

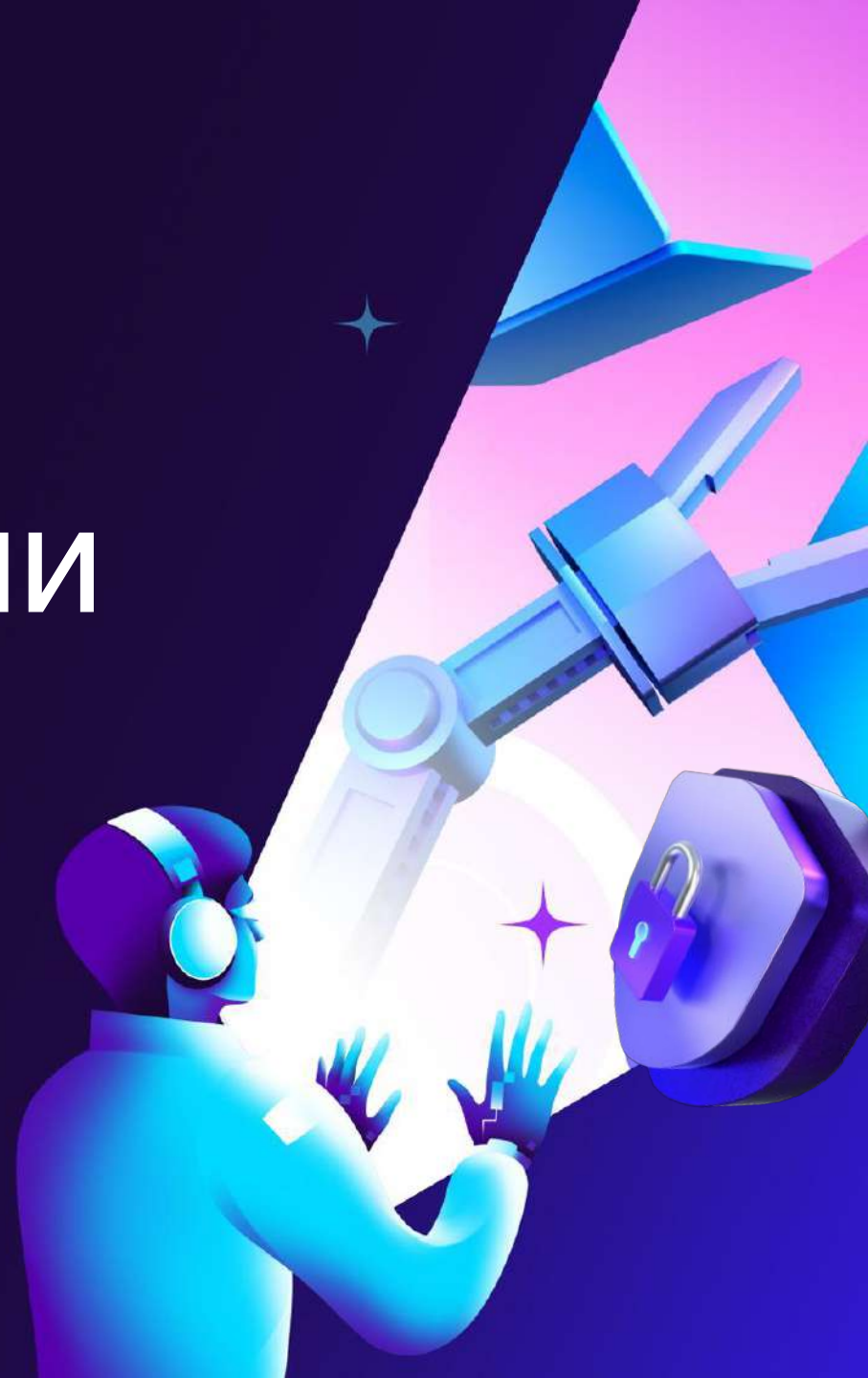
Алексей Смирнов

CEO @CodeScoring

Никита Бесперстов

Research engineer @CodeScoring

Новый *log4shell*, о котором вам забыли рассказать

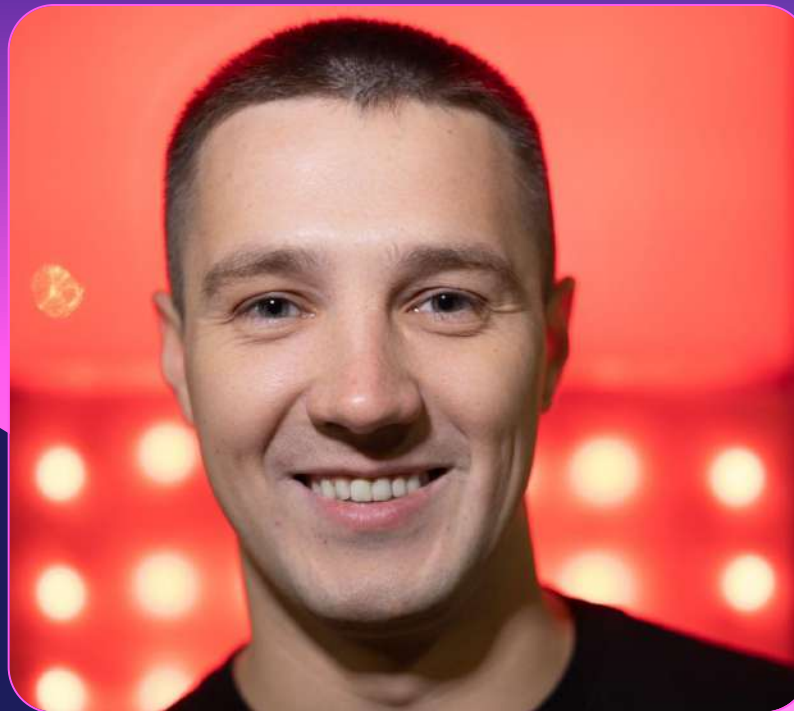




Алексей Смирнов

CEO @CodeScoring

ex-астроном, ex-программист,
не-фаззингист, знаю в AppSec,
Основал CodeScoring



Никита Бесперстов

Research engineer @CodeScoring

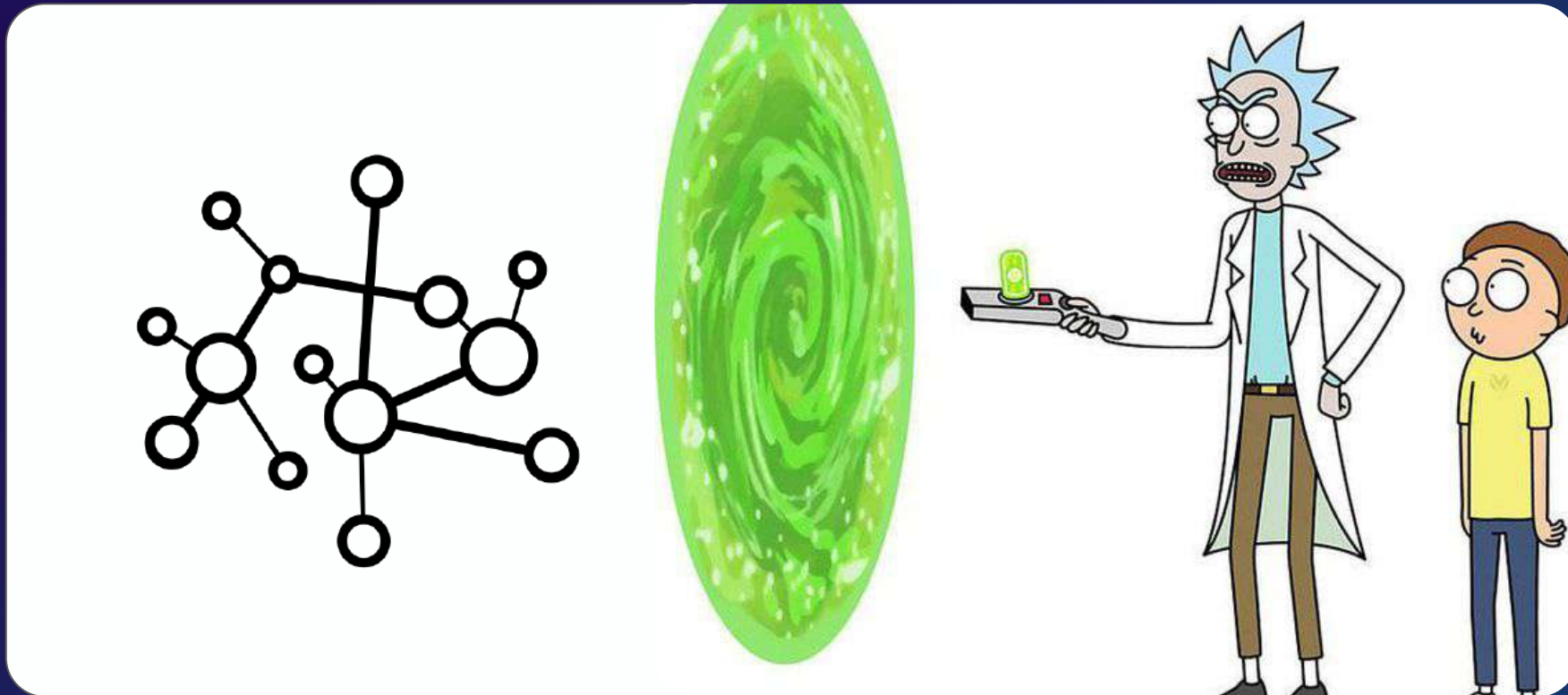
Программист, фаззингист,
AppSec-исследователь.
Пишу в OSS. В отрасли с 2013.

01



Предыстория

Чтобы понять зависимости мы строим Мегаграф всего Open Source



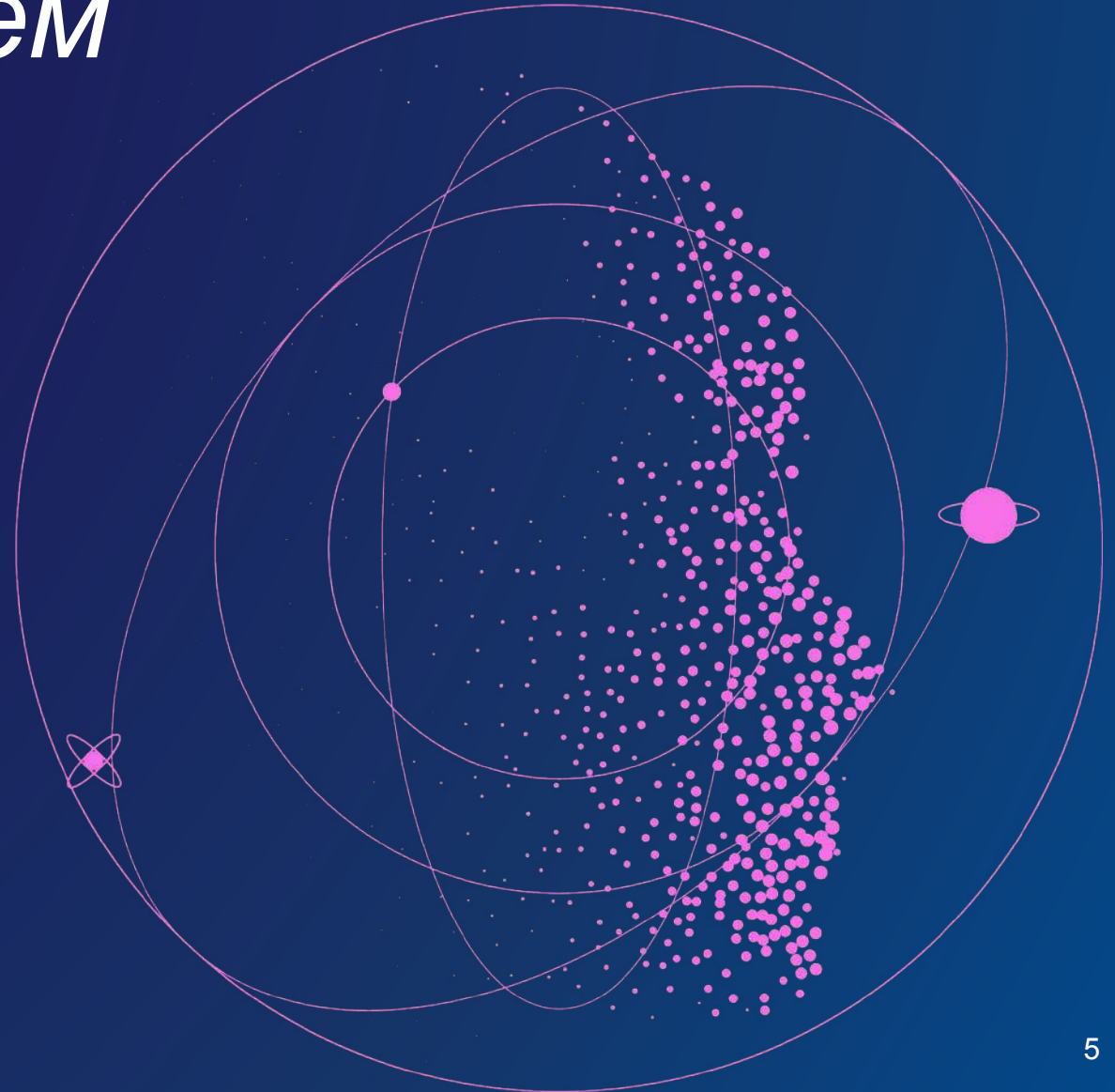
<https://youtu.be/dqjGQBe2yTY>

Доклад: “Проблемы отцов и детей: Аналитика и триаж транзитивных зависимостей”, PHD, 2024

Разрешаем зависимости всех пакетов и знаем

- / Какой процент экосистем затрагивает обнаруженная уязвимость
- / Насколько глубоко залегает обнаруженная проблема в транзитивных зависимостях
- / Как быстро сообщество реагирует на проблемы и выпускает безопасные версии
- / Какие цели анализа безопасности компонентов выбрать в первую очередь

* ну и чтобы добавить продукту **скорингов**



Мегаграф & log4shell

затронуто
8% maven
экосистемы

2024 год:

Затронуто: **20 588** версий
пакетов

Исправлено 28%: **5 837**

Не исправлено 72%: **14 751**



Здорово
понимать «влияние»
уязвимостей
на экосистему



— А что ещё *здорово*?



— А что ещё *здорово*?

— Отнести продукт
на *сертификацию*!



Сертификация подразумевает



- ❖ Сборку **только** из исходников
- ❖ Анализ известных уязвимостей (SCA)
- ❖ Проведение статки для всего (включая OSS)
- ❖ Проведение динамики (фаззинга)
- ❖ Поиск секретов и прочие практики



Всё понятно, но...

- ❖ ~~Сборку только из исходников~~
- ❖ ~~Анализ известных уязвимостей (SCA)~~
- ❖ ~~Проведение статики для всего (включая OSS)~~
- ❖ Проведение динамики (фаззинга)
- ❖ Поиск секретов и прочие практики



02



Фаззинг



Выбираем *инструментарий*



Sydr – инструмент динамического символьного выполнения бинарных программ от ИСП РАН

<https://sydr-fuzz.github.io/>

Нас особенно подкупает:

- ✦ гибридный фаззинг
- ✦ минимизация корпуса запросов

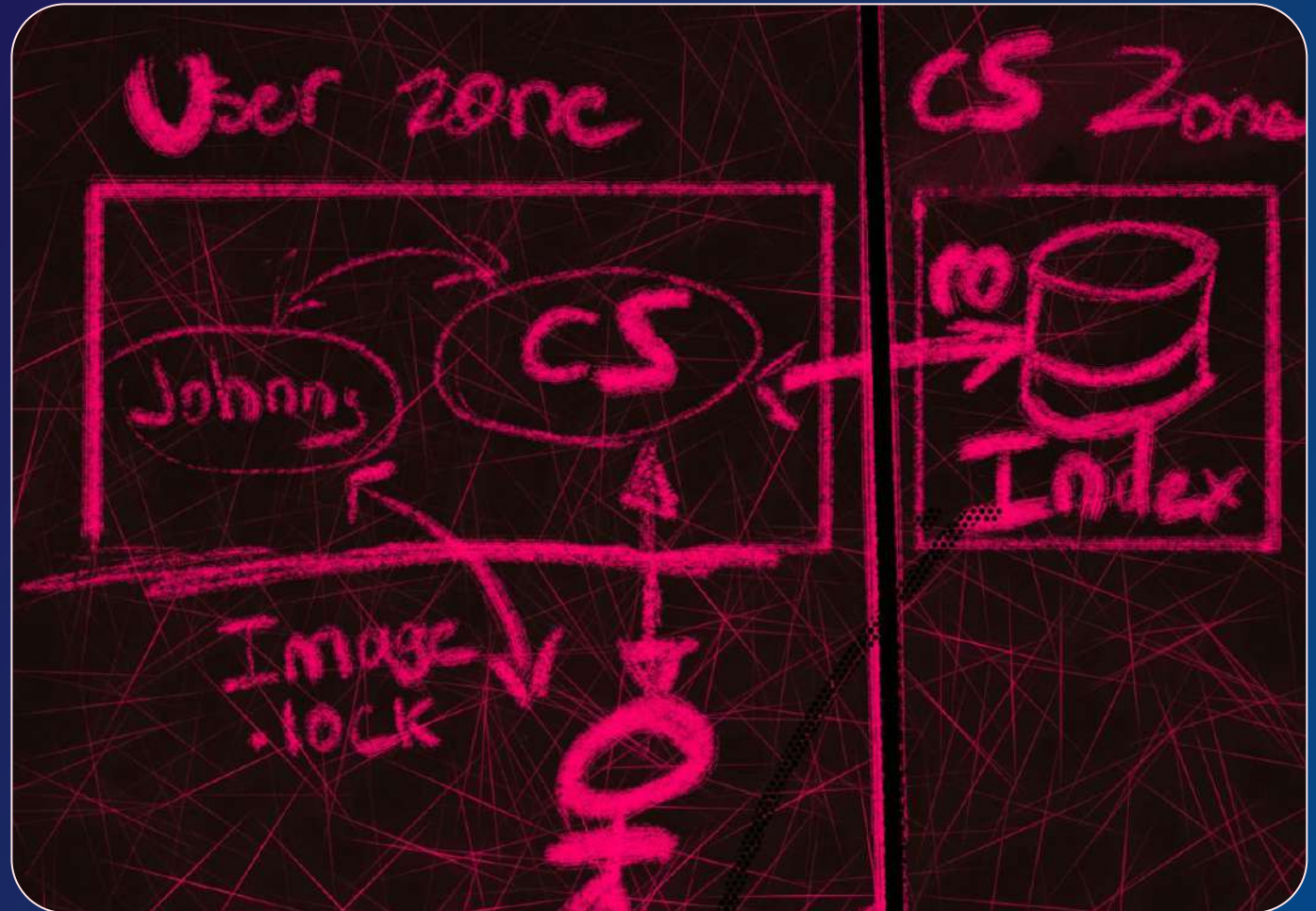


Разглядываем CodeScoring с помощью Natch
https://habr.com/ru/companies/isp_ras/articles/892548/

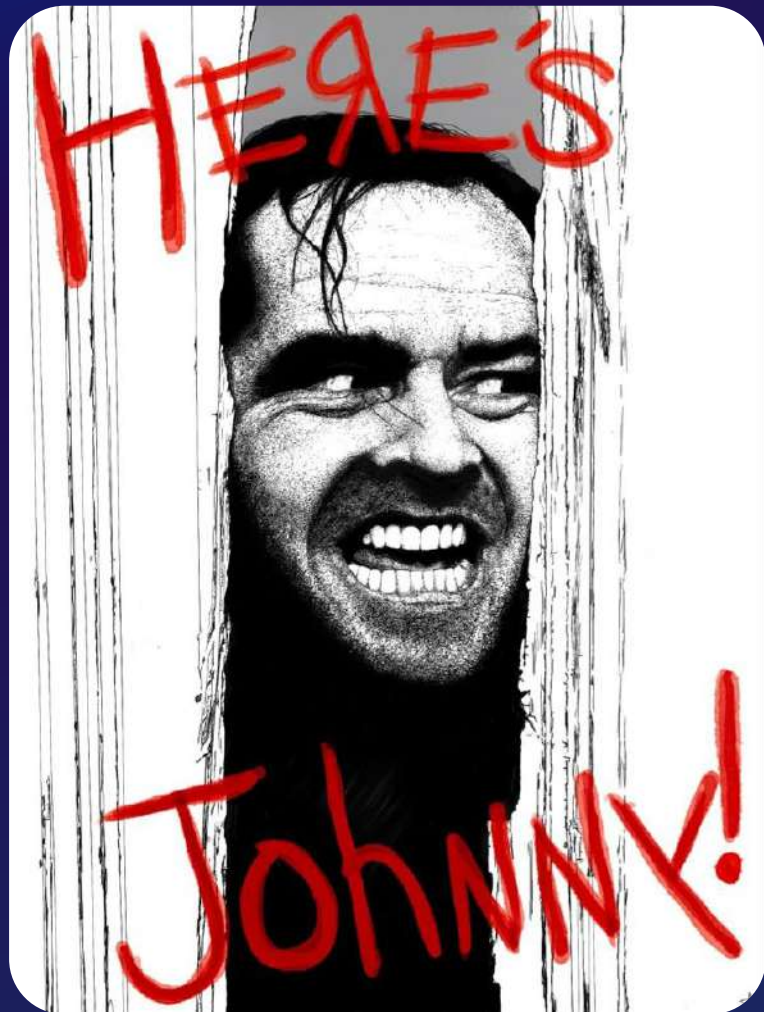
Рисуем архитектуру _(ツ)_/

У нас есть:

- Агент Johnny в пайплайнах
- Инсталляция CodeScoring
- База знаний про OSS & Vulns



Фаззинг агента *johnny*



Агент — это исполняемый бинарный файл, осуществляющий парсинг манифестов известных пакетных менеджеров, сканирование Docker-образов, анализ сборки C и C++, разбор архивов и поиск прямых включений Open Source библиотек по хэшам.

Короче, **умеет разбирать разные форматы данных.**

Фаззинг-план

- ✦ Готовим изолированную сборку
- ✦ Готовим сборку с отладочными символами
- ✦ Учим фаззер выбирать цель фаззинга

```
96 message scenario {
97     oneof scenario_type {
98         check_parser_scenario check_parser = 1;
99         check_archive_scenario check_archive = 2;
100     }
101 }
```

Фаззер теперь сам выбирает цель, не нужно об этом думать

```
1 syntax = "proto2";
2
3 package fuzz;
4 option go_package="proto/fuzz";
5
6 enum parser_type {
7     CONANFILE = 0;
8     CONANFILE_PY = 1;
9     CONAN_LOCK = 2;
10    CONDA_LOCK = 3;
11    CSHARP_DEPENDENCY_REPORT = 4;
12    CSHARP_NUSPEC = 5;
13    CSHARP_PACKAGES_CONFIG = 6;
14    CSHARP_PACKAGES_LOCK = 7;
15    CSHARP_PAKET_DEPENDENCIES = 8;
16    CSHARP_PAKET_LOCK = 9;
17    CSHARP_PROJ = 10;
18    CSHARP_PROJECT_ASSETS = 11;
19    CSHARP_PROJECT_LOCK = 12;
20    GO_MOD = 13;
21    GO_SUM = 14;
22    JAVA_GRADLE = 15;
23    JAVA_GRADLE_DEPENDENCY_TREE = 16;
24    JAVA_GRADLE_KTS = 17;
25    JAVA_GRADLE_LOCK = 18;
26    JAVA_IVY = 19;
27    JAVA_JAR = 20;
28    JAVA_MAVEN_DEPENDENCY_TREE = 21;
29    JAVA_POM = 22;
30    JAVA_SCALA_DEPENDENCY_TREE = 23;
31    JS_NPM_SHRINK_WRAP = 24;
32    JS_PACKAGE_JSON = 25;
33    JS_PACKAGE_LOCK = 26;
34    JS_YARN = 27;
```

Итоги запуска фаззера



Корпус тестов - 110 867 шт., минимизированный - **11 759 шт.**

Длительность выполнения анализа **~1 неделя**

Покрытие нас радует:

	function	fuzz	sydr	diff
/parsers/clang/conanfile_py_parser.go:55	Parse	90.60	96.90	6.30
/parsers/conda/conda_lock_parser.go:31	Parse	40.00	66.70	26.70
/parsers/csharp/csharp_proj_parser.go:26	Parse	6.70	40.00	33.30
/parsers/csharp/dependencyReportjson_parser.go:38	Parse	21.10	36.80	15.70
/parsers/csharp/nuspec_parser.go:36	Parse	25.00	83.30	58.30
/parsers/csharp/packagesconfig_parser.go:21	Parse	16.70	33.30	16.60
/parsers/csharp/packageslockjson_parser.go:20	Parse	20.80	29.20	8.40
/parsers/csharp/paketdependencies_parser.go:73	Parse	30.00	80.00	50.00
/parsers/csharp/paketlock_parser.go:27	Parse	52.90	73.50	20.60
/parsers/csharp/projectassetsjson_parser.go:22	Parse	6.10	20.40	14.30
/parsers/csharp/projectlockjson_parser.go:31	Parse	29.40	47.10	17.70
/parsers/go/gomod_parser.go:20	Parse	86.70	93.30	6.60
/parsers/java/gradle_dependency_tree_parser.go:161	Parse	95.70	97.80	2.10
/parsers/java/gradle_kts_parser.go:32	Parse	32.10	55.40	23.30
/parsers/java/gradlelockfile_parser.go:27	Parse	63.60	90.90	27.30
/parsers/java/ivyxml_parser.go:20	Parse	15.40	30.80	15.40
/parsers/java/pomxml_parser.go:203	Parse	13.00	56.50	43.50
/parsers/java/scala_dependency_tree_parser.go:52	Parse	29.80	94.70	64.90

Сравнение покрытия: go-fuzz и sydr+libfuzzer

03

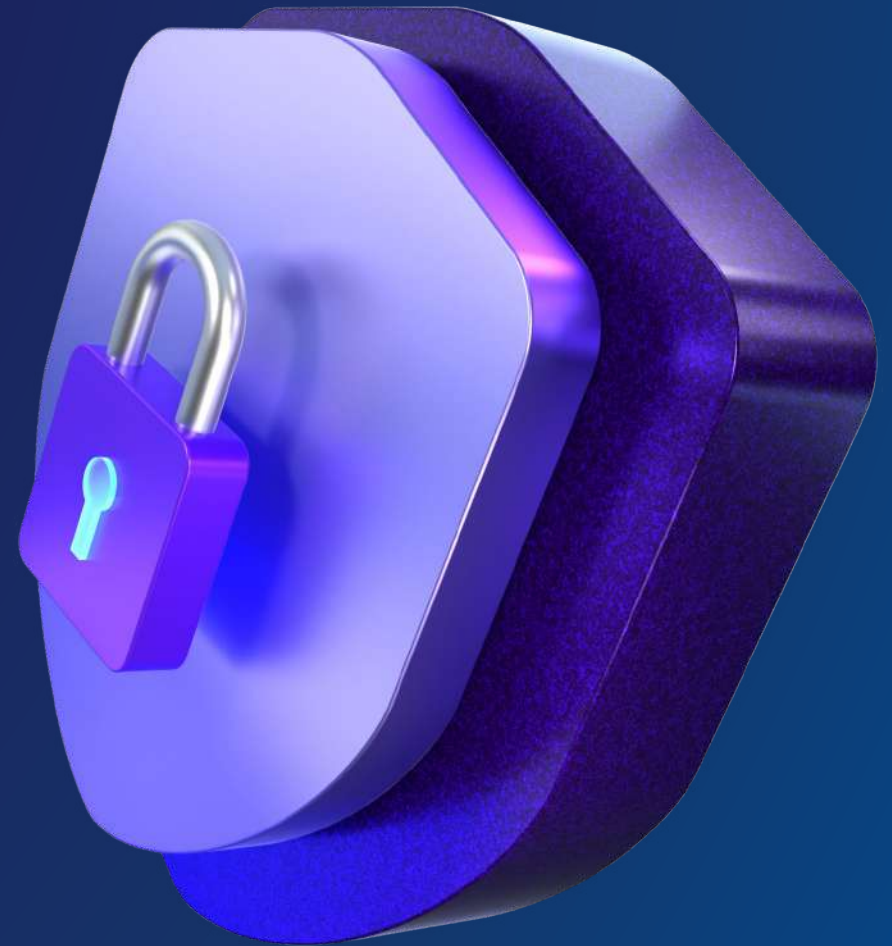


Находка



Фаззер подобрал ключ

```
[  
  0x34, 0x30, 0x79, 0x3a,  
  0x3a, 0x0d, 0x0d, 0x0d,  
  0x0d, 0x0d, 0x0d, 0x0d,  
  0x0d, 0x3c, 0x3c, 0x3a,  
  0x0d, 0x0d, 0x0d, 0x0d,  
  0x0d, 0x0d, 0x0d, 0x2d,  
  0x20, 0x20, 0x2d, 0x20,  
  0x23, 0x0d, 0x3f, 0x0d,  
  0x23, 0x0d, 0x0d, 0x23,  
  0x2d, 0x0d, 0x20, 0x2d,  
  0x0d, 0x3f, 0x0d, 0x23,  
  0x0d, 0x0d, 0x23, 0x2d,  
  0x2d  
]
```



Упала работа с go-yaml

```
1 panic: runtime error: hash of unhashable type []interface {} [recovered]
2   panic: runtime error: hash of unhashable type []interface {}
3
4 goroutine 1 [running]:
5 gopkg.in/yaml%2ev3.handleErr(0x140000cfe88)
6   /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/yaml.go:294 +0x80
7 panic({0x1044f1fe0?, 0x1400008e340?})
8   /go/libexec/src/runtime/panic.go:785 +0x124
9 gopkg.in/yaml%2ev3.(*decoder).merge(0x140000c95e0, 0x140000d2320, 0x140000d25a0, {0x1044f3c20?, 0x1400008e230?, 0x0?})
10  /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/decode.go:966 +0x138
11 gopkg.in/yaml%2ev3.(*decoder).mappingStruct(0x140000c95e0, 0x140000d2320, {0x1044f3c20?, 0x1400008e230?, 0x8?})
12  /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/decode.go:950 +0xc28
13 gopkg.in/yaml%2ev3.(*decoder).mapping(0x140000c95e0, 0x140000d2320, {0x1044f3c20?, 0x1400008e230?, 0x0?})
14  /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/decode.go:786 +0x84
15 gopkg.in/yaml%2ev3.(*decoder).unmarshal(0x140000c95e0, 0x140000d2320, {0x1044f3c20?, 0x1400008e230?, 0x3?})
16  /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/decode.go:510 +0x364
17 gopkg.in/yaml%2ev3.(*decoder).document(...)
18  /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/decode.go:527
19 gopkg.in/yaml%2ev3.(*decoder).unmarshal(0x140000c95e0, 0x140000d2280, {0x1044f3c20?, 0x1400008e230?, 0x12b155228?})
20  /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/decode.go:498 +0x248
21 gopkg.in/yaml%2ev3.unmarshal({0x140000ae080, 0x31, 0x31}, {0x1044e7960, 0x1400008e230}, 0xac?)
22  /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/yaml.go:167 +0x334
23 gopkg.in/yaml%2ev3.Unmarshal(...)
24  /go/pkg/mod/gopkg.in/yaml.v3@v3.0.1/yaml.go:89
25 main.main()
26  /random_project/main.go:30 +0xd8
27
```

Конечно, мы об этом сообщили: <https://github.com/go-yaml/yaml/issues/1050>

Фаззер подобрал ключ: несоответствие контракту

```
[  
    0x34, 0x30, 0x79, 0x3a,  
    0x2d, 0x0d, 0x0d, 0x23,  
    0x2d, 0x2d,  
]
```

yaml.v3@v3.0.1/yaml.go ×

```
85 // See the documentation of Marshal for the format of tags and a list of  
86 // supported tag options.  
87 //  
88 func Unmarshal(in []byte, out interface{}) (err error) {  
89     return unmarshal(in, out, strict: false)  
90 }
```

```
0x0d, 0x0d, 0x23, 0x2d,  
0x2d
```

```
]
```

При любых ошибках – возвращается ошибка.

Жизнь уязвимостей в OSS /1



24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

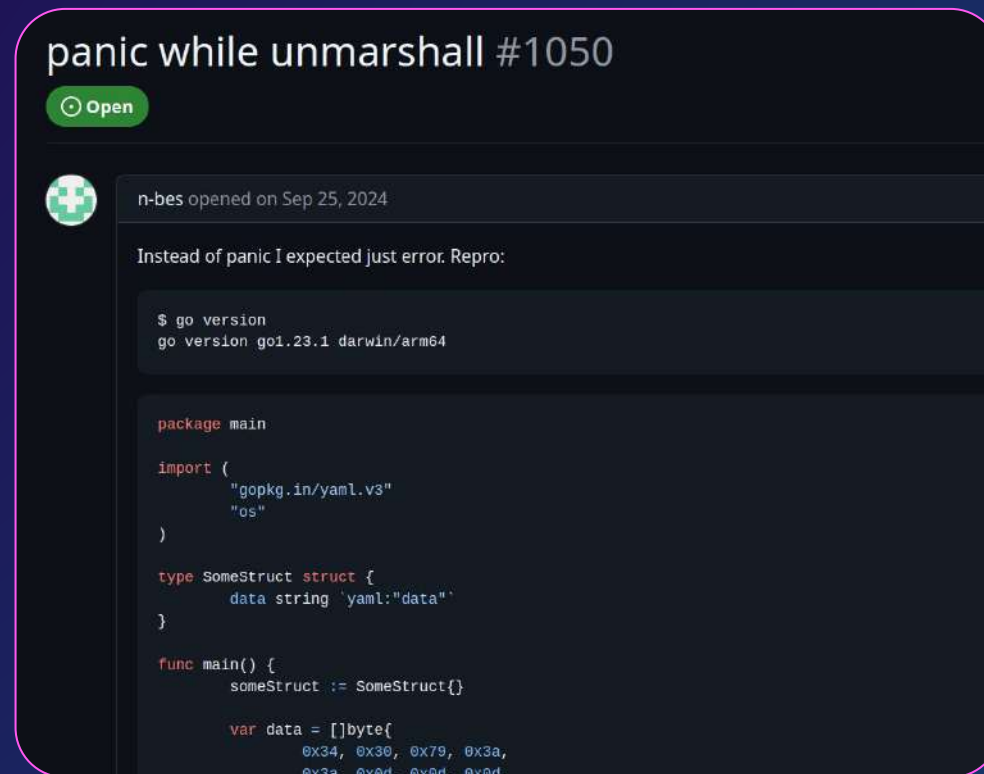
35	35		return nil, err
36	36		}
37		-	err = yaml.Unmarshal(content, &yaml)
	37	+	err = utils.SafeYamlUnmarshal(content, &yaml)
38	38		if err != nil {
39	39		return nil, err
40	40		}

Как фиксируется ошибка (наш вариант)

Жизнь уязвимостей в OSS /2

24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24 - раскрытие авторам go-yaml: <https://github.com/go-yaml/yaml/issues/1050>



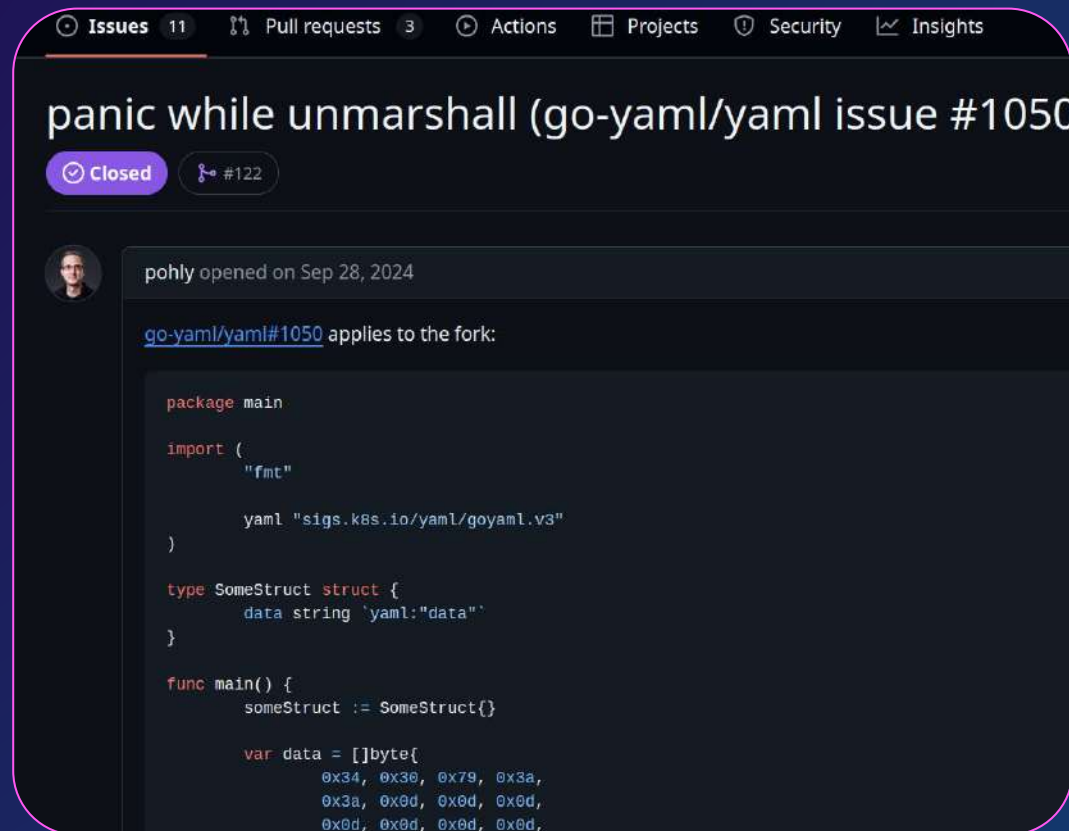
Жизнь уязвимостей в OSS /3



24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24 - раскрытие авторам go-yaml: <https://github.com/go-yaml/yaml/issues/1050>

28.09.24 - открывается задача в форке go-yaml: <https://github.com/kubernetes-sigs/yaml/issues/117>



Жизнь уязвимостей в OSS /4



24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24 - раскрытие авторам go-yaml: <https://github.com/go-yaml/yaml/issues/1050>

28.09.24 - открывается задача в форке go-yaml: <https://github.com/kubernetes-sigs/yaml/issues/117>

04.12.24 - находим схожую CVE-2022-28948, но про < go-yaml.v3.0.0 (недопатчили?)

Severity

High 7.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	None
Availability	High

[Learn more about base metrics](#)

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Жизнь уязвимостей в OSS /5



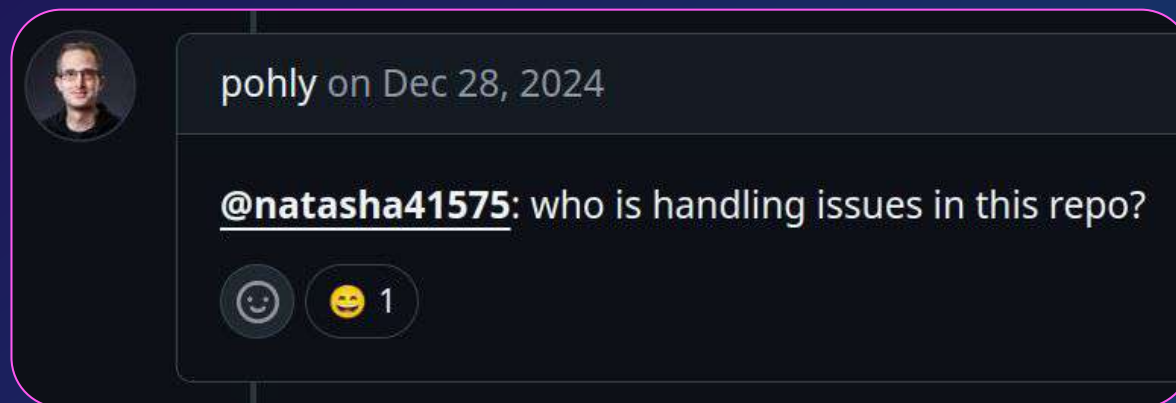
24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24 - раскрытие авторам go-yaml: <https://github.com/go-yaml/yaml/issues/1050>

28.09.24 - открывается задача в форке go-yaml: <https://github.com/kubernetes-sigs/yaml/issues/117>

04.12.24 - находим схожую CVE-2022-28948, но про < go-yaml.v3.0.0 (недопатчили?)

28.12.24 - вопрос мейнтейнеров форка: “who is handling issues in this repo?”



Жизнь уязвимостей в OSS /6

24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24 - раскрытие авторам go-yaml: <https://github.com/go-yaml/yaml/issues/1050>

28.09.24 - открывается задача в форке go-yaml: <https://github.com/kubernetes-sigs/yaml/issues/117>

04.12.24 - находим схожую CVE-2022-28948, но про < go-yaml.v3.0.0 (недопатчили?)

28.12.24 - вопрос мейнтейнеров форка: “who is handling issues in this repo?”

11.01.25 - появляется патч в форке

```
959 + func (d *decoder) setPossiblyUnhashableKey(m map[interface{}]bool, key interface(),
value bool) {
960 +     defer func() {
961 +         if err := recover(); err != nil {
962 +             failf("%v", err)
963 +         }
964 +     }()
965 +     m[key] = value
966 + }
967 +
968 + func (d *decoder) getPossiblyUnhashableKey(m map[interface{}]bool, key interface())
bool {
969 +     defer func() {
970 +         if err := recover(); err != nil {
971 +             failf("%v", err)
972 +         }
973 +     }()
974 +     return m[key]
975 + }
976 +
959 func (d *decoder) merge(parent *Node, merge *Node, out reflect.Value) {
960     mergedFields := d.mergedFields
961     if mergedFields == nil {
962         d.mergedFields = make(map[interface{}]bool)
963         for i := 0; i < len(parent.Content); i += 2 {
964             k := reflect.New(ifaceType).Elem()
965             if d.unmarshal(parent.Content[i], k) {
966 -                 d.mergedFields[k.Interface()] = true
967         }
968     }
977 func (d *decoder) merge(parent *Node, merge *Node, out reflect.Value) {
978     mergedFields := d.mergedFields
979     if mergedFields == nil {
980         d.mergedFields = make(map[interface{}]bool)
981         for i := 0; i < len(parent.Content); i += 2 {
982             k := reflect.New(ifaceType).Elem()
983             if d.unmarshal(parent.Content[i], k) {
984 +                 d.setPossiblyUnhashableKey(d.mergedFields,
k.Interface(), true)
985         }
986     }
```

<https://github.com/kubernetes-sigs/yaml/pull/122/files>

Жизнь уязвимостей в OSS /7



24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24 - раскрытие авторам go-yaml: <https://github.com/go-yaml/yaml/issues/1050>

28.09.24 - открывается задача в форке go-yaml: <https://github.com/kubernetes-sigs/yaml/issues/117>

04.12.24 - находим схожую CVE-2022-28948, но про < go-yaml.v3.0.0 (недопатчили?)

28.12.24 - вопрос мейнтейнеров форка: “who is handling issues in this repo?”

11.01.25 - появляется патч в форке

13.02.25 - патч вливается, но не входит в релиз

Жизнь уязвимостей в OSS /8

24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24

BDU:2025-02344

28.09.24

Вид ▾

04.12.24

28.12.24

Описание уязвимости Уязвимость функции Unmarshal библиотеки YAML языка программирования Go связана с недостатками механизма десериализации. Эксплуатация уязвимости может позволить нарушителю, действующему удалённо, вызвать отказ в обслуживании

11.01.25

13.02.25

14.02.25 - Никита отправляет заявку в БДУ ФСТЭК

05.03.25 - получаем ID — BDU:2025-02344

Жизнь уязвимостей в OSS /9&10



24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24 - раскрыт



28.09.24 - открыва

04.12.24 - находим

28.12.24 - вопрос

11.01.25 - появляется патч в форке

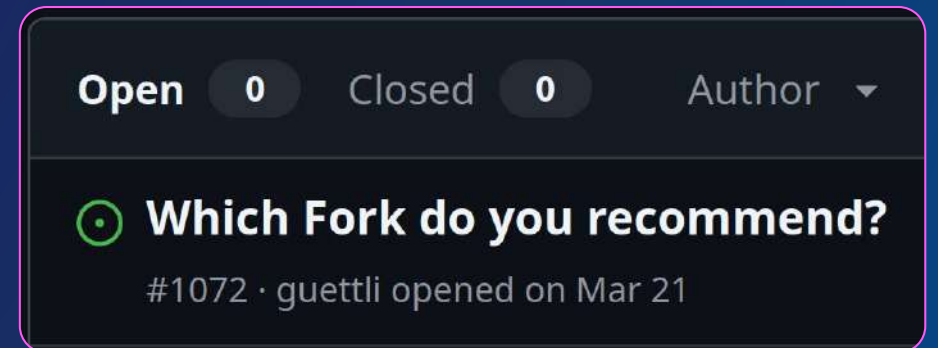
13.02.25 - патч вливается, но не входит в релиз

14.02.25 - Никита отправляет заявку в БДУ ФСТЭК

05.03.25 - получаем ID — BDU:2025-02344

10.03.25 - делаем PoC на helm и показываем на HackerOne [Kubernetes CNA]

01.04.25 - закрывается поддержка основного go-yaml



Жизнь уязвимостей в OSS /all



24.09.24 - обнаружение проблемы в go-yaml.v3.0.1 и само-патчинг в своем ближайшем релизе

25.09.24 - раскрытие авторам go-yaml: <https://github.com/go-yaml/yaml/issues/1050>

28.09.24 - открывается задача в форке go-yaml: <https://github.com/kubernetes-sigs/yaml/issues/117>

04.12.24 - находим схожую CVE-2022-28948, но про < go-yaml.v3.0.0 (недопатчили?)

28.12.24 - вопрос мейнтейнеров форка: “who is handling issues in this repo?”

11.01.25 - появляется патч в форке

13.02.25 - патч вливается, но не входит в релиз

14.02.25 - Никита отправляет заявку в БДУ ФСТЭК

05.03.25 - получаем ID — BDU:2025-02344

10.03.25 - делаем PoC на helm и показываем на HackerOne [Kubernetes CNA]

01.04.25 - закрывается поддержка основного go-yaml

Жизнь уязвимостей в OSS /++



03.04.25 - отправлена заявка на регистрацию в CVE

18.04.25 - ping по CVE, но в ответ тишина

Thank you for your request. You will be receiving a confirmation email shortly. If you need to reach the CVE team, please respond to the confirmation email you received and do not change the subject line. Your ticket number will be included in the email and that is how we will track your request. Thank you.

Information provided on the web form will remain private until the CVE Team receives notice that the vulnerability has been made public. Please use the "Notify CVE about a publication" form to notify us about a publication.

Please note: One confirmation message is sent for multiple CVE ID requests in a single browser session, with a single ticket number for your ease of reference. If you plan to request additional CVE IDs, and would like a different ticket number for your next request, please close your browser and open a new session.

[Submit Another CVE Request](#)

— А что там с *PoS'ом*?



PoC на helm, но не на всё

Получилось *показать*, что можно положить helm plugin list.

Получилось показать, что, *невозможно положить*: k8s, ArgoCD, fluxCD, etc, потому, они применяют конвертацию yaml в json.

- Get helm version:

```
Code 150 Bytes Unwrap lines Copy Download
1 $ helm version
2
3 version.BuildInfo{Version:"v3.17.2", GitCommit:"cc0bbbd6d6276b83880042c1ecb34087e84d41eb",
GitTreeState:"clean", GoVersion:"gol.24.1"}
```

- Prepare yaml (also reproduced with empty file):

```
Code 353 Bytes Unwrap lines Copy Download
1 $ hexdump -C hello-helm-plugin/plugin.yaml
2
3 00000000 34 30 79 3a 3a 0d 0d 0d 0d 0d 0d 0d 0d 3c 3c 3a |40y:~.....<<:|
4 00000010 0d 0d 0d 0d 0d 0d 0d 2d 20 20 2d 20 23 0d 3f 0d |.....- -#.?.|
5 00000020 23 0d 0d 23 2d 0d 20 2d 0d 3f 0d 23 0d 0d 23 2d |#..#-.-.?..#-|
6 00000030 2d | -|
7 00000031
```

- Install plugin:

```
Code 154 Bytes Unwrap lines Copy Download
1 $ helm plugin install hello-helm-plugin
2
3 Error: plugin is installed but unusable: invalid plugin name at "../helm/plugins/hello-helm-
plugin/plugin.yaml"
```

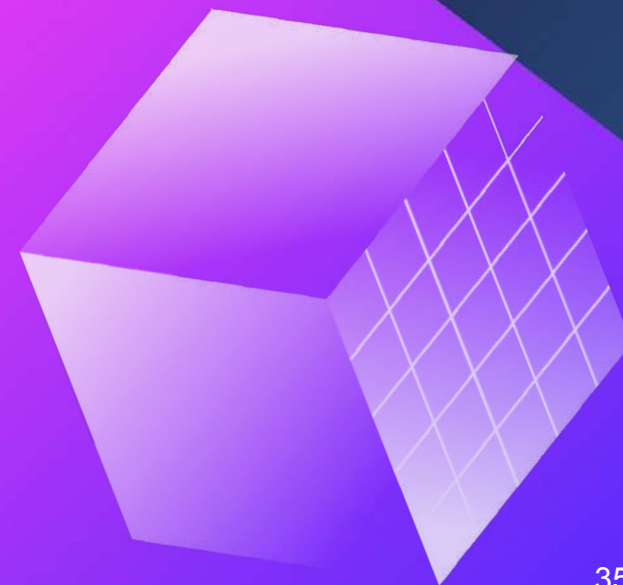
- Command `helm plugin list` not working anymore (other commands are working):

```
Code 196 Bytes Unwrap lines Copy Download
1 $ helm plugin list
2
3 failed to load plugins: invalid plugin name at "../helm/plugins/hello-helm-plugin/plugin.yaml"
4 Error: invalid plugin name at "../helm/plugins/hello-helm-plugin/plugin.yaml"
```

04



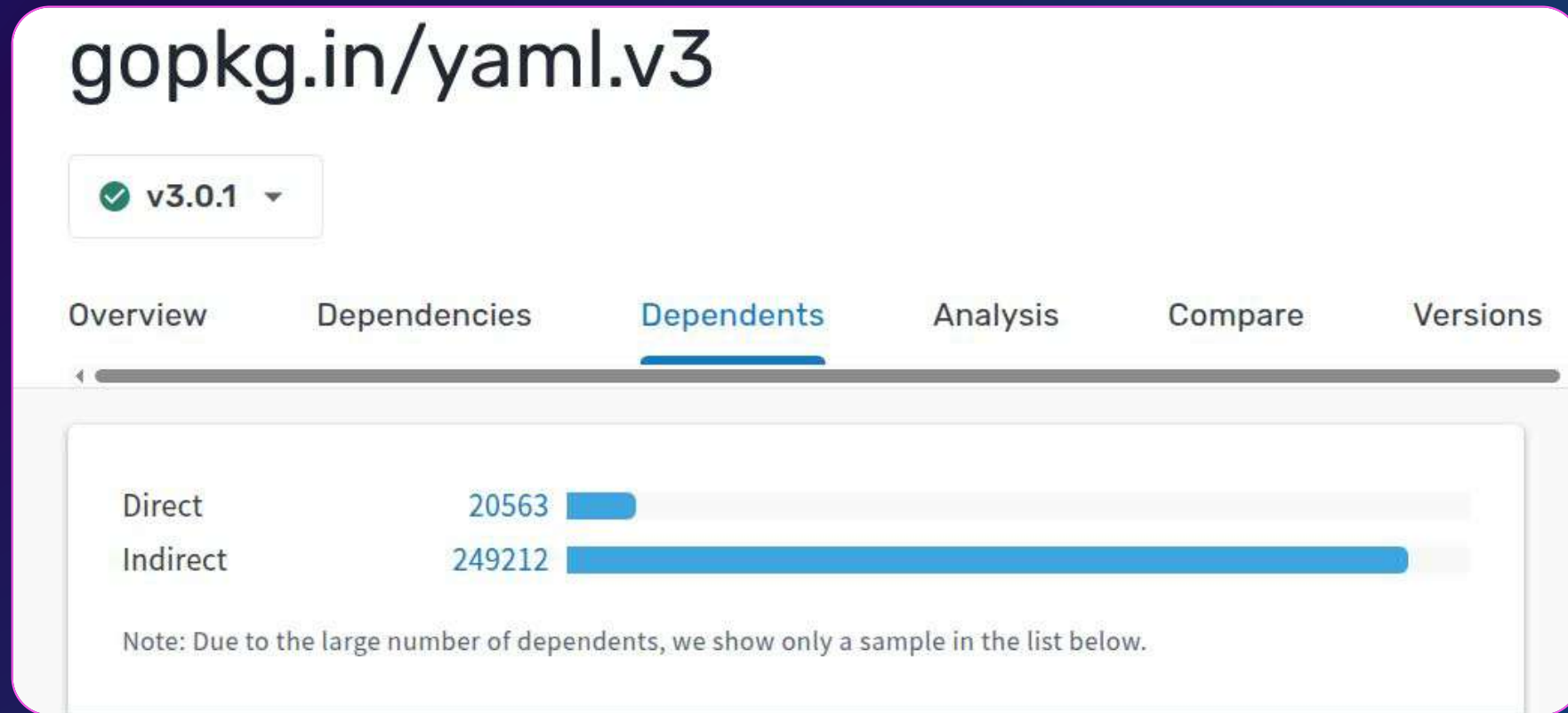
Эффект проблемы для сообщества



Мегаграф для Go
у нас пока в процессе,
но *поверим результатам [deps.dev](https://github.com/depgraph/depgraph)*



Последняя версия пакета



<https://deps.dev/go/gopkg.in%2Fyaml.v3/v3.0.1/dependents>

Последняя версия *форка* пакета



sig.s.k8s.io/yaml

✓ v1.4.0 ▾

Overview Dependencies Dependents Analysis Compare Versions

Direct	4333	<div style="width: 10%;"></div>
Indirect	28839	<div style="width: 80%;"></div>

Note: Due to the large number of dependents, we show only a sample in the list below.

<https://deps.dev/go/sigs.k8s.io%2Fyaml/v1.4.0/dependents>

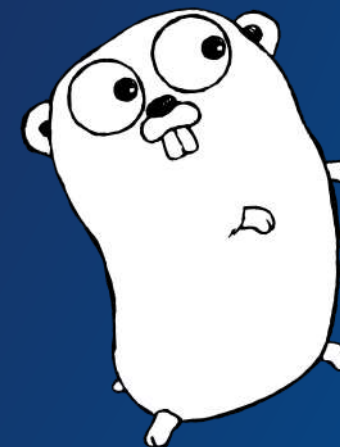
«Зацепление» экосистемы до- пакетов, 1.3 млн. шт.



Пакет	Зависимых пакетов директив	Зависимых пакетов транзитив
gorkg.in/yaml.v3.0.1 (<i>архив</i>)	20 563	249 212
sigs.k8s.io/yaml.v1.4.0 (<i>форк</i>)	4 033	28 839
+иные форки	да	да

Обнаруженная нами

VDU:2025-02344 задевает 23.3% Go-
экосистемы



05



Выводы



Наблюдаемые факторы* *ускорения* патча

- ✦ *Понятный* репортинг авторам
- ✦ *Отстаивание* интересов
- ✦ *Получение идентификатора* в фидах
- ✦ Привлечение *тех кому не всё равно*
- ✦ Медиа-освещение

Экосистема	2021	2022	2023
JavaScript	130	31	27
Java	180	97	39
C#	90	85	80
Python	25	56	30

Скорость патчинга после раскрытия уязвимости (дни)

(с) «Проблемы отцов и детей: Аналитика и триаж транзитивных зависимостей», PHD, 2024

* Работают не всегда.

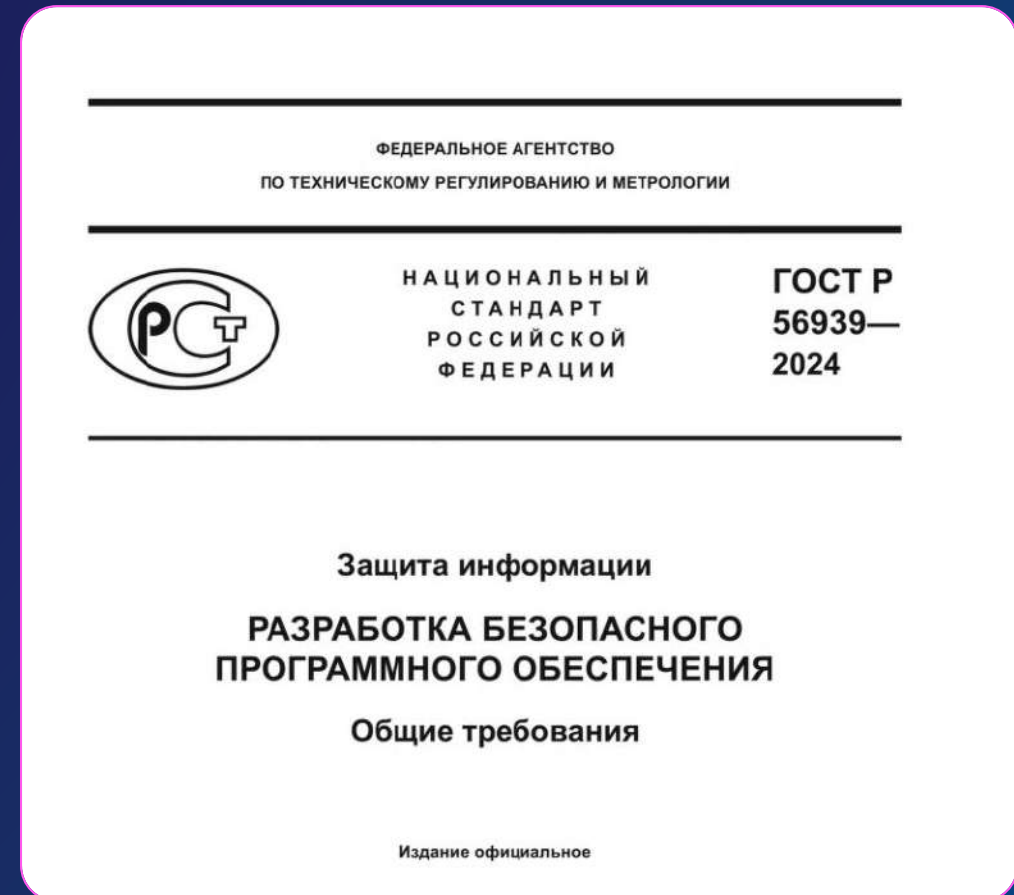
Наблюдение

*Закрадываются подозрения, что
кроме нас в БДУ никто не смотрит :)*

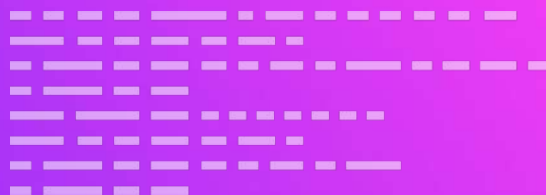
Практики безопасной разработки *никто не отменял*

- ❖ Статический анализ /SAST
- ❖ Динамический анализ /DAST / fu**ing
- ❖ Композиционный анализ /SCA

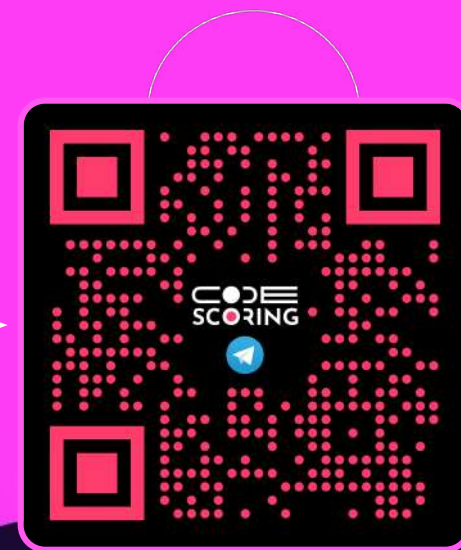
Ну и почитать *ГОСТ 56939-2024* →



Спасибо!



Следите за нами →



Алексей
alexey@codescoring.ru

Никита
n.besperstov@codescoring.ru

