

# IN{ERA} SECURITY}

безопасность встроенная в ДНК

[www.InfEraSecurity.ru](http://www.InfEraSecurity.ru)

тел.: +7 915 234 9900

почта: [info@InfEraSecurity.ru](mailto:info@InfEraSecurity.ru)



# INFERA Security. О компании

## новая Эра информационной безопасности

экспертиза людей **сильнее**  
**«возраста компании»**

«*Наша команда — это не стартап без опыта — это консолидация лучших практик, которые мы десятилетиями оттачивали в крупных ИБ-компаниях, проектах для госструктур и корпораций.*

*Наши ключевые эксперты 15+ лет решают реальные ИБ-кейсы — от защиты крупнейших компаний России до развития ИБ-продуктов и ИБ-направлений в ведущих интеграторах и вендорах.*



{ Игорь Бирюков  
Генеральный директор InfEra Security }

15+ лет в ИБ

- ех-руководитель Киберхаба «Сколково»
- автор статей для ведущих СМИ, спикер и модератор крупнейших ИБ-конференций
- амбассадор клуба резидентов «Кибердома», член клуба КУБИТ
- реализовывал государственные проекты по ИБ, руководил отделами технической защиты и ИБ-консалтингом в ведущих компаниях
- большой опыт получения лицензий ФСТЭК и ФСБ, знание законодательства и нормативных документов ФСТЭК России, ФСБ России, Минцифры

# INFERA Security. Продуктовый портфель

безопасность встроенная, а не прикрученная



добавляем экспертизу ИБ в R&D без посредников

защищаем ИИ-модели и бизнес-данные



## INFERA AI.SafeCode

ВСТРОЕННЫЙ AI.SECURITY АГЕНТ В ПРОЦЕССЫ РАЗРАБОТКИ

ИИ-платформа для ИБ-анализа исходного кода с целью обнаружения уязвимостей и аномалий в режиме реального времени, внедрения подхода Security by Design в R&D, в процессы разработки ПО и использования open-source компонентов



## INFERA AI.Firewall

ЗАЩИТА ИИ-МОДЕЛЕЙ И ИХ БЕЗОПАСНОЕ ИСПОЛЬЗОВАНИЕ

решение для защиты и безопасного использования LLM-моделей, включая контроль доступа и защиту конфиденциальных данных, фильтрацию вредоносных запросов и аудит взаимодействия с ИИ

## КОНСАЛТИНГ И УСЛУГИ

подготовка документации и внедрение процесса DevSecOps, определение метрик, настройка и интеграция DevSecOps инструментов, настройка Security Gates, разработка стратегии защиты LLM

## ТЕХНИЧЕСКАЯ ПОДДЕРЖКА

- стандартная 8x5
- расширенная 24x7

# INFERA AI.SafeCode



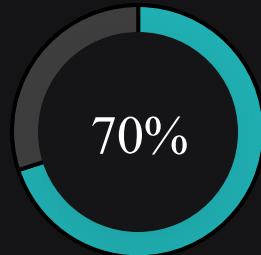
встроенный  
AI.SECURITY агент  
в процессы разработки

ИИ-платформа для ИБ-анализа исходного кода с  
целью обнаружения уязвимостей и аномалий в  
режиме реального времени, внедрения подхода  
**Security by Design** в R&D, в процессы разработки ПО  
и использования open-source компонентов



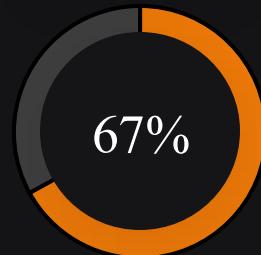
# Почему нужен DevSecOps?

классический подход безопасности не работает в современных скоростях DevOps



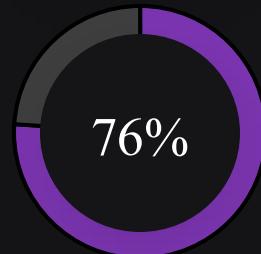
времени уходит на ручную проверку уязвимостей и багов без DevSecOps у безопасников и разработчиков

*GitLab DevSecOps Survey*



разработчиков отмечают рост числа уязвимостей в ПО за последний год

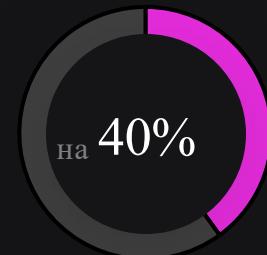
*GitLab Global DevSecOps Report 2024*



CISO признают, что текущие процессы безопасности не успевают за DevOps-циклами и остаются «заплаточными»

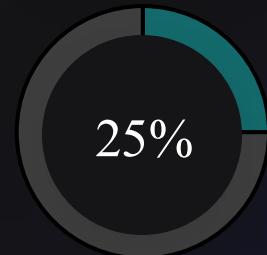
*Gartner Research*

только автоматизация и Shift Left Security позволяют командам безопасно выпускать ПО в высоком темпе



к 2027г. увеличится количество команд R&D, использующих ИИ

*Gartner*



дефектов ПО, попавших в производство, будут вызваны **отсутствием контроля за кодом, сгенерированным ИИ**

*Gartner*

ГОСТ Р 56939-2024, GDPR, ISO 27001, PCI DSS

все требуют, чтобы **безопасность была встроена в процесс разработки, а не добавлялась постфактум**

## анализ кода в реальном времени

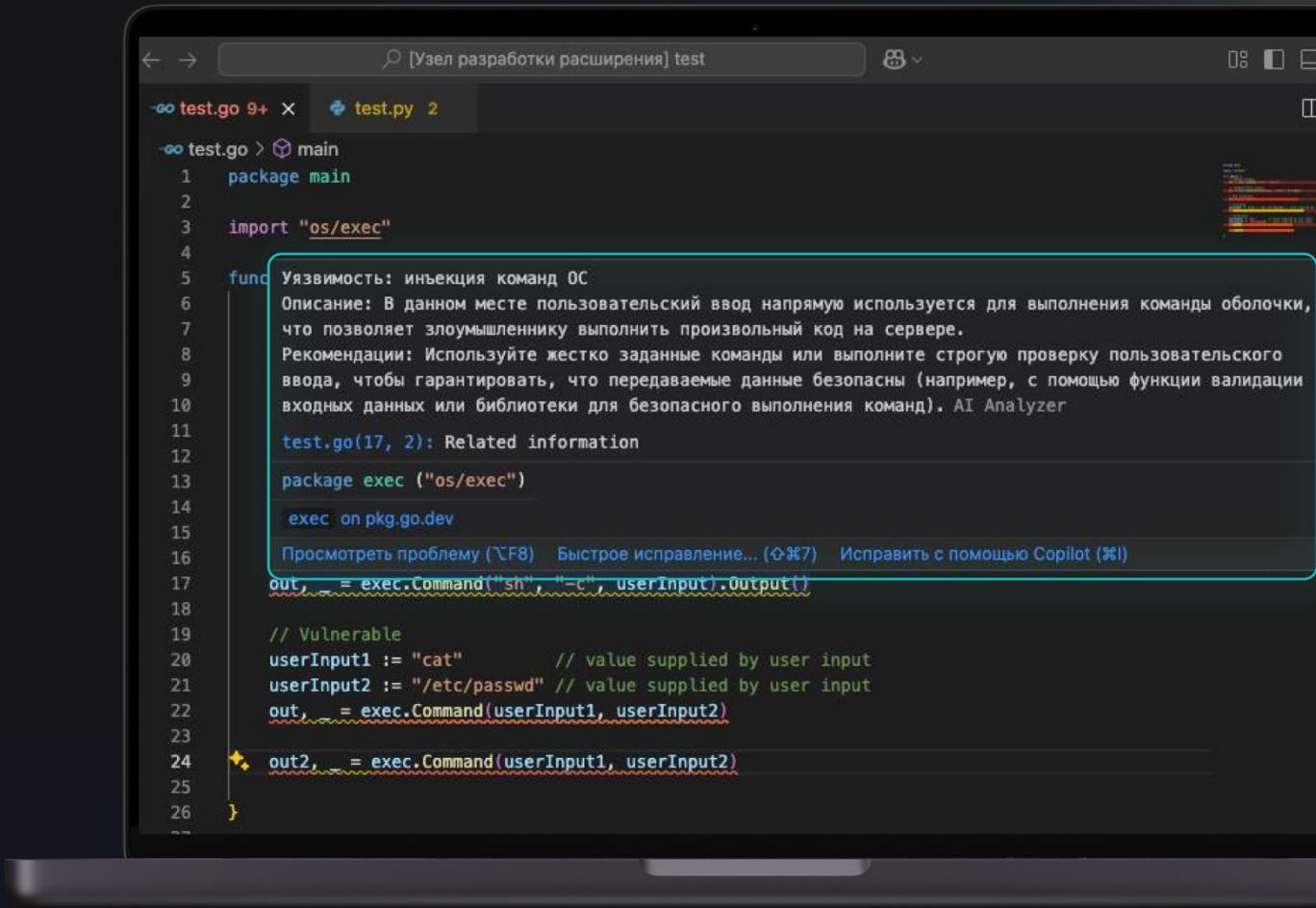
- INFERA AI.SafeCode анализирует код программы на наличие уязвимостей в процессе его написания разработчиком
- в том числе осуществляется проверка кода open-source компонентов перед их использованием в отношении ПО, поставляемого на объекты КИИ

## ИИ-подход

- LLM распознает сложные паттерны и предлагает варианты исправления

## широкая поддержка языков программирования

Python, Java, JavaScript, C++, C#, Typescript, SQL, C, Go, PHP и других



The screenshot shows a code editor interface with two tabs: 'test.go' and 'test.py'. The 'test.go' tab is active, displaying the following code:

```
1 package main
2
3 import "os/exec"
4
5 func main() {
6     // Vulnerable
7     userInput1 := "cat"           // value supplied by user input
8     userInput2 := "/etc/passwd" // value supplied by user input
9
10    out, _ = exec.Command(userInput1, userInput2).Output()
11
12    // Vulnerable
13    out2, _ = exec.Command(userInput1, userInput2).Output()
14}
```

A tooltip is displayed over the line 'out, \_ = exec.Command("cat", "-c", userInput).Output()', highlighting it as a 'Vulnerable' command. The tooltip contains the following information:

Уязвимость: инъекция команд ОС  
Описание: В данном месте пользовательский ввод напрямую используется для выполнения команды оболочки, что позволяет злоумышленнику выполнить произвольный код на сервере.  
Рекомендации: Используйте жестко заданные команды или выполните строгую проверку пользовательского ввода, чтобы гарантировать, что передаваемые данные безопасны (например, с помощью функции валидации входных данных или библиотеки для безопасного выполнения команд). AI Analyzer

test.go(17, 2): Related information

package exec ("os/exec")  
exec on pkg.go.dev

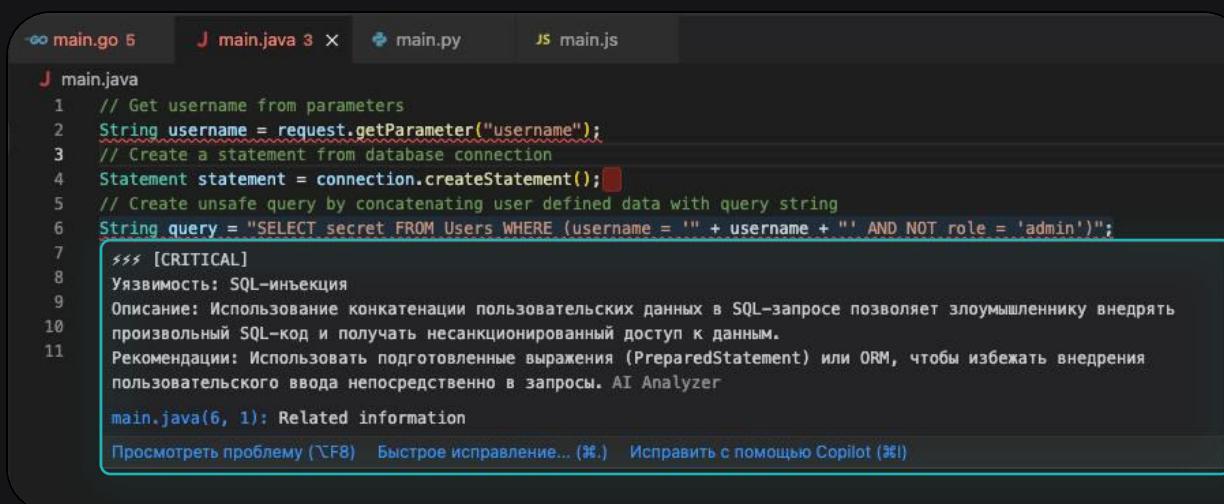
Просмотреть проблему (F8) Быстрое исправление... (Alt+F7) Исправить с помощью Copilot (⌘I)

## поиск уязвимостей в runtime режиме

в процессе написания кода

## интеграция с IDE

- установка плагина занимает считанные минуты
- бесшовно встраивается в процесс разработки и поддерживает распространённые IDE



The screenshot shows a Java code editor with tabs for main.go, main.java, main.py, and main.js. The main.java tab is active, displaying the following code:

```
1 // Get username from parameters
2 String username = request.getParameter("username");
3 // Create a statement from database connection
4 Statement statement = connection.createStatement();
5 // Create unsafe query by concatenating user defined data with query string
6 String query = "SELECT secret FROM Users WHERE (username = '" + username + "' AND NOT role = 'admin')";
7 sss [CRITICAL]
8 Уязвимость: SQL-инъекция
9 Описание: Использование конкатенации пользовательских данных в SQL-запросе позволяет злоумышленнику внедрять произвольный SQL-код и получать несанкционированный доступ к данным.
10 Рекомендации: Использовать подготовленные выражения (PreparedStatement) или ORM, чтобы избежать внедрения
11 пользовательского ввода непосредственно в запросы. AI Analyzer

main.java(6, 1): Related information
```

Below the code, there is a note: "Просмотреть проблему (⌘F8) Быстрое исправление... (⌘.) Исправить с помощью Copilot (⌘I)".

## поддерживаемые платформы

- LPT сервер работает на Linux
- Клиент-серверная модель

## LLM-интеграция

возможность размещения в открытом облаке и в on-premise

## результат

- безопасный код уже на этапе разработки — никаких сюрпризов на стадии тестирования и эксплуатации
- безопасный код без замедления релизов
- выпуск ПО для КИИ в соответствии с требованиями регуляторов

# INFERA AI.SafeCode. ИИ-модель



подход на основе ИИ включает:

- контроль уязвимостей на основе анализа исходного кода с помощью ИИ
- отображение разработчику ПО уязвимого участка кода, описание типа уязвимости, а также варианта для устранения
- факт обнаружения уязвимости отправляется в систему учёта уязвимостей
- при исправлении уязвимость перестаёт отображаться
- при необходимости, возможно добавить перечень ложных срабатываний для последующей фильтрации

The screenshot shows a code editor interface with three tabs: test.txt, test.go, and test.py. The test.py tab contains the following code:

```
test.py > get_user_data
1 import sqlite3
2
3 def get_user_data(user_id):
4     conn = sqlite3.connect("users.db")
5     cursor = conn.cursor()
6     query = f"SELECT * FROM users WHERE id = {user_id}"
7
8
9
10
```

A tooltip is displayed over the line of code at line 6, highlighting the SQL query: `f"SELECT * FROM users WHERE id = {user_id}"`. The tooltip content is as follows:

Уязвимость: SQL-инъекция  
Описание: Пользовательский ввод напрямую вставляется в SQL-запрос без экранирования или параметризации, что позволяет злоумышленнику внедрить вредоносный SQL-код и получить несанкционированный доступ к данным базы.  
Рекомендации: Использовать параметризованные запросы вместо прямой вставки пользовательского ввода.  
Например: cursor.execute("SELECT \* FROM users WHERE id = ?", (user\_id,)) AI Analyzer  
test.py(6, 5): Related information  
Просмотреть проблему (⌘F8) Быстрое исправление... (⌘.) Исправить с помощью Copilot (⌘I)

в INFERA AI.SafeCode машинное обучение и искусственный интеллект используются для улучшения обнаружения уязвимостей и эффективной расстановки приоритетов

The screenshot shows two code editor windows with analysis results from AI.SafeCode.

**Top Window (main.go):**

```

1 import (
2     "fmt"
3     "os"
4     "net"
5 )
6 func main() {
7     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
8         cmd := r.URL.Query().Get("cmd")
9         if cmd == "" {
10            w.Write([]byte("Usage: http://localhost:8080/?cmd=ls"))
11        } else {
12            out, err := exec.Command("bash", "-c", cmd).Output()
13            if err != nil {
14                fmt.Fprintf(w, "Ошибка: %s", err)
15                return
16            }
17            fmt.Fprintf(w, "Результат: %s", out)
18        }
19    })
20    http.ListenAndServe(":8080", nil)
21 }

```

Annotations:

- Line 1:** [CRITICAL] Уязвимость: Командная инъекция
- Line 10:** Описание: Параметр 'cmd' получают непосредственно из пользовательского ввода (GET-параметр), после чего он передается в командную строку без какой-либо проверки, фильтрации или экранирования. Это позволяет злоумышленнику выполнять произвольные команды в системе, что полностью компрометирует сервер.
- Line 11:** Рекомендации: Не использовать пользовательский ввод в exec.Command без строгой валидации и разрешения только безопасных команд. В идеале отказаться от передачи командной строки из пользовательского ввода или реализовать белый список разрешенных команд. AI Analyzer
- Line 12:** main.go(10, 3): Related information
- Line 13:** Просмотреть проблему (F8) Быстрое исправление... (⌘.) Исправить с помощью Copilot (⌘I)

**Bottom Window (main.go):**

```

1 import (
2     "fmt"
3     "os/exec"
4     "net/http"
5 )
6 func main() {
7     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
8         cmd := r.URL.Query().Get("cmd")
9         if cmd == "" {
10            w.Write([]byte("Usage: http://localhost:8080/?cmd=ls"))
11        } else {
12            out, err := exec.Command("bash", "-c", cmd).Output()
13            if err != nil {
14                fmt.Fprintf(w, "Ошибка: %s", err)
15                return
16            }
17            fmt.Fprintf(w, "Результат: %s", out)
18        }
19    })
20    http.ListenAndServe(":8080", nil)
21 }

```

Annotations:

- Line 1:** [WARNING] Уязвимость: Раскрытие чувствительной информации
- Line 10:** Описание: Текст ошибки из exec.Command выводится напрямую пользователю. Это может раскрыть детали внутренней инфраструктуры, например, структуру файловой системы, системные сообщения или переменные среды.
- Line 11:** Рекомендации: Не выдавать пользователю внутренние ошибки сервера. Логировать подробные ошибки только на сервере, пользователю возвращать обобщенное сообщение. AI Analyzer
- Line 12:** main.go(12, 4): Related information
- Line 13:** Просмотреть проблему (F8) Быстрое исправление... (⌘.) Исправить с помощью Copilot (⌘I)

добавляем экспертизу ИБ в R&D без посредников

# Соответствие требованиям законодательства

DevSecOps помогает автоматизировать выполнение требований:

- ГОСТ Р 56939-2024
- ФЗ-152, ФЗ-187
- ISO 27001, PCI DSS, GDPR

ГОСТ Р 56939-2024 — это не просто compliance-документ, а фундамент для внедрения DevSecOps-подхода в российском контексте. Он задаёт минимальный уровень зрелости безопасной разработки и побуждает автоматизировать, моделировать и интегрировать ИБ в каждую фазу разработки ПО



связь ГОСТ Р 56939-2024 с DevSecOps

элемент DevSecOps	↗	ГОСТ Р 56939-2024	↗
Shift Left		требование начинать ИБ с первых этапов проекта	
CI/CD Security		контроль сборки, статический и динамический анализ	
Security as Code		использование IaC + политик как код	
SBOM и supply chain		учет происхождения компонентов, требования к цепочке поставок	
Автоматизация ИБ		рекомендуется в тестировании, сопровождении, мониторинге	
Моделирование угроз		обязательный элемент проектирования	
DevSecOps-чемпионы		ГОСТ требует наличия ответственных за ИБ в проектных ролях	

# INFERA AI.SafeCode. Преимущество использования

обеспечение безопасности на всех этапах разработки и эксплуатации ПО

## ускорение выпуска ПО без жертв для ИБ

- продукт выходит быстрее и безопаснее
- ИИ встроенная в среду разработки позволяет делать проверки автоматически
- команда R&D не ждёт согласований от InfoSec

## уменьшение затрат на устранение уязвимостей

- исправление бага на стадии продакшена может стоить в 10–30 раз дороже, чем на стадии разработки
- сокращение числа инцидентов безопасности до 40–50% при грамотной автоматизации тестов и проверки кода

## устойчивость к supply chain атакам

- снижается киберриск атаки на цепочку поставок (проверка интеграций и подключений)
- безопасная интеграция с другими системами

## соблюдение требований регуляторов

- ГОСТ Р 56939-2024, ФЗ-152, ФЗ-187 и др.

## обнаружение уязвимостей в open-source

- безопасность должна быть встроена в процесс разработки и в проверку используемых компонентов

## безопасное ПО для КИИ

- независимая проверка исходного кода open-source решений перед их использованием в отношении критически важного ПО

INFERA AI.SafeCode — это инвестиции в снижение рисков, ускорение релизов и цифровую устойчивость бизнеса



безопасность встроенная в ДНК

спасибо за внимание

[www.InfEraSecurity.ru](http://www.InfEraSecurity.ru)

тел.: +7 915 234 9900

почта: [info@InfEraSecurity.ru](mailto:info@InfEraSecurity.ru)

