

The Priority We're Getting Wrong



Jiří Brada



Meet Magento DE 2026 · Mainz · 22 Oct 2026

Can open source still compete with SaaS?



Security is
taken for granted.

Hi - I'm Jiří

- Almost **10 years in Magento**
- **Technical Commerce Lead** at  **PushON**
- Creator of **Package Maven** - a public directory of Magento modules with tests





“A game for kids just starting to learn how to hack.”

AI amplifies everything



The good half gets all the airtime



**It amplifies the bad half
exactly as much**

The vulnerabilities aren't new. The cost of weaponising one is.

The threat

04



BlockStrap



StyleSmuggler



PolyShell



Address upload



SessionReaper



CosmicSting

...and now they can do literally anything

- Read env.php → DB + crypt key
- Every order and customer
- Malicious JS • a new admin

The entire store is theirs

Forge an **admin token** → no login, no 2FA

```
<?php
return [
    'crypt' => [
        'key' =>
'base64*****=
    'db' => [
        'table_prefix' => '',
        'connection' => [
            'default' => [
                'host' => 'db',
                'dbname' => 'db',
                'username' => 'db',
                'password' => 'db',
                'model' => 'mysql4',
                'engine' => 'innodb',
                'active' => '1',
                'driver_options' => [
                    1014 => false
                ]
            ]
        ]
    ]
],
```

The moment data leaks, two clocks start



72h → DPA

data protection
authority



24h → BSI

medium & large shops

Both start when you find out - **not when it happened**

Damages?

Fine from the regulator

**Customers claim
compensation**

Brand damage

Ecosystem damage

Where we stand

2022

Security is layers

No single wall · no one setting, WAF, or patch



**Every leak = small failures
lining up**



**Skipped patch · old key ·
exposed file**

Choose your responsibility

Which layers can I *reasonably* own - and which do I delegate?



Delegate the infrastructure

→ Commerce Cloud / specialist hosts



Own what nobody can cover for us

→ Magento itself

Delegating the server does **not delegate Magento**

Why **the weight** goes on updates

Not the only layer that matters - the **cheapest attack**

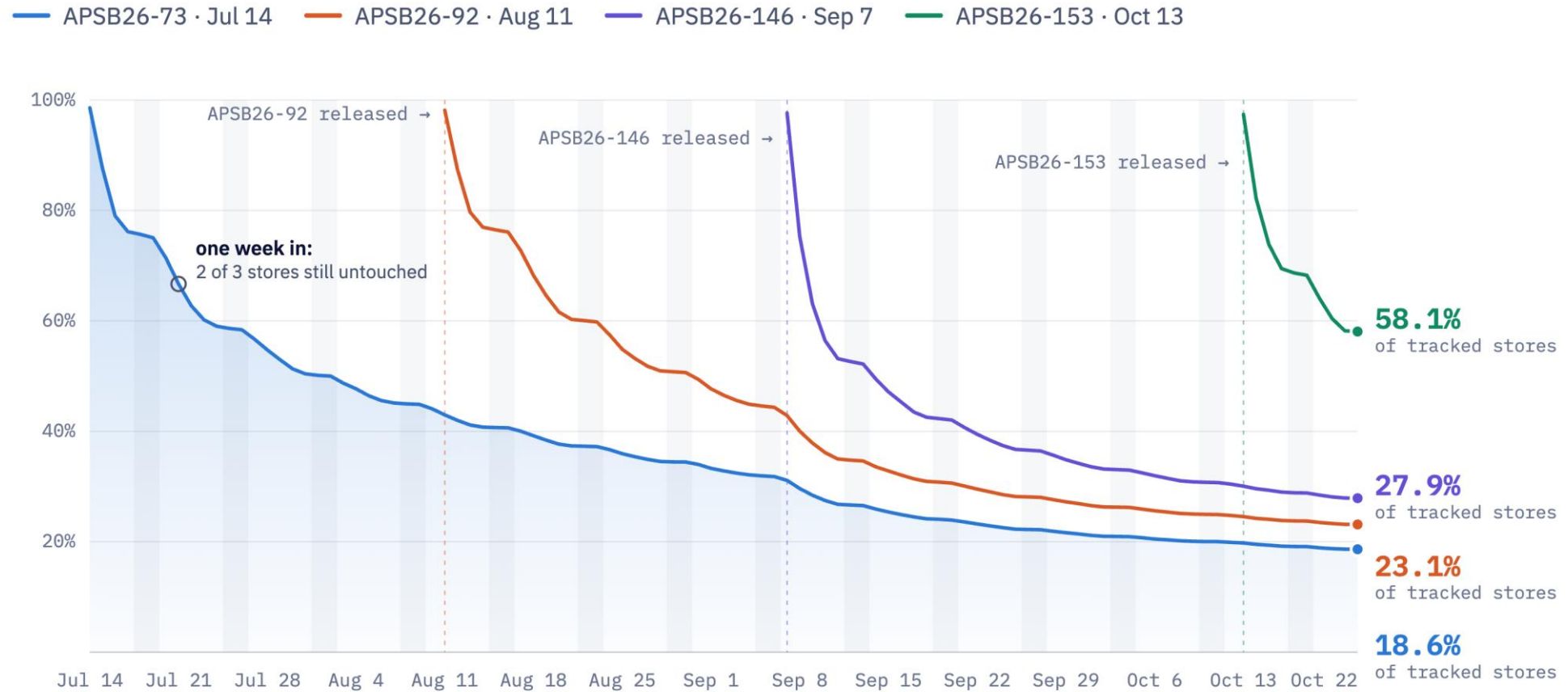


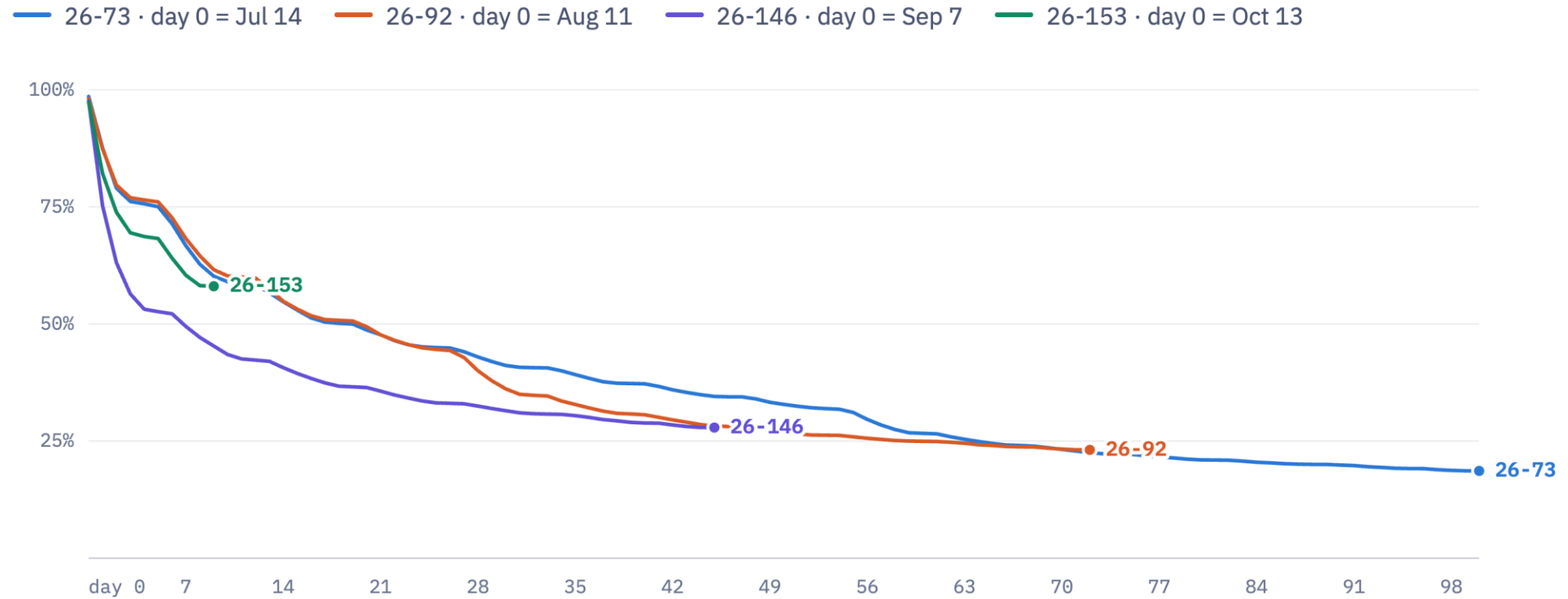
Find a bug + build an exploit → hard



Read the patch + ask a model → easy

Patch → exploit: 63 days → 5 days
A spray doesn't aim - it hits **everyone it can reach**





Why are so many behind? - none of it is laziness

01

Nobody *feels* it

02

Heavy customisation

03

Poor process

04

The ecosystem

Patching should be
boring.

2.4.9 · feature release

once a year · can break modules

2.4.8-p5 · security release

all earlier fixes · no features

Isolated patch · monthly

one fix, on top of the latest -pN

Emergency hotfix

zero-days · deploy now

Isolated patches fix the code. The version never moves.

How an isolated patch reaches your store

Same fix from Adobe - *three* different paths



Adobe Commerce

→ the cloud-patches package

- automated in CI



Magento Open Source

→ patch files, by hand

- package dropped in Sept



Mage-OS

→ its own releases

- ...waits for Adobe's fixes

Every manual step = days added to the curve. Only one path is **boring**.

Magento module vendors **don't** respect semver



Symfony

Bump a patch → you trust it works.

39% break semver



A 3rd party magento module

Bump a module patch → you brace.

48% among 100k+ installs

We ship modules too - **the fix starts at home**

The defence

03

Start with the foundations

The work that makes upgrades boring, is the work that makes AI safe - one investment.

04

Reviews - AI and human

03

CI with deterministic boundaries

02

Best-practice customisations

01

Lean Magento

Shrink the attack surface

Code you don't run *can't* be exploited



Remove what you don't use

→ composer replace



Block what you can't remove

→ /graphql · /soap at the edge

On the latest attack, this kept my clients **out**.

Shrink the data surface

Data you don't copy *can't* leak



Anonymise every dump

→ before it leaves production



Treat every copy as production

→ laptops, Slack, shared drives

Every copy of your database is **another store to protect.**

ampersand/magento2-upgrade-patch-helper



- flags where an upgrade will break



- Tip: hand that list to a coding agent

Created by Luke Rodgers

So you automated it. From where?

```
{
  "name": "magento/project-community-edition",
  "description": "eCommerce Platform for Growth (Community Edition)",
  "config": {
    "allow-plugins": {
      "dealerdirect/phpcodesniffer-composer-installer": true,
      "laminas/laminas-dependency-plugin": true,
      "magento/*": true,
      "php-http/discovery": true
    }
  },
  "repositories": [
    { "type": "composer", "url": "https://vendor-a.example.com/" },
    { "type": "composer", "url": "https://vendor-b.example.com/" },
    { "type": "composer", "url": "https://vendor-c.example.com/" },
    { "type": "composer", "url": "https://vendor-d.example.com/" },
    { "type": "composer", "url": "https://vendor-e.example.com/" },
    { "type": "composer", "url": "https://vendor-f.example.com/" },
    { "type": "composer", "url": "https://vendor-g.example.com/" },
    { "type": "composer", "url": "https://vendor-h.example.com/" },
    { "type": "composer", "url": "https://vendor-i.example.com/" },
    { "type": "composer", "url": "https://vendor-j.example.com/" },
    { "type": "composer", "url": "https://repo.magento.com/" }
  ]
}
```



How Composer picks a package

- First repo with the name wins
- Packagist is added last, silently
- Composer checks the file, not the publisher

```
{
  "repositories": [
    { "type": "composer", "url": "https://vendor-a.example.com/" },
    { "type": "composer", "url": "https://vendor-b.example.com/" },
    { "type": "composer", "url": "https://vendor-c.example.com/" },
    { "type": "composer", "url": "https://vendor-d.example.com/" },
    { "type": "composer", "url": "https://vendor-e.example.com/" },
    { "type": "composer", "url": "https://vendor-f.example.com/" },
    { "type": "composer", "url": "https://vendor-g.example.com/" },
    { "type": "composer", "url": "https://vendor-h.example.com/" },
    { "type": "composer", "url": "https://vendor-i.example.com/" },
    { "type": "composer", "url": "https://vendor-j.example.com/" },
    { "type": "composer", "url": "https://repo.magento.com/" }
  ]
}
```

Composer plugins run code - as you

- Dependency scripts never run
- Plugins run their PHP - as you
- `magento/*` is a wildcard grant

```
{
  "config": {
    "allow-plugins": {
      "dealerdirect/phpcodesniffer-composer-installer": true,
      "laminas/laminas-dependency-plugin": true,
      "magento/*": true,
      "php-http/discovery": true
    }
  }
}
```

Let's combine the two

- A hacked vendor repo ships magento/module-catalog
- The name passes magento/*
- No prompt. Next command, it runs.

```
{
  "name": "magento/project-community-edition",
  "description": "eCommerce Platform for Growth (Community Edition)",
  "config": {
    "allow-plugins": {
      "dealerdirect/phpcodesniffer-composer-installer": true,
      "laminas/laminas-dependency-plugin": true,
      "magento/*": true,
      "php-http/discovery": true
    }
  },
  "repositories": [
    { "type": "composer", "url": "https://vendor-a.example.com/" },
    { "type": "composer", "url": "https://vendor-b.example.com/" },
    { "type": "composer", "url": "https://vendor-c.example.com/" },
    { "type": "composer", "url": "https://vendor-d.example.com/" },
    { "type": "composer", "url": "https://vendor-e.example.com/" },
    { "type": "composer", "url": "https://vendor-f.example.com/" },
    { "type": "composer", "url": "https://vendor-g.example.com/" },
    { "type": "composer", "url": "https://vendor-h.example.com/" },
    { "type": "composer", "url": "https://vendor-i.example.com/" },
    { "type": "composer", "url": "https://vendor-j.example.com/" },
    { "type": "composer", "url": "https://repo.magento.com/" }
  ]
}
```

The same configuration, hardened

```
{
  "name": "magento/project-community-edition",
  "config": {
    "allow-plugins": {
      "magento/magento-composer-installer": true,
      "magento/composer-dependency-version-audit-plugin": true,
      "dealerdirect/phpcodesniffer-composer-installer": true,
      "laminas/laminas-dependency-plugin": true,
      "php-http/discovery": false
    }
  },
  "repositories": {
    "magento": {
      "type": "composer",
      "url": "https://repo.magento.com/",
      "only": ["magento/*"]
    },
    "vendor-a": {
      "type": "composer",
      "url": "https://vendor-a.example.com/",
      "only": ["vendor-a/*"]
    },
    "vendor-b": {
      "type": "composer",
      "url": "https://vendor-b.example.com/",
      "only": ["vendor-b/*"]
    },
    "packagist.org": false,
    "packagist-filtered": {
      "type": "composer",
      "url": "https://repo.packagist.org",
      "exclude": ["pushon/*"]
    }
  }
}
```



BASELINE

REPOSITORY

PLUGINS

COMBINED ATTACK

HARDENED

composer audit



Composer audit - and where it looks



Public feeds

- GitHub + FriendsOfPHP → **composer audit**
- matched by **package name** only



Your own repositories

- any repo can publish **security-advisories**
- **audit asks every repo** - private too

Run composer audit --locked in CI

Code fixed. Composer doesn't know.



The name gap

- advisories filed against **community-edition**
- you install **product-community-edition**
- → **"No advisories found"**



Isolated patches

- fixes ship as **isolated patches**
- patched, but still **2.4.6-p12** → "vulnerable"

Tooling assumes version = code. Magento's patches don't.

Virtual patching



An **edge rule the same afternoon** blocks the exploit

**It doesn't fix the bug - it buys time.
And time is the entire game.**

Two blocking layers, **not** one



Edge - general-purpose WAF

- Cloudflare WAF - **OWASP** rules
- cover what shares the host - e.g. **WordPress**



Inside - Magento-aware

- Sansec Shield
- a module inside the app

Let's get the
priority right.