Управление уязвимостями и РБПО: Взгляд со стороны разработчика.

Александр Дубинин
Группа компаний YADRO

О компании YADRO

YADRO — российская технологическая компания (входит в «ИКС Холдинг»), объединяющая направления разработки и производства вычислительных платформ, систем обработки и хранения данных, телекоммуникационного и сетевого оборудования, персональных и «умных» устройств, микропроцессорных ядер и fabless-разработку микропроцессоров. R&D центры расположены в Москве, Санкт-Петербурге, Екатеринбурге, Нижнем Новгороде и Минске.

О компании YADRO: yadro.com/ru/company

Продукты YADRO: yadro.com/ru/products

Портал инженерной культуры: engineer.yadro.com

Что у нас нового?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)

Что у нас нового?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)
- Проработанная архитектура (как и всегда - следуем принципам конструктивизма и функционального минимализма)

Что у нас нового?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)
- Проработанная архитектура (как и всегда - следуем принципам конструктивизма и функционального минимализма)
- Знаем каждый байт нашего красивого и надежного кода (SCA/SAST/DAST)

Что у нас нового?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)
- Проработанная архитектура (как и всегда - следуем принципам конструктивизма и функционального минимализма)
- Знаем каждый байт нашего красивого и надежного кода (SCA/SAST/DAST)
- Документируется все: от PRD(ТЗ), HLA/HLD (ЭП/ТП) до конкретной реализации

Что у нас нового?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)
- Проработанная архитектура (как и всегда - следуем принципам конструктивизма и функционального минимализма)
- Знаем каждый байт нашего красивого и надежного кода (SCA/SAST/DAST)
- Документируется все: от PRD(ТЗ), HLA/HLD (ЭП/ТП) до конкретной реализации
- **Регулярная работа с уязвимостями и прочими дефектами**

Что у нас нового?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)
- Проработанная архитектура (как и всегда - следуем принципам конструктивизма и функционального минимализма)
- Знаем каждый байт нашего красивого и надежного кода (SCA/SAST/DAST)
- Документируется все: от PRD(ТЗ), HLA/HLD (ЭП/ТП) до конкретной реализации
- Регулярная работа с уязвимостями и прочими дефектами

А еще мы получили возможность подтвердить качество и надежность наших продуктов сертификатами ФСТЭК!

Исторические особенности Enterprise продуктов

Исторически Enterprise решения:

- Крайне консервативная область, множество СХД, например, работают годами без обновлений и исправлений уязвимостей (принцип «Работает - не трожь!»);
- Конечные пользователи практически не работают с самими продуктами и их интерфейсами управления напрямую;
- В основном применяются компенсирующие меры: средства наложенной защиты, изолированные подсети, МСЭ и т.п.;
- Зарубежные вендоры сформировали устойчивое мнение, что «так и должно быть».

Пора меняться!

Современный Enterprise куда более разнообразен:

- Объектные СХД, например, могут отдавать данные непосредственно пользователям в сети Интернет (ставить WAF между ними и пользователями - неэффективно);

Пора меняться!

Современный Enterprise куда более разнообразен:

- Объектные СХД, например, могут отдавать данные непосредственно пользователям в сети Интернет (ставить WAF между ними и пользователями - неэффективно);
- Концепция «Безопасности нулевого доверия» («Zero Trust Security») все популярнее;

Пора меняться!

Современный Enterprise куда более разнообразен:

- Объектные СХД, например, могут отдавать данные непосредственно пользователям в сети Интернет (ставить WAF между ними и пользователями - неэффективно);
- Концепция «Безопасности нулевого доверия» («Zero Trust Security») все популярнее;
- SDS, SDC, SDN... Модно, молодежно, сложно... и более уязвимо;

Пора меняться!

Современный Enterprise куда более разнообразен:

- Объектные СХД, например, могут отдавать данные непосредственно пользователям в сети Интернет (ставить WAF между ними и пользователями - неэффективно);
- Концепция «Безопасности нулевого доверия» («Zero Trust Security») все популярнее;
- SDS, SDC, SDN... Модно, молодежно, сложно... и более уязвимо;
- Растет разнообразие угроз и способов их реализации.

Что мы делаем с уязвимостями и причем тут РБПО?

Процесс управления недостатками и обновлениями существовал и ранее, хотя:

- Обычно уязвимости требовали эскалации к R&D для их анализа и устранения;
- До внедрения инструментов РБПО (особено SCA), мы узнавали об уязвимостях «пост-фактум».

Что мы делаем с уязвимостями и причем тут РБПО?

Процесс управления недостатками и обновлениями существовал и ранее, хотя:

- Обычно уязвимости требовали эскалации к R&D для их анализа и устранения;
- До внедрения инструментов РБПО (особено SCA), мы узнавали об уязвимостях «пост-фактум».

По мере внедрения практик SDL, процес работы с дефектами и уязвимостями менялся. Мы обнаружили, что он практически совпал с процессами в ГОСТ 56939-2024, но:

- Дефектов, найденых SAST/DAST совсем немного (большинство исправлены в процессе разработки, еще до выпуска релиза);

Что мы делаем с уязвимостями и причем тут РБПО?

Процесс управления недостатками и обновлениями существовал и ранее, хотя:

- Обычно уязвимости требовали эскалации к R&D для их анализа и устранения;
- До внедрения инструментов РБПО (особено SCA), мы узнавали об уязвимостях «пост-фактум».

По мере внедрения практик SDL, процес работы с дефектами и уязвимостями менялся. Мы обнаружили, что он практически совпал с процессами в ГОСТ 56939-2024, но:

- Дефектов, найденых SAST/DAST совсем немного (большинство исправлены в процессе разработки, еще до выпуска релиза);
- При большом потоке информации о потенциальных уязвимостях, требуется **быстрая** экспертная оценка и автоматизированая фильтрация нерелевантных случаев;

Что мы делаем с уязвимостями и причем тут РБПО?

Процесс управления недостатками и обновлениями существовал и ранее, хотя:

- Обычно уязвимости требовали эскалации к R&D для их анализа и устранения;
- До внедрения инструментов РБПО (особено SCA), мы узнавали об уязвимостях «пост-фактум».

По мере внедрения практик SDL, процес работы с дефектами и уязвимостями менялся. Мы обнаружили, что он практически совпал с процессами в ГОСТ 56939-2024, но:

- Дефектов, найденых SAST/DAST совсем немного (большинство исправлены в процессе разработки, еще до выпуска релиза);
- При большом потоке информации о потенциальных уязвимостях, требуется быстрая экспертная оценка и автоматизированая фильтрация нерелевантных случаев;
- Полное тестирование каждого изменения сложных продуктов может стать неадекватно ресурсоемким, нужны способы оптимизации.

Какие возникли сложности?

При практической реализации процессов работы с уязвимостями, мы столкнулись со следующим:

- С инструментами анализа поток информации об уязвимостях в компонентах огромен, первичные CVSS часто завышены;

Какие возникли сложности?

При практической реализации процессов работы с уязвимостями, мы столкнулись со следующим:

- С инструментами анализа поток информации об уязвимостях в компонентах огромен, первичные CVSS часто завышены;
- Если в коде найден дефект - это не значит, что он уязвимость и нужно срочно бежать и править;

Какие возникли сложности?

При практической реализации процессов работы с уязвимостями, мы столкнулись со следующим:

- С инструментами анализа поток информации об уязвимостях в компонентах огромен, первичные CVSS часто завышены;
- Если в коде найден дефект - это не значит, что он уязвимость и нужно срочно бежать и править;
- Ручной анализ потенциальных уязвимостей (а это и недостатки!) требует очень много времени и сил;

Какие возникли сложности?

При практической реализации процессов работы с уязвимостями, мы столкнулись со следующим:

- С инструментами анализа поток информации об уязвимостях в компонентах огромен, первичные CVSS часто завышены;
- Если в коде найден дефект - это не значит, что он уязвимость и нужно срочно бежать и править;
- Ручной анализ потенциальных уязвимостей (а это и недостатки!) требует очень много времени и сил;
- Устранение уязвимостей и других дефектов в заимствованных компонентах часто может оказать негативное влияние на функциональность и производительность продуктов, для выявления которого нужно глубокое тестирование.

Что нужно, чтобы не «захлебнуться»?

Из опыта работы и анализа, «выросли» такие условия:

- Требования надежности (включают безопасность!) должны быть заложены и учтены с самого начала;

Что нужно, чтобы не «захлебнуться»?

Из опыта работы и анализа, «выросли» такие условия:

- Требования надежности (включают безопасность!) должны быть заложены и учтены с самого начала;
- Необходима полная проработка сценариев использования «от и до»;

Что нужно, чтобы не «захлебнуться»?

Из опыта работы и анализа, «выросли» такие условия:

- Требования надежности (включают безопасность!) должны быть заложены и учтены с самого начала;
- Необходима полная проработка сценариев использования «от и до»;
- В продукте должно быть только то и только так, что и как нужно для выполнения требований (конструктивизм и функциональный минимализм);

Что нужно, чтобы не «захлебнуться»?

Из опыта работы и анализа, «выросли» такие условия:

- Требования надежности (включают безопасность!) должны быть заложены и учтены с самого начала;
- Необходима полная проработка сценариев использования «от и до»;
- В продукте должно быть только то и только так, что и как нужно для выполнения требований (конструктивизм и функциональный минимализм);
- Архитектура продукта должна позволять изменять его части максимально независимо от других, а имеющиеся зависимости должны быть однозначно определены;

Что нужно, чтобы не «захлебнуться»?

Из опыта работы и анализа, «выросли» такие условия:

- Требования надежности (включают безопасность!) должны быть заложены и учтены с самого начала;
- Необходима полная проработка сценариев использования «от и до»;
- В продукте должно быть только то и только так, что и как нужно для выполнения требований (конструктивизм и функциональный минимализм);
- Архитектура продукта должна позволять изменять его части максимально независимо от других, а имеющиеся зависимости должны быть однозначно определены;
- Автоматизация везде, где только возможно!

Так что же делать? Конкретно?

Выглядит правильным разделить решаемую задачу на несколько областей:

- Архитектура продукта - минимум компонентов, устойчивость к изменениям;
- Работа с источниками информации, первичная фильтрация;
- Эффективная организация процесса работы инженеров (от сервиса до разработчиков) с уязвимостями;
- Система сборки, тестирования и релизный цикл, позволяющие выпускать любые обновления так часто, как надо.

Архитектура продукта

Даже самый прекрасный процесс устранения недостатков забуксует (или будет слишком дорогим), если продукт спроектирован некорректно. Потому мы:

- При проектировании следуем принципам K.I.S.S., «UNIX Way», теории надежности;

Архитектура продукта

Даже самый прекрасный процесс устранения недостатков забуксует (или будет слишком дорогим), если продукт спроектирован некорректно. Потому мы:

- При проектировании следуем принципам K.I.S.S., «UNIX Way», теории надежности;
- Обязательно учитываем модель угроз и сценарии использования (негативные тоже);

Архитектура продукта

Даже самый прекрасный процесс устранения недостатков забуксует (или будет слишком дорогим), если продукт спроектирован некорректно. Потому мы:

- При проектировании следуем принципам K.I.S.S., «UNIX Way», теории надежности;
- Обязательно учитываем модель угроз и сценарии использования (негативные тоже);
- Обеспечиваем возможность расширения функций и независимого обновления, например, добавлением новых модулей, сервисов или даже плагинов;

Архитектура продукта

Даже самый прекрасный процесс устранения недостатков забуксует (или будет слишком дорогим), если продукт спроектирован некорректно. Потому мы:

- При проектировании следуем принципам K.I.S.S., «UNIX Way», теории надежности;
- Обязательно учитываем модель угроз и сценарии использования (негативные тоже);
- Обеспечиваем возможность расширения функций и независимого обновления, например, добавлением новых модулей, сервисов или даже плагинов;
- Каждый сервис, модуль или плагин - реализуется как «вещь в себе», минимально зависящий от среды исполнения (насколько это возможно).

Архитектура продукта

Даже самый прекрасный процесс устранения недостатков забуксует (или будет слишком дорогим), если продукт спроектирован некорректно. Потому мы:

- При проектировании следуем принципам K.I.S.S., «UNIX Way», теории надежности;
- Обязательно учитываем модель угроз и сценарии использования (негативные тоже);
- Обеспечиваем возможность расширения функций и независимого обновления, например, добавлением новых модулей, сервисов или даже плагинов;
- Каждый сервис, модуль или плагин - реализуется как «вещь в себе», минимально зависящий от среды исполнения (насколько это возможно).
- Среда исполнения содержит только то, что нужно для работы функционального слоя (подход встраиваемых систем, в противовес ОС «общего назначения»);

Архитектура продукта

Даже самый прекрасный процесс устранения недостатков забуксует (или будет слишком дорогим), если продукт спроектирован некорректно. Потому мы:

- При проектировании следуем принципам K.I.S.S., «UNIX Way», теории надежности;
- Обязательно учитываем модель угроз и сценарии использования (негативные тоже);
- Обеспечиваем возможность расширения функций и независимого обновления, например, добавлением новых модулей, сервисов или даже плагинов;
- Каждый сервис, модуль или плагин - реализуется как «вещь в себе», минимально зависящий от среды исполнения (насколько это возможно).
- Среда исполнения содержит только то, что нужно для работы функционального слоя (подход встраиваемых систем, в противовес ОС «общего назначения»);
- **Все документируем**, целиком и полностью, включая этапы принятия архитектурных решений (ведем Architecture Decision Records Log).

Работа с источниками

Работа с источниками для нас разделилась (еще до ГОСТ РБПО) на:

- Результаты регулярных запусков инструментов SDL (SAST/DAST/SCA) для продуктов «в поле»;
- Результаты регулярных запусков инструментов SDL (SAST/DAST/SCA) для разрабатываемых релизов;
- Данные о предполагаемых уязвимостях от клиентов или из рассылок.

Работа с источниками

Работа с источниками для нас разделилась (еще до ГОСТ РБПО) на:

- Результаты регулярных запусков инструментов SDL (SAST/DAST/SCA) для продуктов «в поле»;
- Результаты регулярных запусков инструментов SDL (SAST/DAST/SCA) для разрабатываемых релизов;
- Данные о предполагаемых уязвимостях от клиентов или из рассылок.

Поток огромный, но мы предусмотрели следующее (по каждому продукту):

- Настраиваемый набор тестов и правил для фильтрации нерелевантных уязвимостей;
- Тесты и правила *автоматизированного* вычисления CVSS 3.1;

Работа инженеров с уязвимостями

Для эффективной работы с уязвимостями, предусмотрены:

- Информационный портал, где инженеры техподдержки могли бы посмотреть, что есть по уязвимости X для продукта Y и дать соответствующий ответ;

Работа инженеров с уязвимостями

Для эффективной работы с уязвимостями, предусмотрены:

- Информационный портал, где инженеры техподдержки могли бы посмотреть, что есть по уязвимости X для продукта Y и дать соответствующий ответ;
- Единая команда экспертов (SecTeam) - как со знанием общих заимствованных компонентов среды исполнения, так и специфичных для продуктов;

Работа инженеров с уязвимостями

Для эффективной работы с уязвимостями, предусмотрены:

- Информационный портал, где инженеры техподдержки могли бы посмотреть, что есть по уязвимости X для продукта Y и дать соответствующий ответ;
- Единая команда экспертов (SecTeam) - как со знанием общих заимствованных компонентов среды исполнения, так и специфичных для продуктов;
- Оценка релевантности уязвимости и ее уровеня критичности экспертами, а также фиксация знаний в виде документированных правил и тестов;

Работа инженеров с уязвимостями

Для эффективной работы с уязвимостями, предусмотрены:

- Информационный портал, где инженеры техподдержки могли бы посмотреть, что есть по уязвимости X для продукта Y и дать соответствующий ответ;
- Единая команда экспертов (SecTeam) - как со знанием общих заимствованных компонентов среды исполнения, так и специфичных для продуктов;
- Оценка релевантности уязвимости и ее уровеня критичности экспертами, а также фиксация знаний в виде документированных правил и тестов;
- Централизованный инструмент тестирования CVE и анализа, связанный с порталом и ServiceDesk, внутренней системой работы с ошибками (bug-tracking) и системой управления проектом.

Система сборки, тестирование и релизный цикл

Наконец, нужно выпустить обновление. Как его сделать безболезненным для тестировщиков и для клиентов? Для этого:

- Реализована воспроизводимая сборка;
- Однозначно идентифицируются затронутые исправлением модули продукта;
- Однозначно идентифицируется затронутый исправлением функционал продукта;
- При выпуске обновлений безопасности (для уязвимостей критического и высокого уровня) тестовая система проверяет только то, что прямо или косвенно затронуто изменениями;
- При выпуске регулярных патч-релизов (включают все хот-фиксы и исправления дефектов) тестируется все как обычно.

В итоге:

- Огромное количество входящей информации и часто неадекватная оценка значимости дефекта требуют автоматизированной обработки экспертами;
- Правильно спроектированный продукт, минимальная зависимость от внешних факторов - существенно снижают ресурсоемкость, увеличивают качество и скорость технической поддержки;
- Большой поток исправлений в коде требует изменения подходов выпуска релизов и тестирования;
- Установка обновлений должна производиться клиентами самостоятельно, где только это возможно (непривычно для Enterprise);
- Автоматизация, документирование, и еще раз автоматизация!

yadro.com

123376, Москва г., Рочдельская ул., дом 15, строение 15

a.dubinin@yadro.com