УДК 004.056:004.434

АКТИВНАЯ ЗАЩИТА ИСПОЛНЯЕМОГО КОДА С ИСПОЛЬЗОВАНИЕМ АППАРАТНЫХ КРИПТОСРЕДСТВ И ПОЛИМОРФНЫХ КЛАССОВ C++

Н.В. Хаджинова

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники», г. Минск, Республика Беларусь

Аннотация. Рассматривается подход к активной защите исполняемого кода, интегрирующий механизмы объектно-ориентированного программирования с аппаратными криптосредствами (*TPM, HSM*). Предлагается архитектура, где критический компонент представлен полиморфным классом, инкапсулирующим функциональную логику и контекст безопасности. Математически описана процедура верификации целостности в реальном времени. Аппаратные средства обеспечивают корень доверия. Накладные расходы составляют 8–12 %, что приемлемо для распределенных систем управления.

Ключевые слова: активная защита кода; полиморфные классы; аппаратные криптосредства; *TPM*; контроль целостности.

ACTIVE PROTECTION OF EXECUTABLE CODE USING HARDWARE CRYPTOGRAPHIC MEANS AND POLYMORPHIC C++ CLASSES

N.V. Khajynava

Educational Institution “Belarusian State University of Informatics and Radioelectronics”, Minsk, Republic of Belarus

Abstract. An approach to active protection of executable code integrating object-oriented programming with hardware cryptographic means (*TPM, HSM*) is considered. An architecture is proposed where each critical component is represented by a polymorphic class encapsulating functional logic and security context. The real-time integrity verification procedure is mathematically described. Hardware tools provide a root of trust. Overhead costs are 8–12 %, which is acceptable for distributed control systems.
Keywords: active code protection; polymorphic classes; hardware cryptographic means; *TPM*; integrity control.

Введение

Программные системы функционируют в условиях постоянных атак на исполняемый код: модификация, внедрение вредоносных компонентов, нарушение потока выполнения. Особую опасность представляют атаки времени выполнения (эксплуатация уязвимостей памяти, подмена управления). Традиционные методы защиты (статический анализ, стандартный контроль целостности) не всегда оперативно реагируют на динамические угрозы в средах с высокими требованиями к детерминированности.

Основная часть

Предлагаемый подход основан на представлении каждого критического компонента полиморфным классом, наследующим от абстрактного базового класса *SecureComponent* (рисунок 1). Это позволяет оптимизировать распределение ресурсов и инкапсулировать механизмы защиты. При создании экземпляра (в конструкторе) инициируется аттестация:

– вычисление текущего хеша сегмента кода класса;

– сравнение с эталонным «слепком», хранящимся в защищенной памяти (на сервере аттестации или в хранилище, ассоциированном с *TPM*);

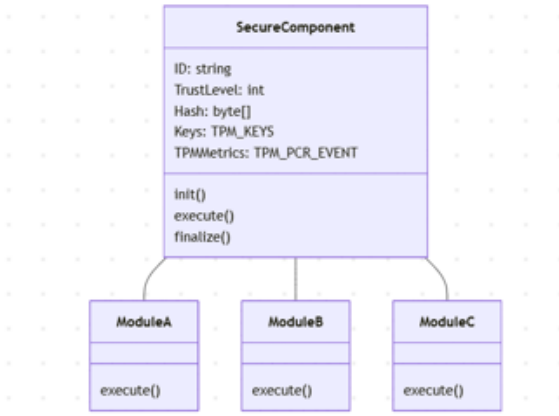
– проверка регистров конфигурации платформы (*PCR*) для подтверждения доверенной среды загрузки.

Аппаратные криптосредства (*HSM/TPM 2.0*) хранят долгоживущие ключи дешифрования [1]. Код может находиться в памяти в зашифрованном виде и расшифровываться *just-in-time* перед вызовом виртуальной функции через *vtable*, которая также мониторится.

На рисунке показана иерархия: базовый класс *SecureComponent* содержит чисто виртуальные методы и защищенные члены для контекста безопасности. Производные классы переопределяют методы и наследуют механизмы верификации, вызываемые в конструкторе и деструкторе.

Создание объекта разрешается только при условии $E(C_i) = 1$. В случае нарушения целостности или несоответствия платформенного контекста

срабатывает механизм экстренного завершения процесса с очисткой данных в деструкторах.



Наследование от абстрактного класса
Inheritance from an abstract class

Процесс верификации компонента C_i при его активации в системе можно представить в виде функции доверия $E(C_i)$:

$$E(C_i) = \Phi(H(P_i) \oplus PCR_j, \sigma_{HSM}), \quad (1)$$

где $H(P_i)$ – хеш-сумма сегмента памяти, занимаемого кодом класса; PCR_j – значение регистра конфигурации платформы *TPM*; σ_{HSM} – эталонная цифровая подпись, вычисленная для корректного состояния компонента и защищенная аппаратным модулем; Ф – функция проверки, выполняемая в защищенном контуре аппаратного модуля (например, *TPM* или *HSM*).

Интеграция с аппаратными средствами позволяет минимизировать временные затраты на криптографические операции. Суммарное время выполнения защищенного вызова T_{total} определяется как:

$$T_{total} = T_{exec} + T_{verify} + T_{decrypt}, \quad (2)$$

где T_{verify} – время, затрачиваемое на вычисление хеша процессором и его верификацию с эталонным значением, защищенным *TPM* [2]. $T_{вхвс}$ – время выполнения самой защищаемой функции. Для сегментов кода размером до 1 МБ операция верификации на процессорах среднего класса (например, *Intel Core i5-10600KF* на частоте 3,5 ГГц) не превышает 1,5 мс.

Для оценки применимости подхода проведен эксперимент на прототипе системы управления сетевыми ресурсами. Конфигурация: *Intel Core i5-10600KF*, 16 ГБ RAM, *TPM* 2.0, *Linux (kernel 5.x)* с поддержкой измеряемой загрузки. Тестовое приложение включало 10 компонентов – полиморфных классов C++, наследующих от SecureComponent.

Оценивалось влияние защиты на время отклика (*latency*). Базовый уровень – проверка целостности кода при инициализации (SHA-256,

эталонные значения в *TPM*). Средний уровень – контроль целостности потока выполнения при вызове виртуальных методов. Полный уровень – динамическое шифрование сегментов кода с ключами, привязанными к состоянию платформы.

Тесты (не менее 30 замеров на режим) показали: накладные расходы *CPU* составляют 3–5 % для базовой верификации и до 12 % при полном шифровании.

Заключение

Предложенный подход обеспечивает интеграцию механизмов безопасности в архитектуру ПО за счет сочетания полиморфных абстракций C++ и аппаратных криптосредств. Экспериментально подтверждены практическая применимость и приемлемость накладных расходов (8–12 % в полном режиме). Дальнейшее развитие связано с расширением защиты для распределенных систем и автоматизацией контроля доверенного состояния.

Список использованных источников

1. Субашини С., Кавита В. (2011) Обзор проблем безопасности в моделях предоставления облачных вычислений. Journal of Network and Computer Applications, 34 (1), 1–11.

2. Артур У., Челленер Д., Гольдман К. (2014) Практическое руководство по TPM 2.0: Использование TPM в новой эре безопасности. Syngress.

References

1. Subashini S., Kavitha V. (2011) A Survey on Security Issues in Service Delivery Models of Cloud Computing. Journal of Network and Computer Applications, 34 (1), 1–11 (in Russian).

2. Arthur W., Challener D., Goldman K. (2014) A Practical Guide to TPM 2.0: Using the TPM in the New Age of Security. Syngress.

Сведения об авторе

Хаджинова Наталья Владимировна, старший преподаватель кафедры ИТАС, учреждение образования «Белорусский государственный университет информатики и радиоэлектроники», khajynova@bsuir.by.

Information about the author

Khajynava, **N.,** Senior Lecturer at the Department of Information Technologies of Automated Systems, Educational Institution "Belarusian State University of Informatics and Radioelectronics", khajynova@bsuir.by.