Реферат на тему ‘Нарушения безопасности доступа к памяти: переполнения буфера, висячие указатели’

Основы информационной безопасности

Андрианова Марина Георгиевна

Содержание

1 Цель работы

Рассмотреть причины возникновения, последствия переполнений буфера и висячих указателей, а также методы защиты от них.

2 Введение

Одним из наиболее распространённых классов уязвимостей программного обеспечения являются проблемы безопасности памяти. Данный тип уязвимости известен с 1980-х годов. Безопасность памяти подразумевает предотвращение попыток использовать или модифицировать данные, если это не было намерено разрешено программистом при создании программного продукта.

3 Переполнение буфера

Переполнение буфера — запись за пределами выделенного в памяти буфера. Возникает при попытке записи в буфер блока данных, превышающего размер этого буфера. В результате переполнения могут быть испорчены данные, расположенные рядом с буфером, либо программа вовсе изменит своё поведение, вплоть до интерпретации записанных данных как исполняемого кода.

3.1 Причины возникновения

Переполнение буфера происходит, когда в программе не реализована должная проверка границ буфера. Рассмотрим основные моменты, способствующие возникновению этой уязвимости:

1. Недостаточная проверка входных данных:

- Программы часто принимают данные от пользователей или других систем. Если не проверять размер входных данных перед их записью в буфер, это может привести к переполнению. Например, функция, которая принимает строку, может не проверять, превышает ли длина этой строки размер выделенного буфера.

2. Использование небезопасных функций:

- В языках программирования, таких как C и C++, существуют функции, которые не проверяют размер буфера. Например, функции strcpy(), gets(), scanf() и другие могут записывать данные в буфер без проверки его размера. Это делает программы уязвимыми к переполнению.

3. Стековая и кучи:

- Переполнение буфера может происходить как на стеке, так и в куче. В случае стекового переполнения злоумышленник может перезаписать адрес возврата, что позволяет ему выполнить произвольный код. В куче переполнение может привести к нарушению управления памятью, перезаписи метаданных, используемых для управления выделением памяти.

4. Отсутствие защиты компилятора:

- Некоторые компиляторы предлагают встроенные средства защиты от переполнений буфера, такие как Stack Smashing Protector (SSP) или использование защищенных функций для работы со строками. Однако, если такие механизмы не включены, программы остаются уязвимыми.

Таким образом, переполнение буфера возникает из-за недостатков в проектировании и реализации программного обеспечения, что делает его одной из наиболее распространенных уязвимостей в области кибербезопасности.

3.2 Последствия переполнения буфера

Переполнение буфера может иметь серьезные последствия для безопасности и стабильности программного обеспечения, включая:

1. Исполнение произвольного кода.
2. Утечка конфиденциальной информации.
3. Сбои и нестабильность системы.
4. Повреждение данных.
5. Увеличение уязвимости системы.

4 Висячие указатели

Висячий указатель (или «висячая ссылка») — это указатель, который ссылается на область памяти, которая была освобождена или не существует. Возникает, когда объект был удалён (или перемещён), но значение указателя не изменили на нулевое. В данном случае он всё ещё указывает на область памяти, где находился данный объект. В некоторых случаях это может стать причиной получения конфиденциальной информации злоумышленником; либо, если система уже перераспределила адресуемую память под другой объект, доступ по висячему указателю может повредить расположенные там данные.

4.1 Причины возникновения:

Висячие указатели возникают по нескольким причинам:

- Освобождение памяти: Когда память выделяется динамически (например, с помощью malloc() или new), и после использования она освобождается, но указатель, ссылающийся на эту память, остается без изменений.
- Изменение объема памяти: При перераспределении памяти (например, с использованием realloc() в C) старый указатель может стать висячим, если не обновить его для указания на новый адрес выделенной памяти.
- Ошибки в логике программы: Ошибки в коде могут привести к тому, что указатели не обновляются или не обнуляются после освобождения памяти.

4.2 Последствия висячих указателей

Последствия использования висячих указателей могут быть серьезными и включать:

1. Сбой программы.
2. Непредсказуемое поведение.
3. Уязвимости безопасности.
4. Повреждение данных.

5 Методы защиты и предотвращения

5.1 Защита от переполнения буфера

Существует несколько методов защиты от переполнения буфера, которые могут быть реализованы на уровне программирования, компиляции и операционных систем:

1. Использование безопасных функций:

- Избегайте использования небезопасных функций, таких как strcpy(), strcat(), sprintf(), и используйте безопасные альтернативы, такие как strncpy(), strncat(), snprintf(), которые позволяют контролировать количество копируемых или добавляемых байтов.

2. Проверка границ:

- Всегда проверяйте границы массивов и буферов перед записью данных. Это можно сделать с помощью явной проверки длины входных данных.

3. Использование автоматического управления памятью:

- Используйте языки программирования с автоматическим управлением памятью (например, Java, Python), которые минимизируют риск переполнения буфера за счет неявного управления памятью.

4. Компиляция с защитными флагами:

- Используйте компиляторы с флагами безопасности, такими как -fstack-protector в GCC, которые добавляют защитные механизмы, такие как защитные маркеры вокруг функций, чтобы обнаруживать переполнение стека.

5. Использование адресного пространства:

- Включение механизма защиты, такого как ASLR (Address Space Layout Randomization), который усложняет предсказание адресов памяти и делает атаки, основанные на переполнении буфера, менее эффективными.

6. Использование стековых защитников:

- Применение технологий, таких как DEP (Data Execution Prevention), которые предотвращают выполнение кода в определенных областях памяти.

5.2 Защита от висячих указателей

Для предотвращения проблем, связанных с висячими указателями, можно использовать следующие методы:

1. Обнуление указателей:

- После освобождения памяти всегда обнуляйте указатели, чтобы избежать доступа к освобожденной памяти. Например, после вызова free(ptr), следует установить ptr = NULL.

2. Управление памятью с помощью умных указателей: -

В C++ используйте умные указатели (например, std::unique_ptr, std::shared_ptr), которые автоматически управляют временем жизни объектов и освобождают память, когда они больше не нужны.

3. Использование систем управления памятью:

- Используйте библиотеки или фреймворки, которые предоставляют безопасные методы управления памятью и защищают от ошибок, связанных с висячими указателями.

4. Ведение учета выделенной памяти:

- Используйте методы отслеживания и ведения учета выделенной памяти, чтобы выявлять утечки памяти и висячие указатели во время выполнения программы.

5. Проверка состояния указателей:

- Вводите проверки состояния указателей перед их использованием, чтобы убедиться, что они указывают на действительные объекты или память.

6 Заключение

- Понимание механизмов нарушений безопасности доступа к памяти, таких как переполнение буфера и висячие указатели, является критически важным для разработки надежного программного обеспечения. Эти уязвимости могут привести к серьезным последствиям, включая потерю данных, несанкционированный доступ к системам и выполнение произвольного кода.

- Обеспечение кибербезопасности требует комплексного подхода, который включает не только технические меры, но и обучение пользователей, политику безопасности и регулярный аудит систем. Технологии и методы, направленные на предотвращение уязвимостей, должны сочетаться с регулярными обновлениями программного обеспечения, мониторингом и анализом угроз.

7 Выводы

Рассмотрели причины возникновения, последствия переполнений буфера и висячих указателей, а также методы защиты от них.

Список литературы

• Джеймс Фостер, Майк Прайс. Защита от взлома: сокеты, эксплойты, shell-код = Sockets, Shellcode, Porting, & Coding. — М.: Издательский Дом ДМК-пресс, 2006. — С. 35, 532. — 784 с. — ISBN 5-9706-0019-9.

• Джон Эриксон. 0x320 Переполнение буфера // Хакинг: искусство эксплойта = Hacking: The Art of Exploitation. — 2-е издание. — СПб.: Символ-Плюс, 2010. — С. 139. — 512 с. — ISBN 978-5-93286-158-5.

• https://ru.ruwiki.ru/wiki/Безопасность_доступа_к_памяти#Типы_ошибок_памяти