Автоматизация процессов анализа безопасности операционной системы Astra Linux.

Егорова Виктория Вячеславовна, vegorova@astralinux.ru. ПАО Группа Астра. 117105, г. Москва, Варшавское ш., д.26, стр.11

При разработке операционной системы подходы по анализу безопасности программного кода, такие как статический и динамический анализ, выделяются как ключевые методы обнаружения ошибок и уязвимостей на ранних этапах цикла непрерывной разработки. При этом автоматизация и совместное применение данных подходов, когда речь идет о разработке и тестировании такого масштабного продукта, как операционная система, также является важной задачей, играющей решающую роль в обеспечении безопасности, способствуя своевременному обнаружению и устранению ошибок и уязвимостей.

Для обеспечения доверия к операционной системе Astra Linux в качестве основных инструментов анализа применяются наиболее передовые подходы и программные средства, позволяющие в первую очередь эффективно анализировать компоненты, образующие поверхность атаки, важнейшей частью которой являются собственные средства защиты информации (далее – СЗИ), функционирующие как в пользовательском пространстве, так и на уровне ядра операционной системы, защищенные средства контейнеризации и виртуализации, СУБД и другие, реализующие функции безопасности операционной системы и ее компонентов. Так, помимо анализа защищенности собственных СЗИ, в компании проводится анализ программного обеспечения с открытым исходным кодом, которое играет немаловажную роль, в том числе в реализации функций защиты информации.

В рамках работ по развитию и автоматизации процессов фаззинга ядра ОС, ведется разработка собственного фреймворка, в том числе включающего в себя веб-приложение, которое заменит текущую панель мониторинга, интегрированную в syzkaller. Данный инструмент позволяет пользователям получить доступ к информации об активных настройках фаззинг-тестирования, обнаруженных ошибках и данных для их воспроизведения. Помимо этого, с целью проведения фаззинг-тестирования в рамках цикла непрерывной разработки, в платформу включена функциональность интеграции кода собственных модулей безопасности ядра с ядром операционной системы и сохранение результатов в указанный репозиторий. Такой подход позволяет избежать необходимости модификации исходного кода syzkaller, что исключает конфликты между исходным репозиторием и локальной версией, тем самым упрощая обновление стендов фаззинга. Помимо этого, выделение функционала в отдельный сервис облегчает автоматизацию запуска процессов фаззинга в рамках CI и позволяет интегрировать модули безопасности в ядро и пересобирать образы для всех экземпляров syz-manager с помощью простых действий в рамках веб-интерфейса. Кроме того, это не ограничивает добавление дополнительного необходимого функционала.

Все процессы фаззинга компонентов пользовательского пространства также интегрированы в разрабатываемую внутри Группы Компаний платформу автоматизации, встраиваемую в CI, учитывающую специфику тестируемых компонентов и предоставляющую результаты тестирования с помощью различных способов – посредством графического интерфейса и в рамках CI-пайплайнов. Помимо этого, основная архитектура была спроектирована таким образом, чтобы предоставлять возможность автоматического масштабирования вычислительных мощностей в соответствии с текущими потребностями, которые могут возникать в процессе выполнения различных задач фаззинга. Более того, платформа нацелена на максимально эффективное использование доступных вычислительных ресурсов, что достигается за счет умного динамического распределения задач между всеми доступными вычислительными узлами. Разработанный инструмент взаимодействует с внутренней трекинг-системой, а вся информация об обнаруженных ошибках экспортируется в формате SARIF для дальнейшей обработки.

В качестве основных инструментов статического анализа используются ряд инструментов, также положительно зарекомендовавших себя в сообществе, среди них SVACE, Clang SA, Cppcheck, appScreener, AKBC-3 и CodeQL. При этом каждый из инструментов обладает собственной спецификой и выбирается в соответствии с особенностями конкретного анализируемого компонента. Статический анализ проводится как в рамках релизного цикла, так и в CI-пайплайнах в рамках цикла непрерывной разработки, что позволяет своевременно реагировать на обнаруженные недостатки и устранять обнаруженные ошибки на ранних этапах.

Все результаты анализа хранятся и обрабатываются в рамках собственного инструментального комплекса, представляющего собой единую платформу сбора, хранения и обработки результатов работы средств анализа исходного кода компонентов операционной системы. Инструментальный комплекс позволяет производить полный комплекс работ по обработке результатов средств анализа, в том числе реализует приоритизацию обнаруженных ошибок с помощью алгоритмов машинного обучения, а механизмы синхронизации с платформами автоматизации процессов фаззинга, позволяют коррелировать и пересчитывать полученные ранее результаты на обновленных данных. Таким образом, результаты анализа приоритизируются не только на основе информации от анализаторов, но и с использованием всех имеющихся на платформе данных.

Совокупность используемых подходов позволяет минимизировать участие специалистов в настройке средств анализа и запуске стендов тестирования, а также сократить время, затрачиваемое на первичный анализ полученных результатов и дедупликацию срабатываний, полученных в результате тестирования. Более того, предложенный подход автоматизации процессов в конечном итоге исключает для специалиста необходимость запускать стенды фаззинга самостоятельно при внесении изменений разработчиком в систему контроля версий. Разработанные платформы для автоматизации процессов фаззинга ядра ОС и компонентов пространства пользователя позволяют существенно сократить время обнаружения ошибок в новых коммитах, что является важным критерием для оценки результативности.

Таким образом, применяемые подходы позволяют не только минимизировать количество ошибок в коде системного и прикладного ПО на протяжении всего жизненного цикла разработки, выявляя возможные уязвимости на ранних этапах, но и предоставляет исчерпывающую статистику по результатам, обнаруженным за все время анализа. Такая статистика позволяет анализировать состояние компонентов в контексте безопасности и принимать решения о переходе на новые версии, сравнивая обнаруженные ошибки. Помимо этого, возможность коррелировать средства анализа между собой также ускоряет работу специалистов, занимающихся анализом полученных результатов. Платформы интегрированы со внутренним трекером задач, что позволяет в автоматическом режиме передавать информацию о результатах анализа разработчиком и также ускоряет процессы анализа безопасности.