По результатам исследования в последовательностях, полученных из генераторов QCG-I, CCG, BBSG, MSG, G-SHA были выявлены небольшие статистические отклонения – 5-6 последовательностей из 100 не прошли тесты при норме 4 или меньше. В генераторах LCG, XORG и архиваторах gz, rar, xz тесты на основе вычисления расстояния Хемминга и Левенштейна выявили серьёзные статистические отклонения – более 10 последовательностей из 100 не прошли тесты. NIST STS также выявил отклонения в генераторах QCG-I, CCG, BBSG, MSG, G-SHA, LCG, XORG и архиваторах gz, rar, xz.

Можно сделать вывод, что тесты на основе вычисления расстояния Хэмминга и Левенштейна не хуже пакета NIST STS, так как забраковали те же самые генераторы. Так же стоит отметить, что предложенные в работе тесты имеют преимущества по скорости выполнения и ресурсным затратам, что позволяет использовать их для оценки ГСЧ в смартфонах, планшетах и других устройствах, не имеющих больших вычислительных мощностей.

Литература

1. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Special Publications 800-22. Revision 1.a April, 2010.

2. Б. Я. Рябко, А. И. Пестунов. «Стопка книг» как новый статистический тест для случайных чисел. Проблемы передачи информации. 2004. том 40, выпуск 1. С. 73–78

3. А. И. Миненко. Экспериментальное исследование эффективности тестов для проверки генераторов случайных чисел. Вестник СибГУТИ. 2010. № 4. С. 36-46

4. Чугунков И.В. Методы и средства оценки качества генераторов случайных последовательностей, ориентированных на решение задач защиты информации: Учебное пособие. М.: НИЯУ МИФИ, 2012. – 236 с.

Р.Р. Закиров

ЗАЩИТА ПРОГРАММ ОТ ОТЛАДКИ

(Казанский национальный исследовательский технический университет имени А.Н. Туполева – КАИ)

Процесс отладки позволяет злоумышленнику выяснить, по какому адресу выполняется программа, также подделать ее переменные в необходимые значения. Благодаря этому изучение программы конкурентами или пользователями, которые хотят воспользоваться платной программой бесплатно, облегчается в разы, чем при дизассемблировании и статичном изучении машинного кода программы.

Из этого следует, что защита от данного метода реверс-инжиниринга является наиболее актуальной для программистов. Поскольку сейчас популярна

не только архитектура x86, но и x64, многие старые средства обнаружения отладчиков устарели. В данной статье будут проанализированы самые полезные методы антиотладки, которые работают и на x86, и на x64 архитектурах.

Функция IsDebuggerPresent().

Самый простой способ – применение функции Api Windows IsDebuggerPresent(), которая есть во всех версиях Windows, начиная с Windows 98. Для этого достаточно внести в код строку «if (IsDebuggerPresent())» и, при положительном значении, например, вызвать экстренное завершение программы.

Так как данная функция очень, а ее имя будет находиться в исполняемом файле, настоятельно рекомендуется изменить ее имя на какое-либо менее заметное. Однако опытные хакеры смогут обойти данную защиту, поэтому далее будут приведены нетривиальные способы обнаружения отладчиков.

Измерение времени выполнения команд.

Когда отладчик присутствует и выполняет пошаговую трассировку, появляется существенная задержка между выполнением отдельных команд, если сравнивать с обычным выполнением. Существует огромное количество способов замера временного промежутка. Например:

- команда RDTSC;
- API-функция GetTickCount;
- API-функция timeGetTime (из winmm.dll);
- API-функция QueryPerformanceCounter;
- API-функция GetSystemTimeAsFileTime;
- API-функция GetProcessTimes;
- API-функция KiGetTickCount (или вызов прерывания int 0x2A);
- API-функция NtQueryInformationProcess (ProcessInformationClass = ProcessTimes (0x04);
- API-функция NtQueryInformationThread (ThreadInformationClass = ThreadTimes (0x01);
- поля структуры KUSER_SHARED_DATA.

Следует помнить, что в некоторых программах для отладки существуют готовые плагины и скрипты нейтрализации некоторых функций. Рекомендуется измерять достаточно большой промежуток времени (например, 10 секунд) и проводить проверку изменения значения регистров, отвечающих за выполнение функций замера времени. Саму функцию замера времени следует запускать в отдельном потоке, чтобы не мешать выполнению основных команд при этом усложняя нейтрализацию данного способа злоумышленником.

Проверка родительского процесса.

Данный метод является одним из самых действенных. Здесь проверяется кем именно было запущено защищаемое приложение: пользователем или отладчиком. Существует два пути реализации данного способа – проверить, является ли родительским процессом explorer.exe или не выступает ли в этой роли процесс из списка известных программ отладки.

Проверять родительский процесс мы будем при помощи уже известной

нам функции NtQueryInformationProcess и структуры PROCESS_BASIC_INFORMATION (поле InheritedFromUniqueProcessId), а получать список всех запущенных процессов в системе можно при помощи CreateToolhelp32Snapshot/Process32First/Process32Next. Основной код получения ID родительского процесса и основная проверка:

```
PROCESS_BASIC_INFORMATION baseInf;
NtQueryInformationProcess(NtCurrentProcess(), ProcessBasicInformation, &baseInf, sizeof(baseInf), NULL);
```

Таким образом, в baseInf.InheritedFromUniqueProcessId будет находиться ID процесса, который порождает защищаемый процесс. Его можно использовать как угодно: например, получить из него имя файла, название процесса и сравнить с именами отладчиков или проверять, не explorer.exe ли это.

NTSETINFORMATIONTHREAD.

Этот нетривиальный метод антиотладки основан на передаче флага HideFromDebugger (нахоится в структуре _ETHREAD за номером 0x11) в функцию NtSetInformationThread. Так выглядит прототип функции:

```
NTSTATUS ZwSetInformationThread(
_In_  HANDLE           ThreadHandle,
_In_  THREADINFOCLASS  ThreadInformationClass,
_In_  PVOID            ThreadInformation,
_In_  ULONG            ThreadInformationLength
);
```

Этот прием спрячет поток защищаемого процесса от отладчика, переставая отправлять ему отладочные события, например такие, как срабатывание точек останова. Особенность этого метода в том, что он универсален и работает благодаря штатным возможностям ОС. Код, который реализует отсоединение главного потока программы от отладчика:

```
NTSTATUS stat = NtSetInformationThread(GetCurrentThread(), 0x11, NULL, 0);
```

Таким образом, в данном докладе проведен анализ наиболее актуальных способов выявления и защиты от отладочных программ. Использование данных методов однозначно затруднит процесс отладки и поможет защитить приложения от любопытных реверсеров и автоматических систем распаковки и анализа. Стоит понимать, что данные методы следует использовать в совокупности друг с другом, чтобы намного усложнить процесс отладки.

Литература

1. Крупнейший в России и Европе ресурс в области прикладной информационной безопасности [Электронный ресурс]. URL https://xakep.ru/ (дата обращения: 15.03.2018)

2. Справочник исследователя программ [Электронный ресурс]. URL https://exelab.ru/ (дата обращения: 15.03.2018)

3. Microsoft Developer Network [Электронный ресурс]. URL https://msdn.microsoft.com/ (дата обращения: 15.03.2018)