Godefroid, Schäfer
Anomalicious: Automated Detection of Anomalous and Potentially Malicious Commits on GitHub
ICSE-SEIP 2021 · arXiv:2103.03846

## PAPER TWO

Ganz, Ashraf, Härterich, Rieck
Detecting Backdoors in Collaboration Graphs of Software Repositories
CODASPY 2023 · doi:10.1145/3577923.3583657

## CODE AND INCIDENT

Anomalicious had no public source, so paper two reimplemented its rules and released it
Reproduction: SAP-archive/security-research-commit-anomaly-detection
PHP backdoor: github.com/php/php-src commit c730aa2