Лабораторная работа № 2 – Изучение принципов поиска уязвимостей в программном обеспечении без исходных кодов

Цель работы

Изучение типовых ошибок и принципов поиска уязвимостей в программном обеспечении без исходных кодов

Формулировка задания

Одним из наиболее распространенных способов поиска ошибок и уязвимостей в программном обеспечении в условиях отсутствия исходного кода является фаззинг. В данной лабораторной работе предлагается осуществить мутационный фаззинг формата файла. Так как формат конфигурационного файла не известен, но есть пример правильного файла, то следует осуществлять фаззинг путем модификации (мутации) оригинального файла. Разрабатываемый фаззер должен иметь режим работы, основанный на осуществлении мутаций входных данных с учетом покрытия кода программы.

В рамках лабораторной работы необходимо получить в соответствии со своим вариантом исполняемый файл, конфигурационный файл и динамические библиотеки. Лабораторная работа выполняется СТРОГО в соответствии со своим вариантом.

В ходе выполнения лабораторной работы необходимо выполнить следующие действия:

1. Изучить основные типы ошибок в программном обеспечении (целочисленное переполнение, отсутствие проверки длины копируемых данных, переполнение буфера и другие).
2. Получить у преподавателя файлы в соответствии со своим вариантом.
3. Реализовать программу, осуществляющую фаззинг формата файла.
4. Реализованная программа должна осуществлять следующие действия:

— осуществлять изменение оригинального файла (однобайтовая замена, замена нескольких байт, дозапись в файл);

— заменять байты на граничные значения (0x00, 0xFF, 0xFFFF, 0xFFFFFF, 0xFFFFFFFF, 0xFFFF/2, 0xFFFF/2+1, 0xFFFF/2-1 и т.д.);

— иметь автоматический режим работы, при котором производится последовательная замена байт в файле;

— находить в файле символы, разделяющие поля (“,:=;”);

— расширять значения полей в файле (дописывать в конец, увеличивать длину строк в файле);

— осуществлять запуск исследуемой программы;

— используя средство динамической бинарной инструментации (DBI) (Intel Pin / DynamoRIO) осуществлять измерение покрытия кода во время фаззинга;

— реализовать режим работы фаззера с обратной связью на основе покрытия кода, основанный на сохранении измененных байт в файле с учетом их влияния на покрытие кода программы;

— обнаруживать возникновение ошибки в исследуемом приложении;

— получать код ошибки и состояние стека и регистров на момент возникновения ошибки;

— логировать в файл информацию о произошедших ошибках и соответствующих им входных параметрах (произведенные замены).

5. Разработать IDC/IDAPython-скрипт, осуществляющий следующие действия:

— поиск в программе функций ввода данных (fread, fscanf, read, fgets, …);

— поиск вызовов небезопасных функций (strcpy, sprintf, strncpy, memcpy, memmove, …).

6. Разработанный IDC/IDAPython-скрипт должен выводить следующую информацию:

— название найденной функции;

— адрес, откуда вызывается данная функция.

7. Используя разработанную программу для фаззинга формата файлов осуществить поиск в выданной программе уязвимости (целочисленное переполнение, отсутствие проверки длины данных), приводящей к переполнению буфера.

8. Сформировать файл, приводящий к краху исследуемого приложения.

9. Изучить с использованием дизассемблера и отладчика структуру программы и найти участок кода с уязвимостью. Провести исследование найденной уязвимости и условий ее эксплуатации.

10. В отчете необходимо привести следующую информацию:

— исходный код программы, осуществляющей фаззинг формата файла;

— описание методов, применяемых при фаззинге;

— участок кода с найденной уязвимостью (из дизассемблера), с комментариями;

— восстановленный исходный код (на языке C) участка с уязвимостью;

— описание найденной уязвимости (тип ошибки, причины, ограничение при эксплуатации);

— размер буфера, адрес начала буфера, адрес сохраненного на стеке адреса возврата (return address, за буфером);

— формат файла, используемого для эксплуатации, с указанием значения полей, приводящих к краху программы;

— описание уязвимости и механизмов ее эксплуатации;

— исходный код разработанных IDC/IDAPython-скриптов;

— описание возможного способа исправления уязвимости.

Ссылки

1. Майкл Дж.Д. Саттон, Адам Грин. Fuzzing. Исследование уязвимостей методом грубой силы.
2. Тобиас Клейн. Дневник охотника за ошибками. Путешествие через джунгли проблем безопасности программного обеспечения.
3. Gray Hat Python.

https://codeby.net/threads/kniga-gray-hat-python-s-perevodom.58819/

4. IDAPython

https://github.com/idapython/src

https://github.com/idapython/bin

5. The Beginner's Guide to IDAPython.

https://leanpub.com/IDAPython-Book

6. Practical Malware Anaysis. Michael Sikorski, Andrew Honig.
https://doc.lagout.org/security/Malware%20%26%20Forensics/Practical%20Malware%20Analysis.pdf
7. The IDA Pro Book.

https://paper.bobylive.com/Tools/The%20IDA%20Pro%20Book%20%20The%20Unofficial%20Guide%20to%20the%20World%20%20039%20s%20Most%20Popular%20Disassembler.pdf