Обнаружение уязвимостей класса «удаленное исполнение кода» на основе абстрактной интерпретации

Холин Роман Вадимович

Аспирант, Младший научный сотрудник

Факультет ВМК МГУ имени М. В. Ломоносова, Москва, Россия

E-mail: romankholin94@seclab.cs.msu.ru

Научный руководитель *— Гамаюнов Денис Юрьевич*

Атаки «удаленное исполнение кода» были и остаются самыми опасным видом атак, они могут привести к разглашению конфиденциальных данных, несанкционированному удалению данных, перехват управления устройства и другим уязвимостям, поэтому важно уметь автоматически находить уязвимые места в программах.

Удаленное исполнение кода возможно из-за того, что в программах есть функции, которые выполняют поступающим им на вход код на некотором языке программирования (например, функция «eval» в языке программирования javascript, функция «echo» в языке программирования php или функция выполнения запроса к базе «mysql_query» в языке программирования php), причем этот код может формироваться динамически и в него может попасть пользовательский ввод, т.е. может произойти «инъекция» в код через пользовательский ввод. «Инъекция» в код, который поступает на вход функциям «запрос к базе» (например, в ход «mysql_query» в php) называют SQL-инъекцией, в код, который поступает на вход функции «eval» («исполнить» код; такие функции есть, например, в языках js и php) — инъекцией кода. Функции, описанные выше, будем называть «чувствительными функциями», а аргумент, который они принимают — «запрос к чувствительной функции».

Предлагается обобщить задачу обнаружения уязвимостей внедрения кода для широкого класса инъекций и искать их с помощью одного их методов абстрактной интерпретации — символического исполнения. Для этого будем искать в программах искать чувствительные функции и какие входные данные подать на вход программе (в случае веб–приложения — какой сделать запрос веб–приложению), чтобы программа, запущенная на этих входных данных, так же дошла до данной чувствительной функции выполнила её.

В работе сделано обобщение модели уязвимостей рассматриваемых классов и прототипная реализация для SQLi и NodeJS. Для реализации прототипного решения задачи модифицировано средство

генерации тестовых примеров для языка JavaScript «ExpoSE» [2], использующую средство инструментации кода для языка JavaScript «Jalangi2» [4]. Оно будет основано на поиске чувствительных функций с помощью конкретно–символического обхода программы (подробно об этом методе написано, например, в этом обзоре [1]). С помощью этого метода можно обойти все пути исполнение программы, если количество таких путей конечно; если количество таких путей бесконечно, то значительную их часть. Проходя через операторы ветвления, собираются ограничения на входные данные с помощью символьного исполнения. Используя эти ограничения, и SMT–решатели (например, «Z3» [3]) можно генерировать входные данные, которые бы удовлетворяли новым, ещё не пройденным путям обхода программы. Дойдя до чувствительной функции, можно сгенерировать запрос, который приводит нас к данной точке программы.

На данном этапе разработки наше прототипное решение ищет уязвимости с помощью средства «sqlmap» [5] уязвимости, передав получившийся запрос данному средству (из этого запроса «sqlmap» поймет, какие параметры запроса исследовать в первую очередь, что повысит вероятность нахождения уязвимостей и ускорит поиск уязвимостей, если они есть).

В заключение автор выражает признательность, доценту кафедры ИБ ВМК МГУ заведующему лабораторией МПКБ к.ф.–м.н. Д. Ю. Гамаюнову за научное руководство.

Литература

1. Baldoni R. et al. A survey of symbolic execution techniques //ACM Computing Surveys (CSUR). – 2018. – Т. 51. – №. 3. – С. 1–39.
2. Loring B., Mitchell D., Kinder J. ExpoSE: practical symbolic execution of standalone JavaScript //Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software. – 2017. – С. 196–199.
3. Moura L., Bjørner N. Z3: An efficient SMT solver //International conference on Tools and Algorithms for the Construction and Analysis of Systems. – Springer, Berlin, Heidelberg, 2008. – С. 337–340.
4. https://github.com/Samsung/jalangi2
5. https://sqlmap.org/