УДК 004.052.42

ФОРМАЛЬНАЯ ВЕРИФИКАЦИЯ АЛЛОКАТОРОВ ПАМЯТИ С ПРИМЕНЕНИЕМ СИМВОЛЬНОГО ВЫПОЛНЕНИЯ

Д. С. Садырин
Университет ИТМО, Санкт-Петербург

Научный руководитель – А.М. Дергачев
Университет ИТМО, Санкт-Петербург

Введение. В настоящее время программное обеспечение (ПО) используется в разных областях человеческой деятельности. Ошибки в ПО могут привести к потерям и ущербу, поэтому задача верификации ПО с целью обнаружения ошибок является актуальной. Одним из способов решения задачи верификации ПО является метод ограниченной проверки моделей (Bounded Model Checking). В языке С выделением динамической памяти занимается аллокатор (от англ. allocator). Ошибки в программном коде аллокатора являются причиной уязвимостей в прикладном коде программ.

Целью работы является разработка механизмов верификации аллокаторов динамической памяти с использованием метода ограниченной проверки моделей.

Базовые положения исследования. Для поиска ошибок в методе проверки моделей используется рассмотрение полного пространства состояний заданной программы с целью поиска состояний, нарушающих те или иные свойства корректности. Ограниченная проверка моделей позволяет сократить используемые ресурсы за счет обхода модели на определенную глубину, называемую границей (bound). Ограниченное пространство состояний представляется в виде формул пропозициональной логики, которые затем комбинируются с условиями корректности программы и проверяются на непротиворечивость при помощи SMT (Satisfiability Modulo Theories) решателя.

Для обхода состояний модели был применен алгоритм символьного выполнение (symbolic execution). Техника символьного выполнения позволяет проводить моделирование выполнения программы, при котором часть входных переменных представляется в символьном виде. Символьная переменная обозначает множество значений входной переменной программы из области ее определения. Каждое символьное выполнение эквивалентно выполнению программы на наборе конкретных тестовых значений входных переменных, что сокращает мощность множества создаваемых тестов. От символьного выполнения требуется подобрать входные данные, на которых произойдет ошибка.

Промежуточные результаты. Была исследована работа аллокатора ptmalloc библиотеки glibc, рассмотрены техники эксплуатации переполнения кучи, взятые из журнала Phrack. Для каждой техники были составлены тестовые примеры, которые затем описаны с помощью транзакций: M – malloc, F – free, O – overflow, UAF – use-after-free, DF – double free, FF – fake free. Транзакция определяется как операция, которая изменяет состояние кучи программы. Каждая транзакция представлена как заглушка кода, выполняемая желаемые действия. Комбинация этих заглушек кода затем создает исходный код тестовых примеров.

В тестовых примерах входные параметры для функции malloc были заменены символьным представлением, также, как и данные из стандартного потока ввода (stdin).

Далее были заданы свойства модели, описывающие отрицательные характеристики аллокатора: NHA (Non Heap Allocation) – возвращается адрес за пределами кучи; OA (Overlapped Allocation) – перекрытие выделенных чанков памяти; AW (Arbitrary Write) – запись по произвольному адресу памяти.

Каждое свойство задается в виде пропозициональной формулы. Производится символьное выполнение всех тестовых примеров и дальнейшая проверка выполнимости

свойств модели, которая будет являться условием наличия ошибки в коде аллокатора. Далее все символьные переменные в тестовых примерах будут заменены посчитанными конкретными значениями.

В работе рассмотрен подход к верификации аллокаторов с применением метода ограниченной проверки моделей. В дальнейшем данный подход будет применен для верификации аллокаторов памяти, таких как ptmalloc, dlmalloc, tcmalloc, jemalloc, musl различных версий. По набранной статистике предполагается сделать вывод о наличии уязвимостей, связанных с различными типами ошибок: NHA, OA, AW.