Применение возможностей KasperskyOS и принципов SOLID при проектировании информационных систем

Сорокин Игорь Александрович, Igor.Sorokin@kaspersky.com
АО «Лаборатория Касперского», Россия, Москва, 125212, Ленинградское шоссе, д.39А, стр.3, БЦ «Олимпия Парк»

1. Мотивация создания

В современных ОС изоляция процессов является ключевым механизмом обеспечения безопасности, стабильности и эффективности. Ключевую роль в управлении изоляцией процессов и повышения уровня защиты информационных систем (ИС) играют политики безопасности, применяемые на уровне ядра (SELinux/AppArmor, KasperskyOS). При этом следует отметить, что принципы использования политик безопасности в KasperskyOS не используют классические «права доступа» (как в SELinux), а требуют явной декларации взаимодействия между процессами с указанием конкретных интерфейсов (бизнес-ориентированные IPC-каналы).

Однако, кроме требований безопасности к информационной системе предъявляются и другие требования: прежде всего функциональные, а также масштабируемости и расширяемости. При проектировании такой системы архитектору необходимо балансировать между безопасностью и другими требованиями, что требует методичного подхода к декомпозиции системы на сервисы\процессы.

Рассмотрим, какие шаги необходимо выполнить для декомпозиции и как принципы SOLID и возможности KasperskyOS могут помочь на данном шаге проектирования ИС.

2. Шаги проектирования

При проектировании ИС необходимо:

- Анализ функциональных требований и критичности компонентов.
- Составить матрицу зависимостей всех функций системы.
- Разделить систему на компоненты\сервисы и определить какие интерфейсы должны быть предоставлены каждым сервисом.
- Определить SLA для каждого компонента (например, доступность, производительность и т.п.).
- Выделить критические домены и определить уровни безопасности.
- Сопоставить требования безопасности и SLA.

3. Разделение системы на компоненты и использование принципов SOLID

Данный этап представляет собой наиболее сложную задачу. Принципы SOLID, разработанные Робертом С. Мартином, могут быть адаптированы для создания ИС с оптимальным балансом безопасности, производительности и расширяемости. Рассмотрим данные принципы в случае проектирования ИС.

Single Responsibility Principle (SRP) - каждый сервиса должен отвечать только за одну бизнес-функцию, что:

- уменьшает поверхность атаки (меньше кода, а значит меньше уязвимостей);
- позволяет применять точечные политики безопасности.

Open/Closed Principle (OCP) - системы открыта для расширения, но закрыта для модификаций. Это означает, что новые бизнес-функции реализуются в системе через добавление новых сервисов (или плагинов существующих сервисов). Ядро ИС остается неизменным.

Liskov Substitution Principle (LSP) - подтипы должны замещать свои базовые типы без нарушения корректности. Это означает, что API должен оставаться стабильным, а значит обеспечивает безопасное (с точки зрения функционирования системы) обновление сервисов и не требует изменения политик безопасности.

Interface Segregation Principle (ISP) - Минимизация привилегий, то есть сервисы предоставляют клиентам минимально необходимый API. Это значительно облегчает разработку политик безопасности, сокращает поверхность атаки.

Dependency Inversion Principle (DIP) - Зависимость должна быть от абстракций, а не от конкретных реализаций, то есть должны быть использованы единые контракты. Использование DIP позволяет обновлять механизмы безопасности в ИС без переписывания кода. Например, выбор криптографического алгоритма осуществляется в процессе работы ИС (на runtime). При этом интерфейс для шифрования данных не меняется.

Следует отметить, что использование данных принципов базируется на применение интерфейсов, что наиболее естественно для KasperskyOS.

4. Балансировка требований через SOLID

Как было отмечено выше, разделение на домены безопасности является итерационным процессом, при котором SOLID может помочь пересмотреть архитектуру ИС в соответствии с требованиями.

Требование	**SOLID-принцип**	**Реализация**
Безопасность	SRP + ISP	Изолированные процессы с минимальными правами
Производительность	LSP + DSP	Возможность выбирать более производительные алгоритмы без изменения кода ядра ИС
Расширяемость	OCP + DIP	Возможность использования дополнительных сервисов или плагинов для наращивания функциональности
Масштабируемость	LSP + ISP	Возможность использования stateless-сервисов. Это позволяет запускать несколько сервисов в зависимости от нагрузки на ИС.

5. Система мониторинга и контроль работоспособности

Одной из важных частей ИС с точки зрения безопасности является подсистема мониторинга. Данная подсистема обеспечивает следующие функции.

Функция	**Описание**
Мониторинг поведения процессов	Сбор данных о системных вызовах, обращениях к файлам и сети
Обнаружение аномалий	Сравнение текущего поведения с эталонным профилем
Автоматическое реагирование	Остановка, перезапуск или блокировка подозрительного компонента
Логирование событий	Хранение событий для последующего анализа и аудита

В KasperksyOS реализована подсистема System Health Management, которая реализует данные функции. Особенно следует отметить алгоритмы обнаружения аномалий. Данные алгоритмы основаны на использовании Control Flow Graph, формируемых из исходного кода процессов. Такой подход дает точное формирование эталонного профиля с

минимальными усилиями со стороны программистов и позволяет обнаруживать аномалии наиболее эффективно.

6. Заключение

Объединение принципов SOLID и возможностей KasperskyOS позволяет строить системы, в которых:

- каждый компонент имеет свою чётко определённую роль,
- процессы изолированы и защищены,
- политики безопасности задаются на уровне компиляции,
- возможен мониторинг поведения и автоматическое реагирование на угрозы.

Это делает такую архитектуру идеальной для применения в:

- промышленной автоматизации,
- системах безопасности,
- устройствах IoT/IoE,
- государственных и военных проектах.