

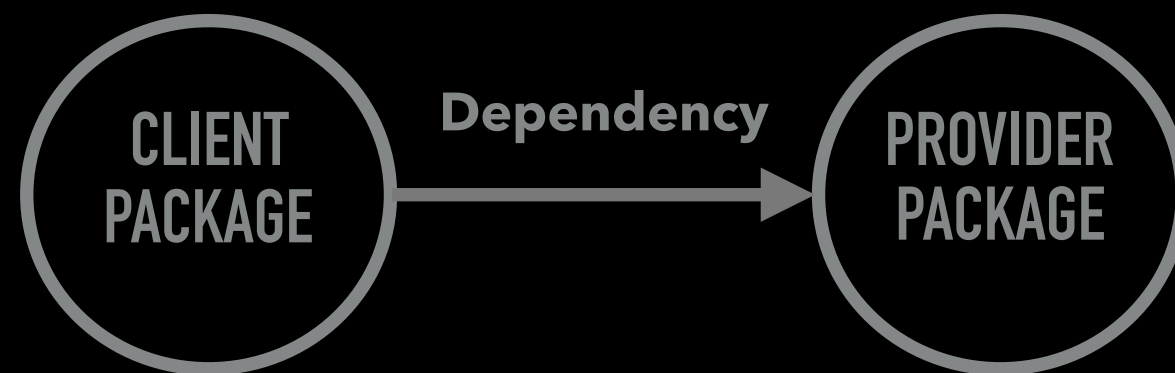
PRESENTED BY **FILIBE COGO**

# MINING DATA FROM DEPENDENCIES IN SOFTWARE ECOSYSTEMS



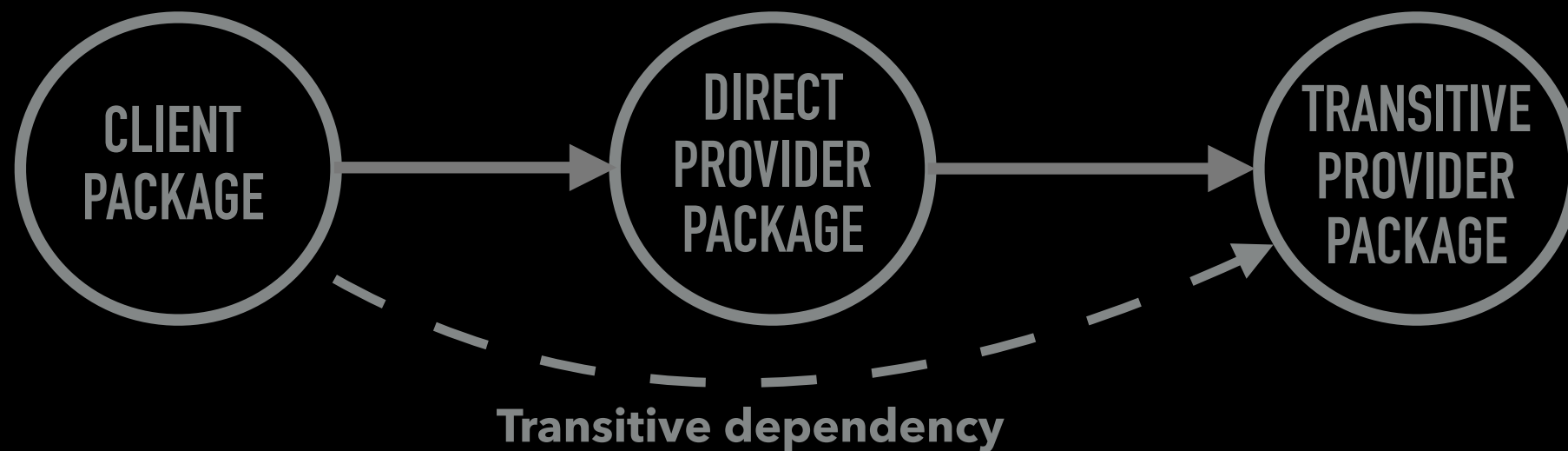
# SOFTWARE ECOSYSTEMS ENABLE SOFTWARE REUSE

- **Client** packages can **reuse** features of a **provider** package



# SOFTWARE ECOSYSTEMS ENABLE SOFTWARE REUSE

- **Client** packages can **reuse** features of a **provider** package



# EVERY MODERN PROGRAMMING LANGUAGE HAS AN ASSOCIATED PACKAGE MANAGER



Rust Cargo

38K packages



CRAN

15K packages



PHP Packagist

261K packages



PyPi

226K packages



Javascript

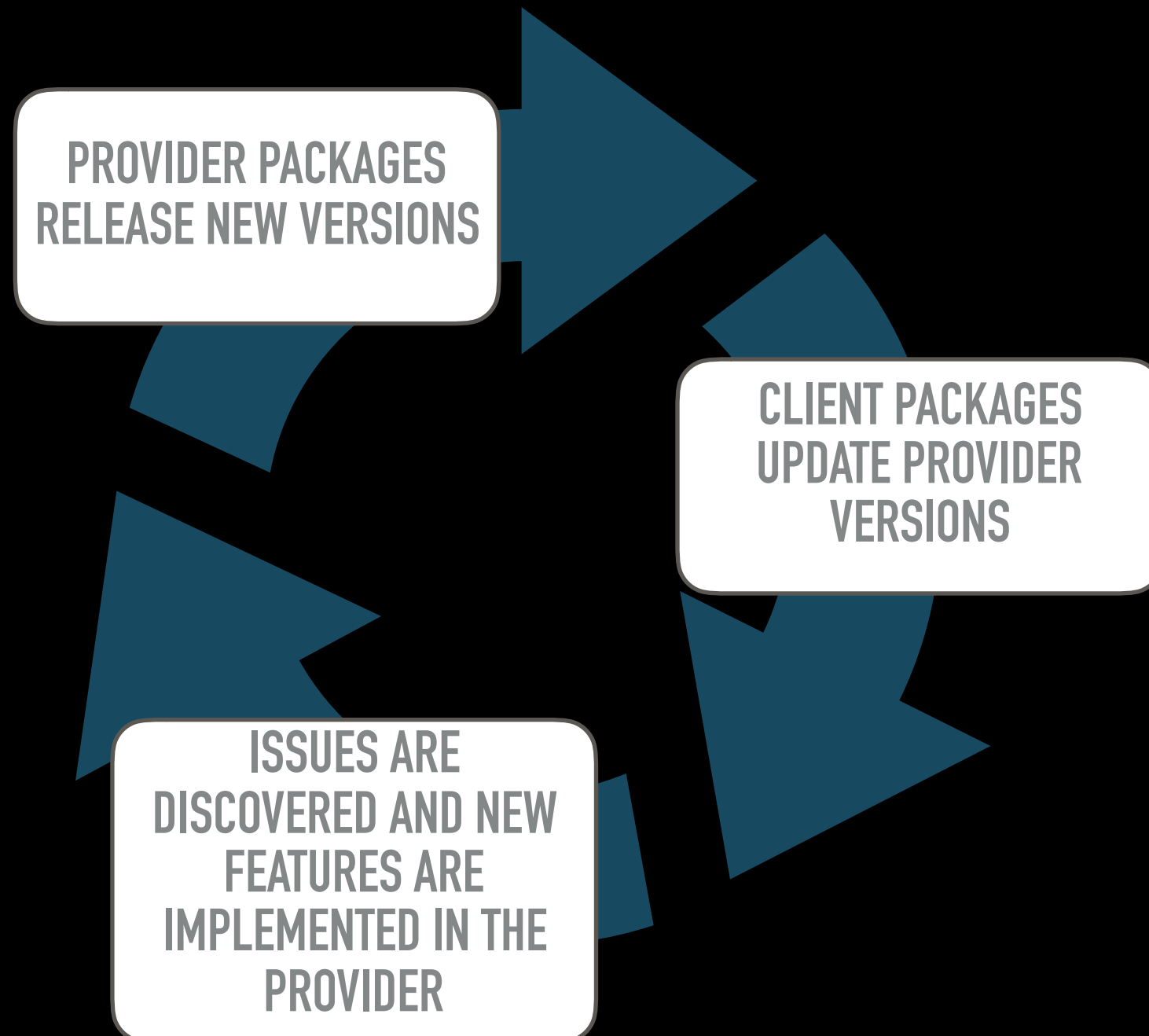
1.3M packages



Rubygems

158K packages

# DEPENDENCY MANAGEMENT IS CRUCIAL TO MAINTAIN WELL FUNCTIONING PACKAGES



*"Dependency management - the management of networks of libraries, packages, and dependencies that we don't control - is one of the least understood and most challenging problems in software engineering."*

–SOFTWARE ENGINEERING AT GOOGLE: LESSONS  
LEARNED FROM PROGRAMMING OVER TIME

# DEPENDENCY MANAGEMENT HAVE A LARGE IMPACT ON DAILY DEVELOPERS' ACTIVITIES

NPM ERR!

Software

What happens when tl  
killing someone with a

What will be the fate of an one

By Thom

{\* SOFTWARE \*}

**NBD: A popular HTTP-fetching npm  
code library used by 48,000 other  
modules retires no more updates**

eeek goes to prison for

WIRED

BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY

LILY HAY NEWMAN

SECURITY 09.14.2017 01:27 PM

## Equifax Officially Has No Excuse

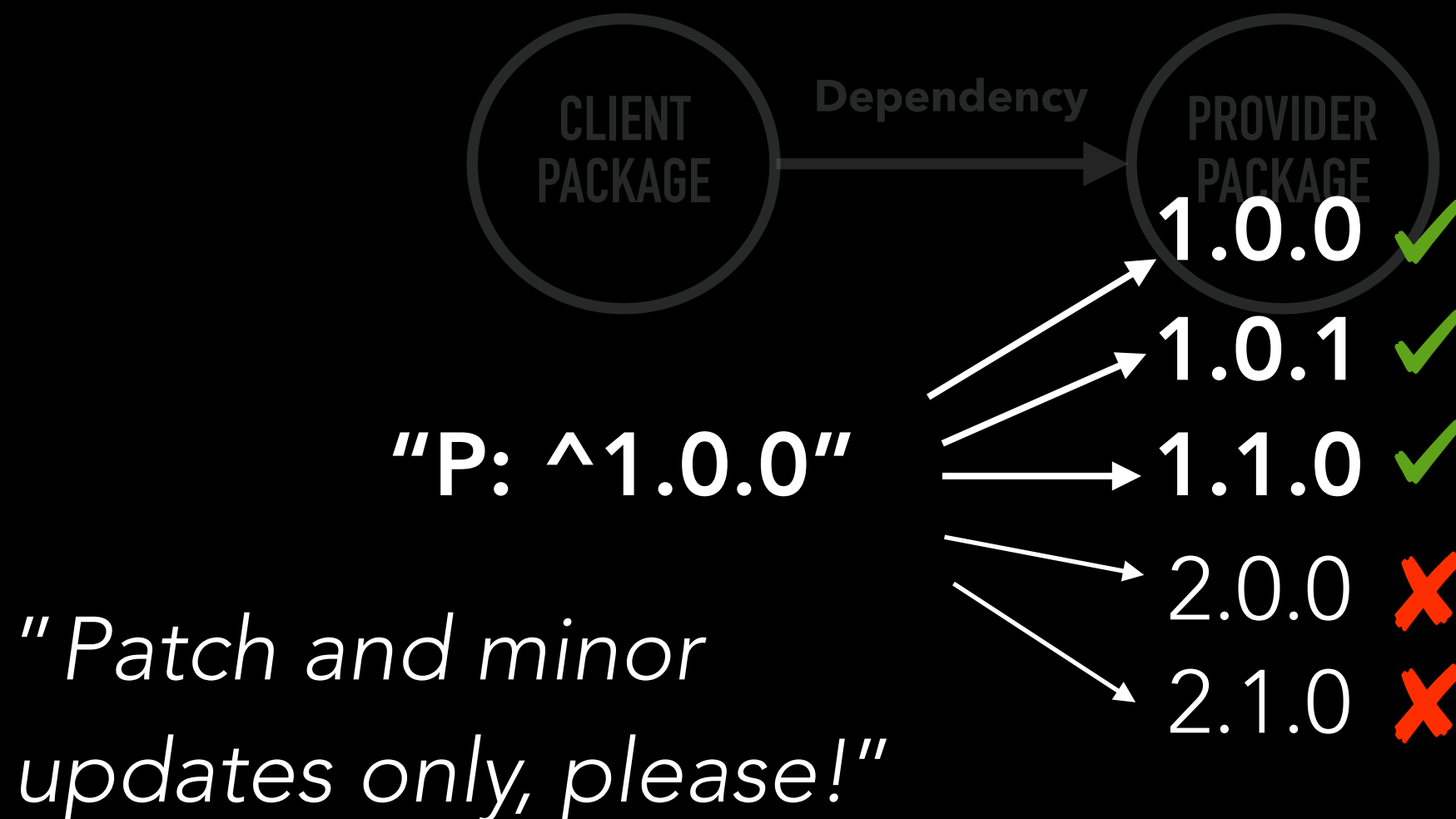
A patch that would have prevented the devastating Equifax breach had been available for months.



```
9 }  
10 return str;  
11 }
```

# PACKAGE MANAGERS OFFER BASIC SUPPORT FOR DEPENDENCY MANAGEMENT

- Mechanism of **automatic provider updates**

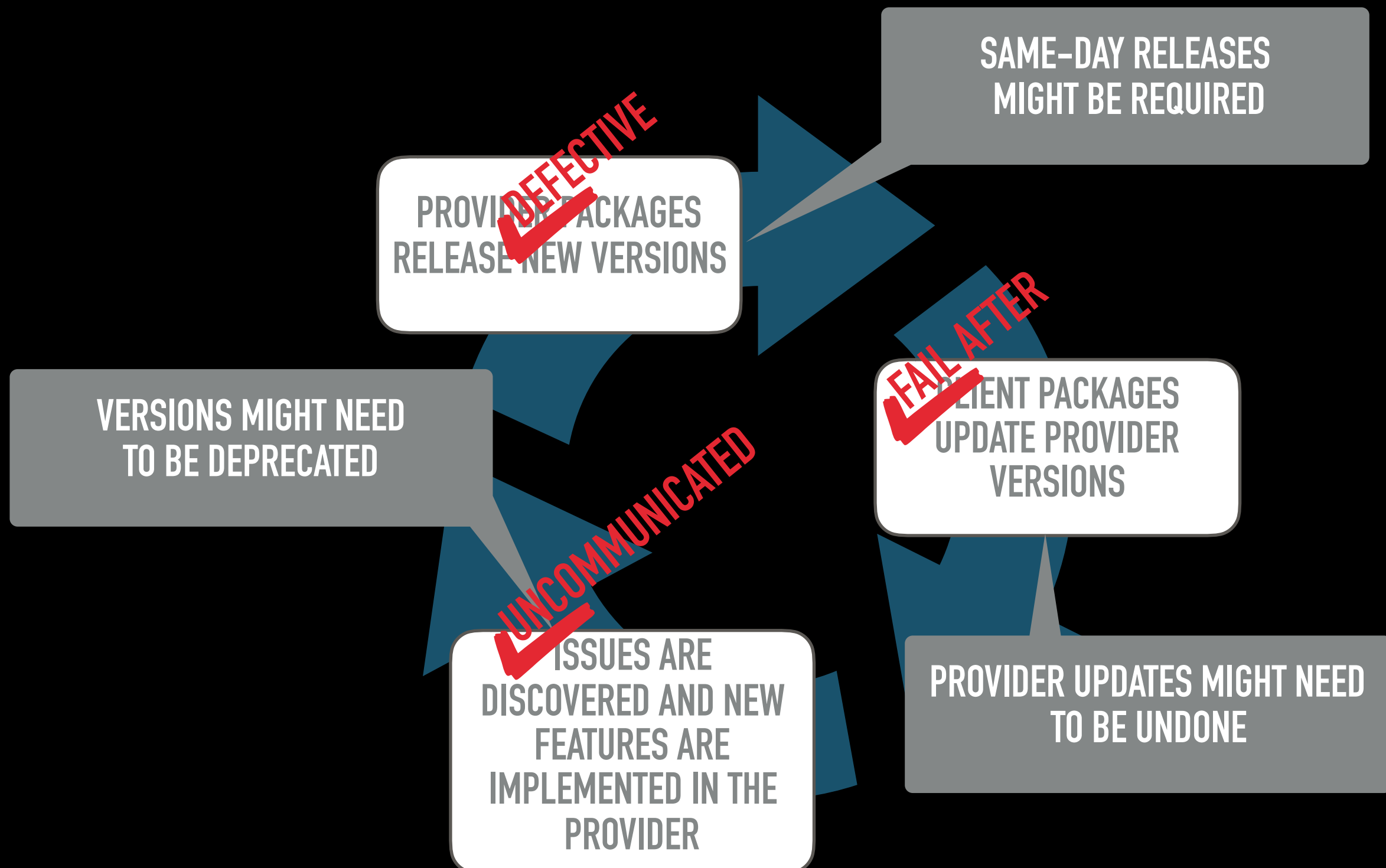




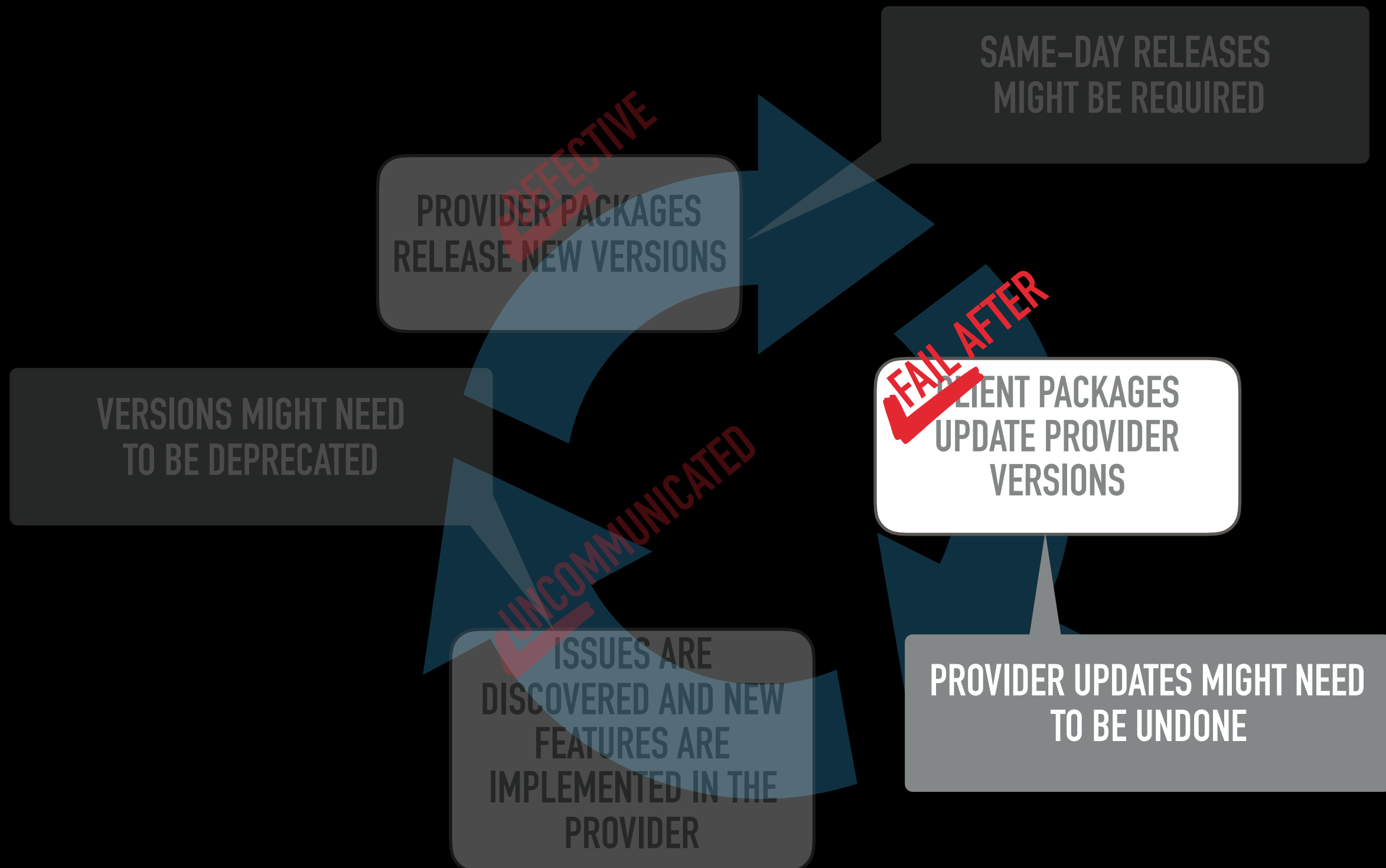
# MANY PROBLEMS ARE FACED WHEN MANAGING DEPENDENCIES

- Monitoring new provider versions
- Prioritizing updates (e.g., security vulnerabilities)
- Shielding against breaking changes from provider packages
- Reducing bloated and dead code
- ...

# PRIOR RESEARCH AT SAIL EXPLORED DIFFERENT ISSUES REGARDING DEPENDENCY MANAGEMENT



# DOWNGRADES ARE A SIGN OF PROBLEMATIC DEPENDENCY UPDATES

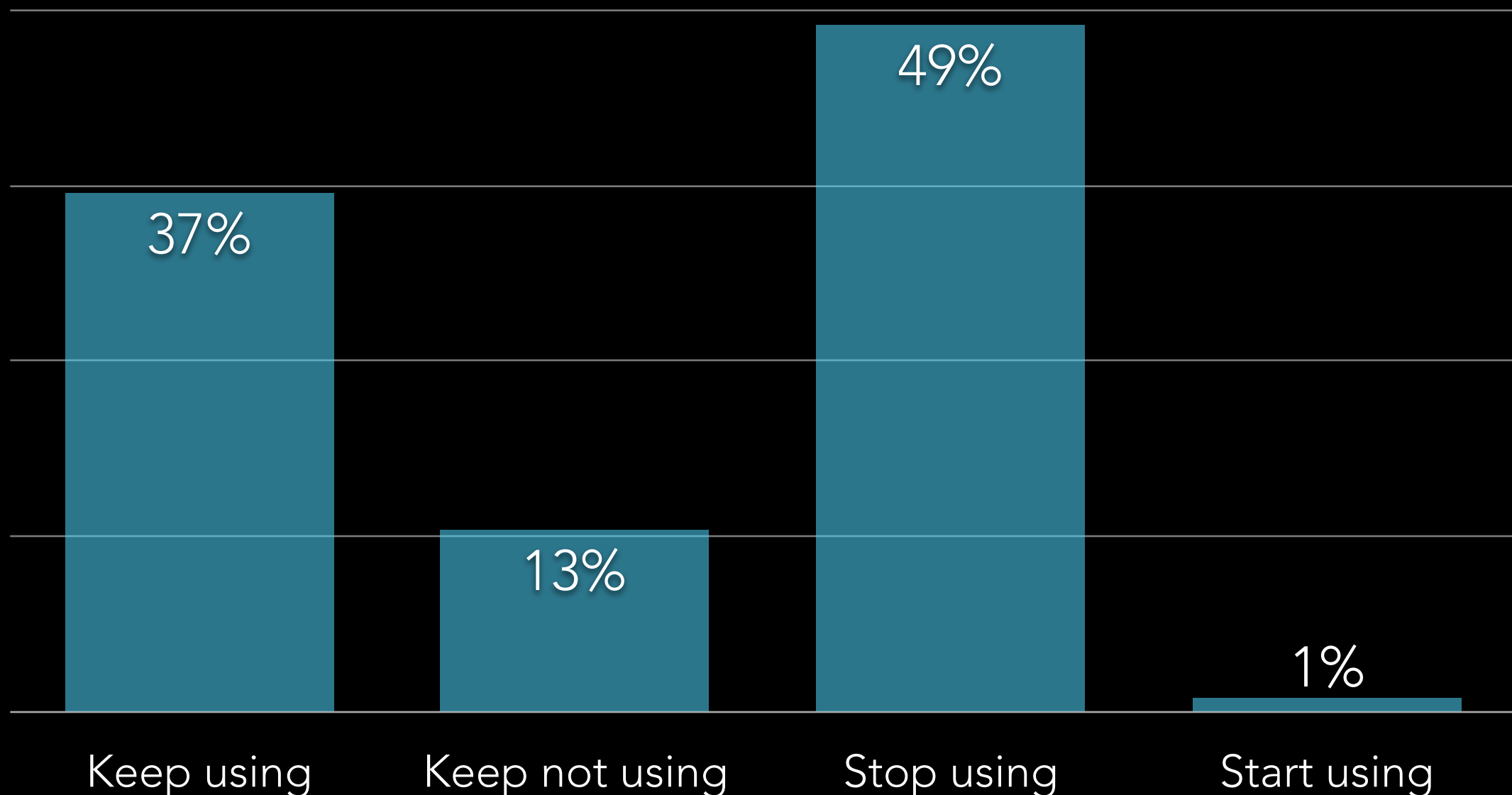


# DOWNGRADES CAN BE EITHER REACTIVE OF PREVENTIVE

- **Reaction** to a failed update
  - **Defect** *"The most recent 0.1.x (0.1.15) broke the build"*
  - **Feature change** *"Reverting mysql to 2.1.1 (...) Unfortunately mysql has changed the way it handles the charset setting (...)"*
  - **Incompatibility** *"reverted jquery version to 2 for jquery-ui compatability [sic]"*
- **Prevention** from a potential, future failed update

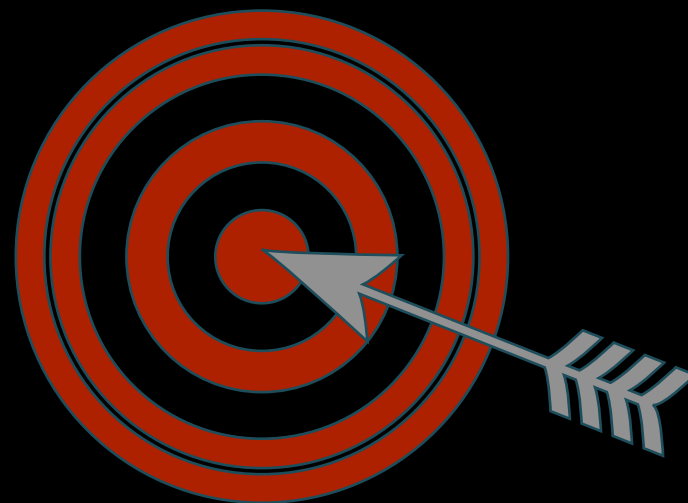
# DOWNGRADES ARE ASSOCIATED WITH THE DEACTIVATION OF AUTOMATIC UPDATES

USAGE OF THE AUTOMATIC UPDATE MECHANISM AFTER DOWNGRADING

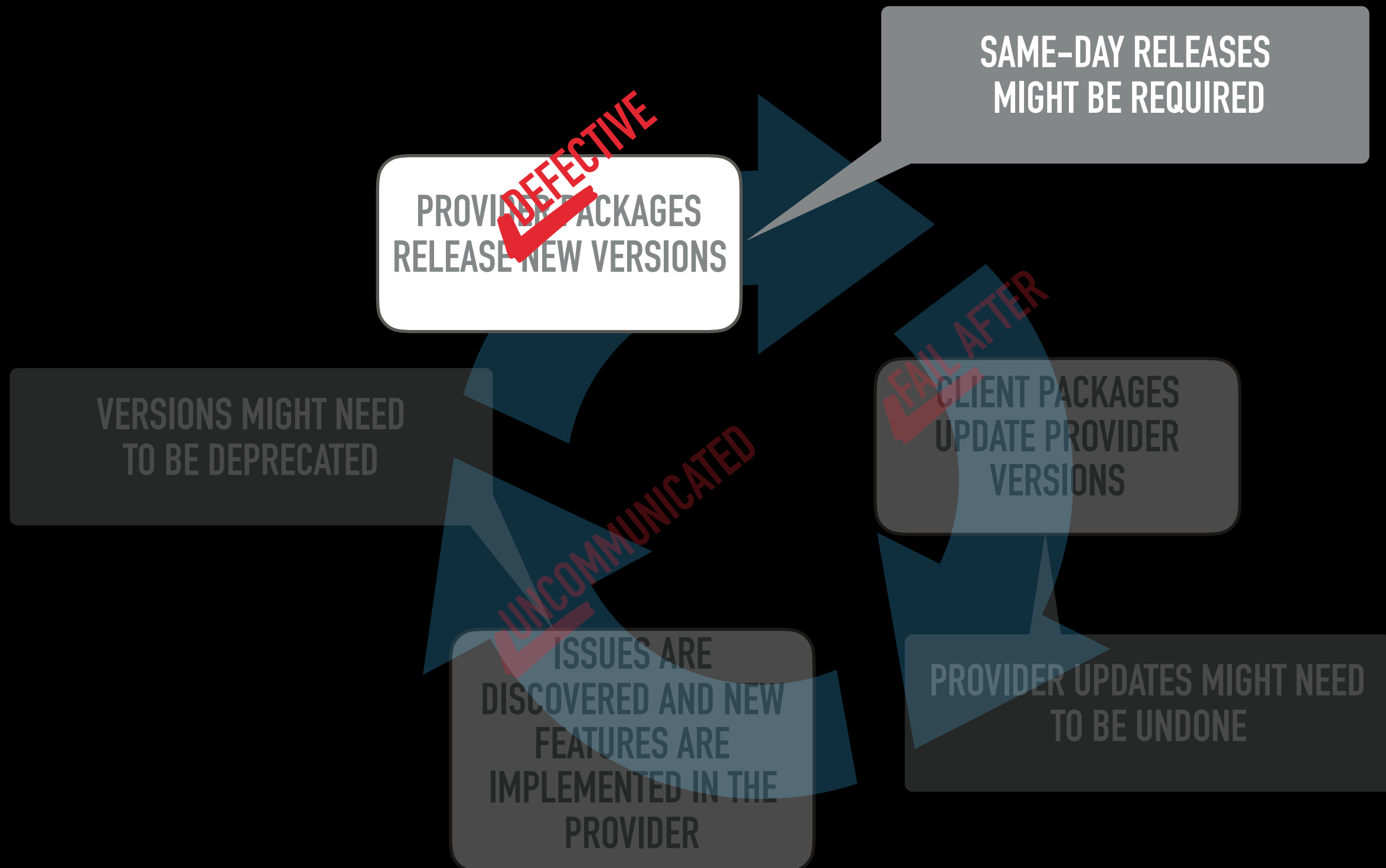


USUALLY TRUST IS LOST ON A SPECIFIC PROVIDER,  
NOT ON THE WHOLE AUTOMATIC UPDATE MECHANISM

In 75% of the client releases that have at least one  
downgrade, the automatic update mechanism is  
deactivated for a **unique provider**



# SAME-DAY RELEASE IS A QUICK REACTION TO A SPECIFIC DEMAND FROM THE PRIOR RELEASE



# SAME-DAY RELEASES INTRODUCE IMPORTANT CHANGES SUCH AS BUG FIXES

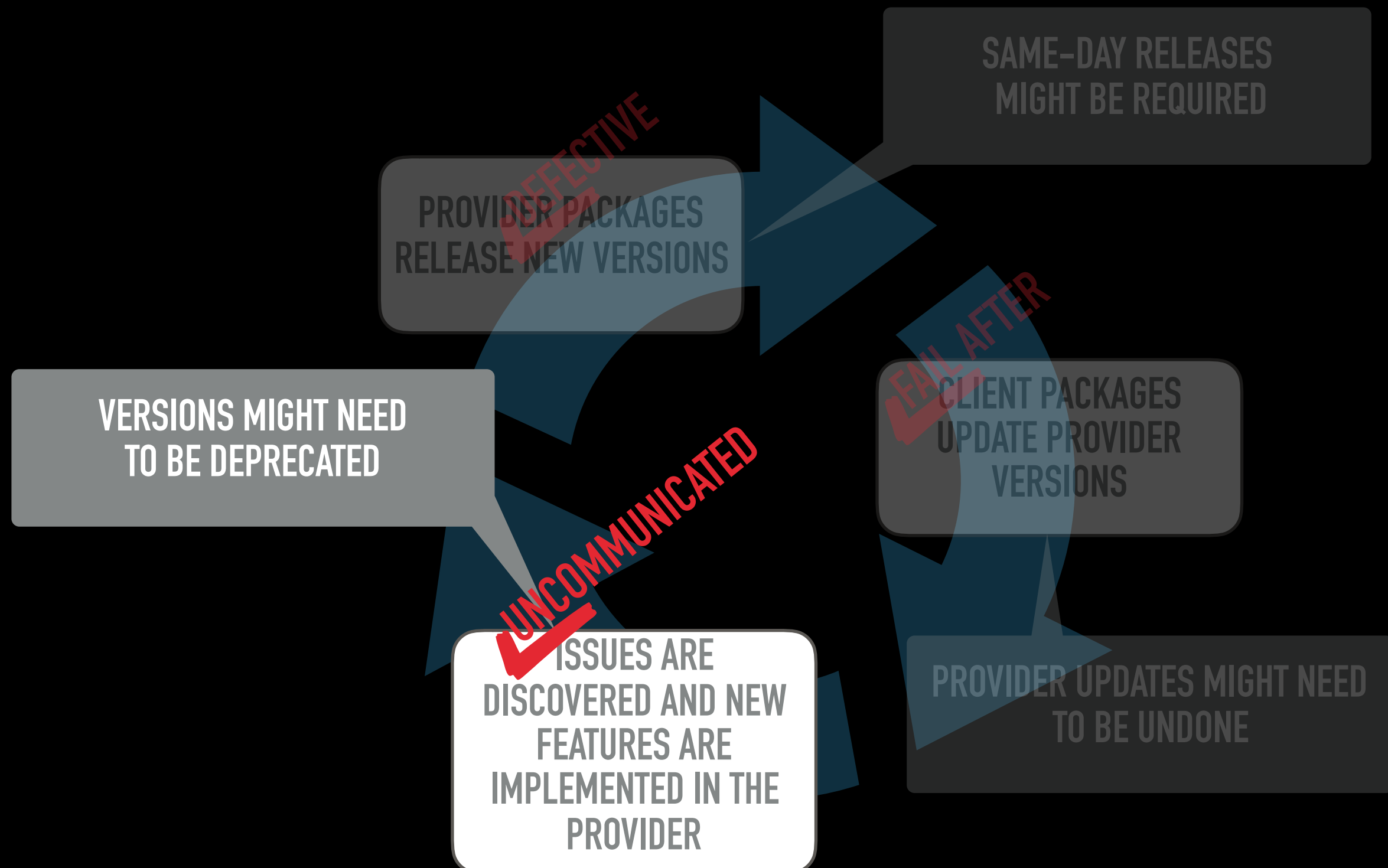
- 96% of the popular packages publish at least one same-day release
- One third of the same-day releases are followed by another same-day release
- 24% of the same-day releases have larger changes than their associated prior release
- Same-day releases are generally adopted faster than regular releases



TO (ADOPT THIS) RELEASE OR NOT TO  
(ADOPT THIS) RELEASE, THAT'S THE QUESTION

- There is a trade-off between the risk and value of a same-day release
  - Especially considering software ecosystems with intensive usage of dependencies
- Existing tools offer limited support to help developers reason about this trade-off

# DEPRECATION COMMUNICATES THAT THE USAGE OF A CERTAIN VERSION SHOULD BE AVOIDED



# THERE ARE FIVE REASONS FOR DEPRECATING A PACKAGE OR RELEASE

Rationale for deprecation	Package deprecation	Release deprecation	Example
Withdrawal	49.0%	12.0%	"This module is no longer maintained"
Supersession	45.0%	20.0%	"Version 1.x (...) superseded by 2.x"
Defect	0.5%	63.0%	"Buggy implementation of class mixins"
Test	5.0%	2.5%	"(...) this was a pre-release (...)"
Incompatibility	0.5%	2.5%	"(...) not compatible with sqb >0.7.0"

# DEPRECATED RELEASES ARE FAIRLY ADOPTED BY CLIENT PACKAGES

- 27% of the client packages **directly** adopt a deprecated release
- 54% of the client packages **transitively** adopt a deprecated release
- Transitive adoptions can be traced with relative ease
  - In general, a **single direct provider is accountable for the transitive adoption** of a deprecated release
  - In general, no further than 4 degrees of separation

## DEVELOPERS OFTEN MISUSE THE DEPRECATION MECHANISM, WHICH REQUIRES SPECIFIC IMPROVEMENTS

- 31% do not indicate a **replacement** package or release
- 36% do not state the **rationale** for the deprecation
- 65% deprecate only the latest release when **intended** to deprecate the whole package

# THERE IS FRUITFUL RESEARCH TO BE DONE ON DEPENDENCY MANAGEMENT

- Supporting provider packages on understanding how their changes impact client packages
  - Will this change break my clients code?
  - How relevant is this change to my clients?
- Implications of how dependencies are employed at code level
  - How important is a vulnerability update to me?
  - Should I increase test coverage to any of my provider packages?
- Automation of dependency-related tasks (e.g., dependency bots)
  - How to reduce false alerts from bots?
  - What else should bots do beyond testing dependency updates?

# DEPENDENCY MANAGEMENT HAVE A LARGE IMPACT ON DAILY DEVELOPERS' ACTIVITIES

NPM ERR!

Software

What happens when t  
killing someone with a

What will be the fate of an one

By Thom

{\* SOFTWARE \*}

**NBD: A popular HTTP-fetching npm  
code library used by 48,000 other  
modules retires no more updates**

eeek goes to prison for

WIREDBACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY

LILY HAY NEWMAN SECURITY 09.14.2017 01:27 PM

## Equifax Officially Has No Excuse

A patch that would have prevented the devastating Equifax breach had been available for months.



```
9 }  
10 return str;  
11 }
```

# DEPENDENCY MANAGEMENT HAVE A LARGE IMPACT ON DAILY DEVELOPERS' ACTIVITIES

NPM ERR!

Software

What happens when t  
killing someone with a

What will be the fate of an one

By Thom

{\* SOFTWARE \*}

NBD: A popular HTTP-fetching npm  
code library used by 48,000 other  
modules retires no more updates

eeek goes to prison for


LILY HAY NEWMAN

SECURITY

09.14.2017 01:27 PM

Equifax Officially Has No Excuse

A patch that would have prevented the devastating Equifax breach had been available for months.



9

}

10

return str;

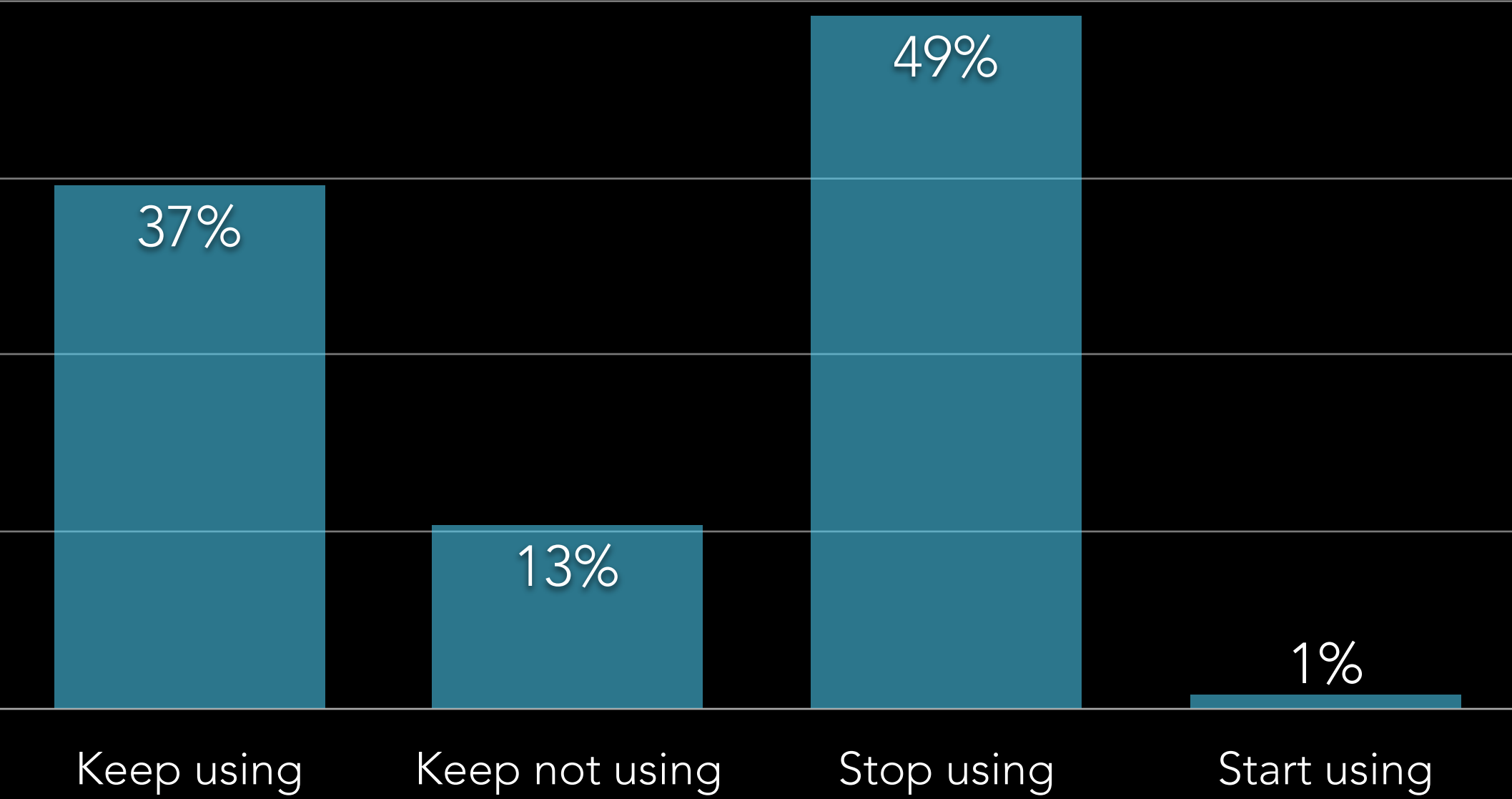
11

}



# DOWNGRADES ARE ASSOCIATED WITH THE DEACTIVATION OF AUTOMATIC UPDATES

USAGE OF THE AUTOMATIC UPDATE MECHANISM AFTER DOWNGRADING



DEPENDENCY MANAGEMENT HAVE A LARGE  
IMPACT ON DAILY DEVELOPERS' ACTIVITIES

DOWNGRADES ARE ASSOCIATED WITH THE  
DEACTIVATION OF AUTOMATIC UPDATES

NPM ERR!

Software

What happens when t  
killing someone with a

What will be the fate of an one

By Thom

{\* SOFTWARE \*}

NBD: A popular HTTP-fetching npm  
code library used by 48,000 other  
modules retires no more updates

eeek goes to prison for

LILY HAY NEWMAN

SECURITY

09.14.2017 01:27 PM

Equifax Officially Has No Excuse

A patch that would have prevented the devastating Equifax breach had been available for months.

9 }

10 return str;

11 }

USAGE OF THE AUTOMATIC UPDATE MECHANISM AFTER DOWNGRADING

A bar chart with four categories on the x-axis: 'Keep using', 'Keep not using', 'Stop using', and 'Start using'. The y-axis represents percentages. The bars are blue. The values are: 'Keep using' at 37%, 'Keep not using' at 13%, 'Stop using' at 49%, and 'Start using' at 1%.

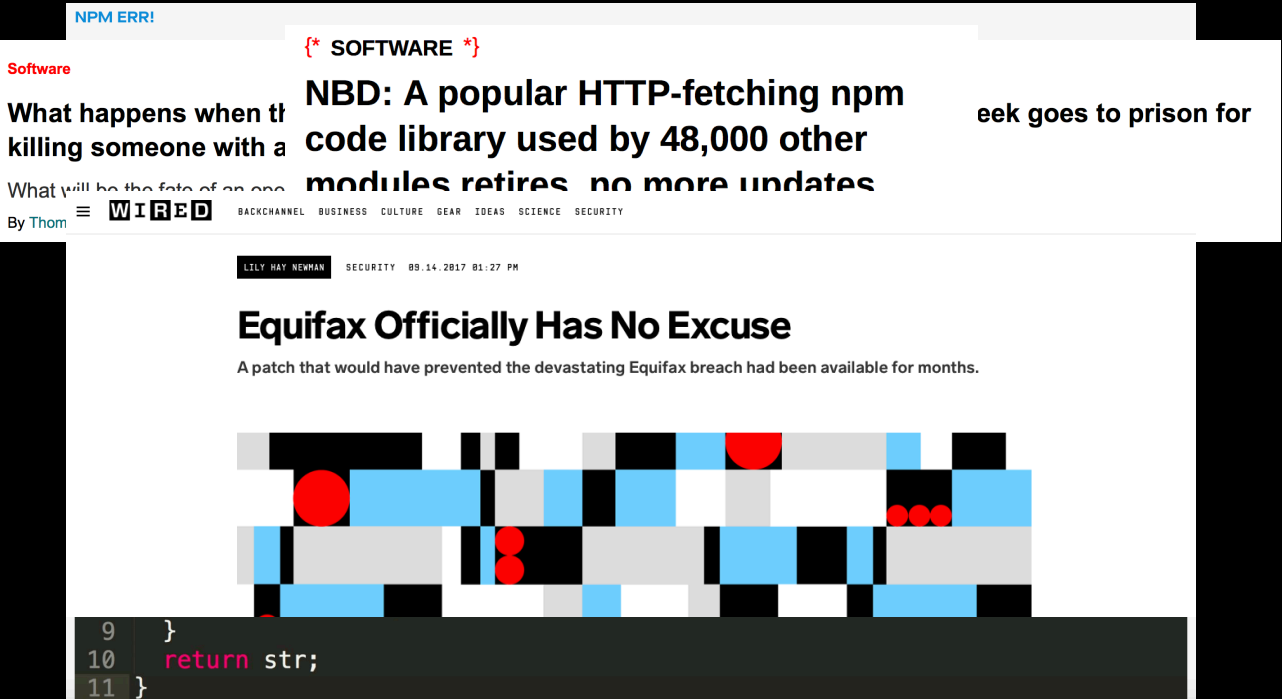
Usage Category	Percentage
Keep using	37%
Keep not using	13%
Stop using	49%
Start using	1%

## DEVELOPERS OFTEN MISUSE THE DEPRECATION MECHANISM, WHICH REQUIRES SPECIFIC IMPROVEMENTS

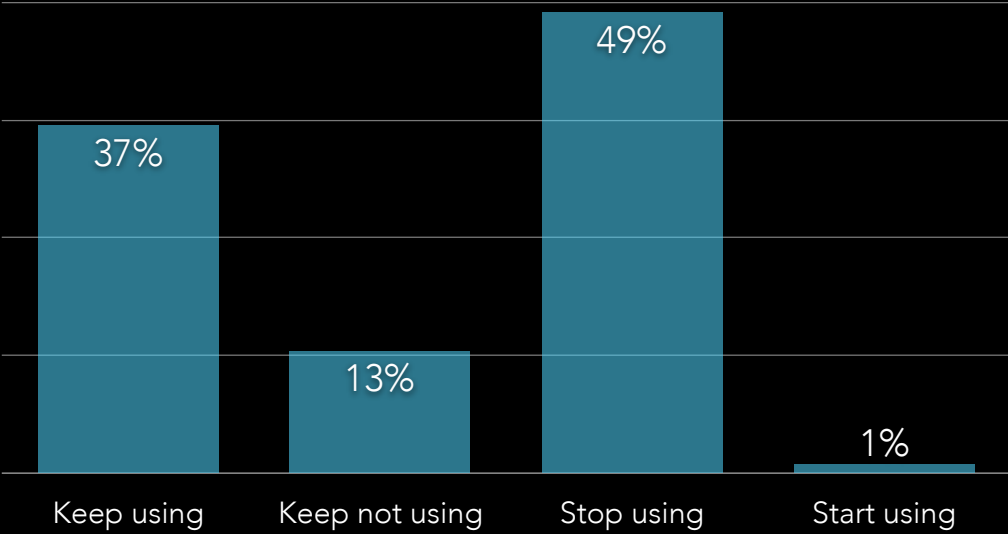
- 31% do not indicate a **replacement** package or release
- 36% do not state the **rationale** for the deprecation
- 65% deprecate only the latest release when **intended** to deprecate the whole package

DEPENDENCY MANAGEMENT HAVE A LARGE IMPACT ON DAILY DEVELOPERS' ACTIVITIES

DOWNGRADES ARE ASSOCIATED WITH THE DEACTIVATION OF AUTOMATIC UPDATES



USAGE OF THE AUTOMATIC UPDATE MECHANISM AFTER DOWNGRADING



DEVELOPERS OFTEN MISUSE THE DEPRECATION MECHANISM, WHICH REQUIRES SPECIFIC IMPROVEMENTS

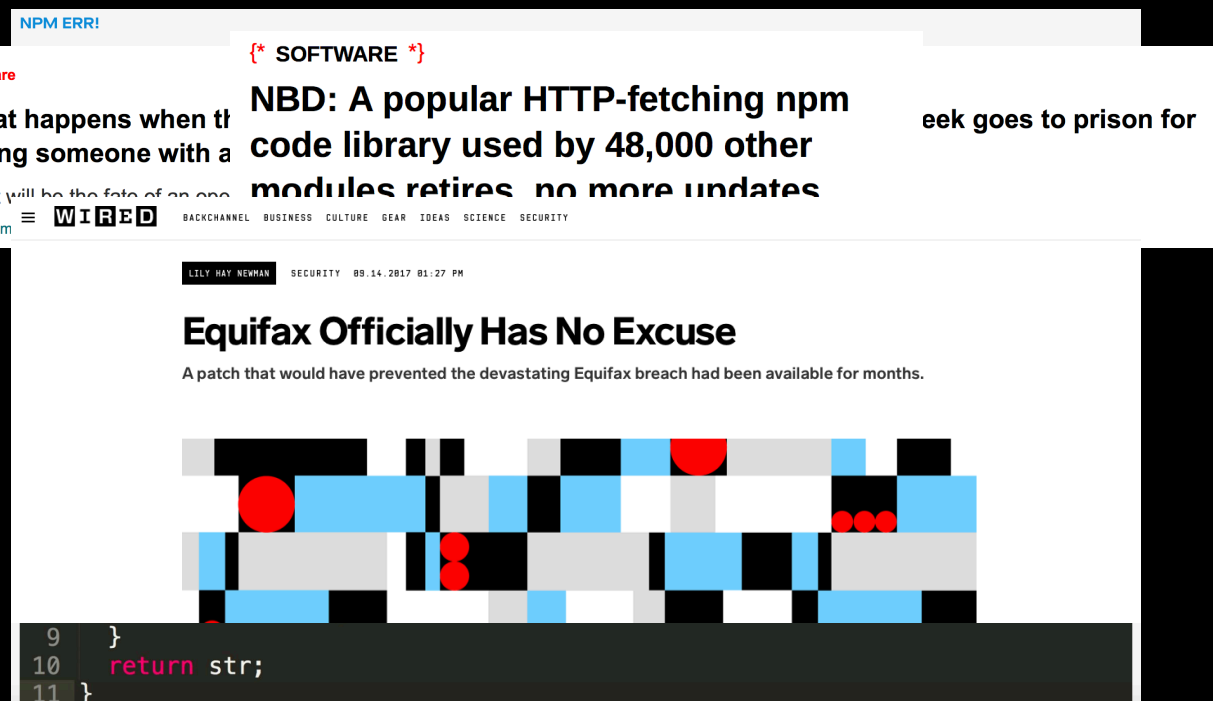
- 31% do not indicate a **replacement** package or release
- 36% do not state the **rationale** for the deprecation
- 65% deprecate only the latest release when **intended** to deprecate the whole package

# THERE IS FRUITFUL RESEARCH TO BE DONE ON DEPENDENCY MANAGEMENT

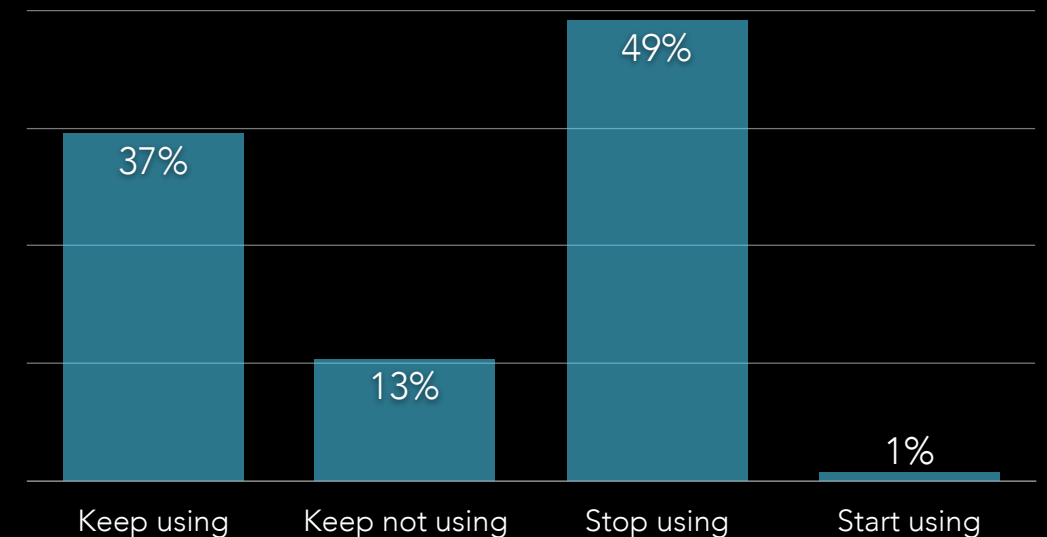
- Supporting provider packages on understanding how their changes impact client packages
  - Will this change break my clients code?
  - How relevant is this change to my clients?
- Implications of how dependencies are employed at code level
  - How important is a vulnerability update to me?
  - Should I increase test coverage to any of my provider packages?
- Automation of dependency-related tasks (e.g., dependency bots)
  - How to reduce false alerts from bots?
  - What else should bots do beyond testing dependency updates?

## DEPENDENCY MANAGEMENT HAVE A LARGE IMPACT ON DAILY DEVELOPERS' ACTIVITIES

## DOWNGRADES ARE ASSOCIATED WITH THE DEACTIVATION OF AUTOMATIC UPDATES



USAGE OF THE AUTOMATIC UPDATE MECHANISM AFTER DOWNGRADING



## DEVELOPERS OFTEN MISUSE THE DEPRECATION MECHANISM, WHICH REQUIRES SPECIFIC IMPROVEMENTS

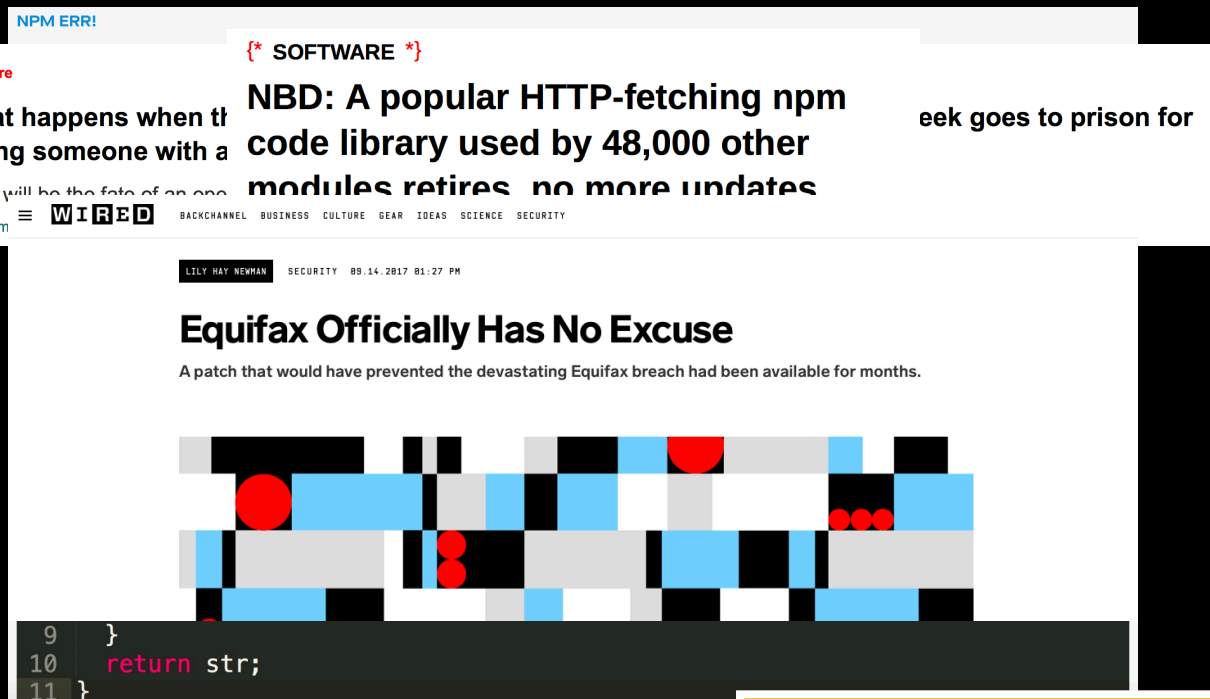
## THERE IS FRUITFUL RESEARCH TO BE DONE ON DEPENDENCY MANAGEMENT

- 31% do not indicate a **replacement** package or release
- 36% do not state the **rationale** for the deprecation
- 65% deprecate only the latest release when **intended** to deprecate the whole package

- Supporting provider packages on understanding how their changes impact client packages
  - Will this change break my clients code?
  - How relevant is this change to my clients?
- Implications of how dependencies are employed at code level
  - How important is a vulnerability update to me?
  - Should I increase test coverage to any of my provider packages?
- Automation of dependency-related tasks (e.g., dependency bots)
  - How to reduce false alerts from bots?
  - What else should bots do beyond testing dependency updates?

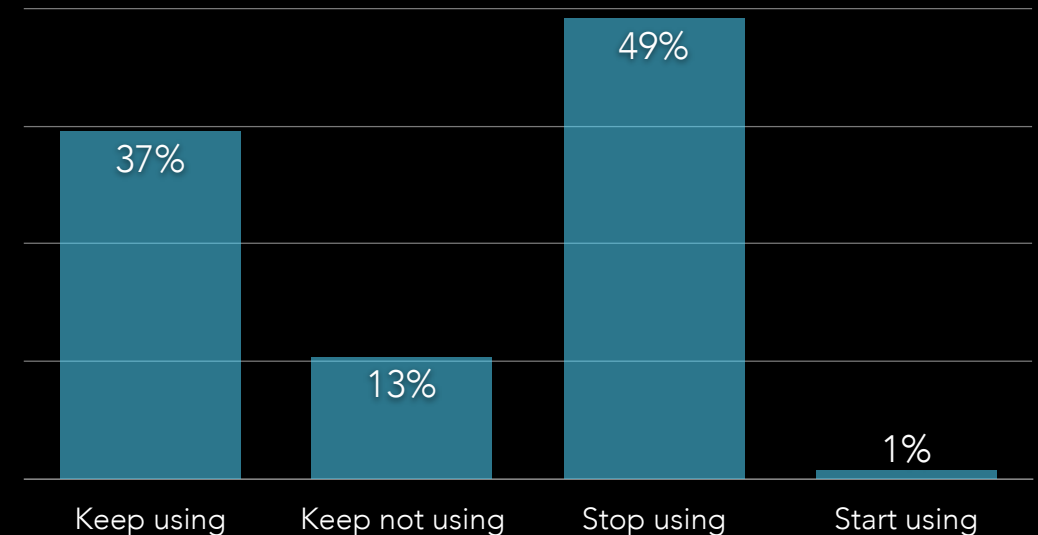
DEPENDENCY MANAGEMENT HAVE A LARGE  
IMPACT ON DAILY DEVELOPERS' ACTIVITIES

DOWNGRADES ARE ASSOCIATED WITH THE  
DEACTIVATION OF AUTOMATIC UPDATES



FILIPPE.COGO@GMAIL.COM

USAGE OF THE AUTOMATIC UPDATE MECHANISM AFTER DOWNGRADING



DEVELOPERS OFTEN MISUSE THE DEPRECATION  
MECHANISM, WHICH REQUIRES SPECIFIC IMPROVEMENTS

- 31% do not indicate a **replacement** package or release
- 36% do not state the **rationale** for the deprecation
- 65% deprecate only the latest release when **intended** to deprecate the whole package

THERE IS FRUITFUL RESEARCH TO BE  
DONE ON DEPENDENCY MANAGEMENT

- Supporting provider packages on understanding how their changes impact client packages
  - Will this change break my clients code?
  - How relevant is this change to my clients?
- Implications of how dependencies are employed at code level
  - How important is a vulnerability update to me?
  - Should I increase test coverage to any of my provider packages?
- Automation of dependency-related tasks (e.g., dependency bots)
  - How to reduce false alerts from bots?
  - What else should bots do beyond testing dependency updates?