МЕТОДЫ ПРОТИВОДЕЙСТВИЯ РЕВЕРСИНГУ

Автор: Полянин Андрей, студент 4 курса
Руководитель: Кульман Никита Юрьевич, доцент кафедры «Информационные технологии»
Образовательное учреждение: филиал «Протвино» Международного университета природы, общества и человека «Дубна», г. Протвино

METHODS OF RESISTANCE TO REVERSING

Polyanin A.G.

Защита программ

По данным международной ассоциации Business Software Alliance среднемировой показатель пиратства составил в 2011 году 42%. В России это величина составляет 64%. Это означает, что вы рискуете потерять до 2/3 своего дохода, если вовремя не позаботитесь о соответствующих мерах защиты программ от взлома, копирования и нелегального распространения. Необходимость в защите программного обеспечения (ПО) возникает не только на государственном или корпоративном уровне, но и на уровне средних и малых предприятий, и даже — на индивидуальном. При разработке любой важной программы необходимо использовать системы защиты для предотвращения нелицензионного копирования и распространения. В случае пренебрежения к защите продукта компания или индивидуальный разработчик могут понести серьёзные убытки. Наблюдая в сети Интернет многочисленные пиратские версии продуктов типа *Photoshop, 3dMax* и др. можно только догадываться, сколько потеряли компании-разработчики. И это несмотря на то, что, скорее всего, деньги на защиту программ были выделены серьёзные.

Помощь в защите продукта даёт и законодательство России. Вступление России в ВТО также помогает защищать авторские права.

Реверсинг

«…Термин *Reverse engineering* (или *reversing*) определяется как анализ, разбор (расшифровка) конструкции, структуры, построения программного или аппаратного изделия; восстановление структурной схемы и алгоритма работы; проектирование по образцу; воспроизведение, обратная инженерия. Всё это обычно представляет собой процесс систематического разбора программы (восстановления её исходного текста и структуры) или микросхемы для изучения алгоритмов работы с целью имитации или повторения некоторых или всех её функций в другой форме или на более высоком уровне абстракции, снятия защиты, добавления новых возможностей, восстановления протоколов или исправления ошибок и др…»[1]

От реверсинга необходимо защищаться, поскольку такое исследование продукта приводит к нежелательным последствиям, например, созданию «кряков» программы. Существуют некоторые программы, которые не поддаются дизассемблированию, например *Skype*.

Антиреверсинг

Для антиреверсинга применяются различные методы — от обнаружения отладчика и «родительского процесса» программы до определения использования виртуальной машины и подключения чужих программ с выполнением впоследствии несанкционированных действий.[2]

Рассмотрим некоторые из методов, которыми должен пользоваться программист для защиты своих программ от исследования.

Для затруднения исследования программы под отладчиком **ну**жно шифровать строковые переменные. Например, хранить данные в зашифрованном виде, и расшифровывать их только на короткий промежуток времени, необходимый для выполнения операций, а затем снова скрывать. Это позволит не предоставлять информацию в явном виде в большой отрезок времени, что осложнит процесс вскрытия программы. Можно пойти дальше и зашифровать не только данные, но и код. Однако, это сулит большие сложности разработчикам для обеспечения работоспособности программы.

Для выявления отладчика можно использовать несколько специальных API-функций замера времени выполнения команд: RDTSC, GetTickCount, timeGetTime и др. Некоторые функции измеряют время в тактах процессора, некоторые — в секундах, но схема работы примерно одинакова. Идея состоит в том, чтобы измерить время до и после выполнения некоторой команды и сравнить это время с установленной величиной. Если она укладывается в нужный период и соответствует ожиданиям, то отладчика нет. Если же она выходит за рамки, то нас отлаживают. В настоящее время большинство таких функций известно, и в плагинах популярных отладчиков, таких как OllyDbg, их можно перехватить и обойти. Но реверсерам придётся делать это аккуратно, потому что с помощью таких функций и обработчиков прерываний можно отследить перехват и нейтрализовать его.[3]

Наличие отладчика можно выявить и «напрямую», применяя другие специальные функции такие, как DebugBreak и RaiseException. Эти функции тоже «популярны» и есть вероятность, что их перехватят. Но здесь, как и при анализе времени выполнения, можно использовать следующий способ защиты: обработчики исключений и флагов.

При перехвате и нейтрализации API-функций генерируются исключения, особые значения некоторых флагов, которые мы можем перехватить и использовать в своих целях. Например, наличие отладчика можно определить по значению PEB+0x02 флага BeingDebugged.

Достигнутые результаты

Для того, чтобы узнать, отлаживает ли кто-то нашу программу используем функцию GetTickCount:

```
DWORD TimeStart=GetTickCount();
Sleep(100);
DWORD TimeEnd=GetTickCount();
```

Таким образом, запомним время до выполнения команды Sleep() и сравним с тем, что будет после выполнения:

```
if (((TimeEnd-TimeStart) < 91) || ((TimeEnd-TimeStart) > 113))
        b=true;
  else
        b=false;
```

Если интервал больше или меньше положенного времени, то обнаружим наличие отладчика:

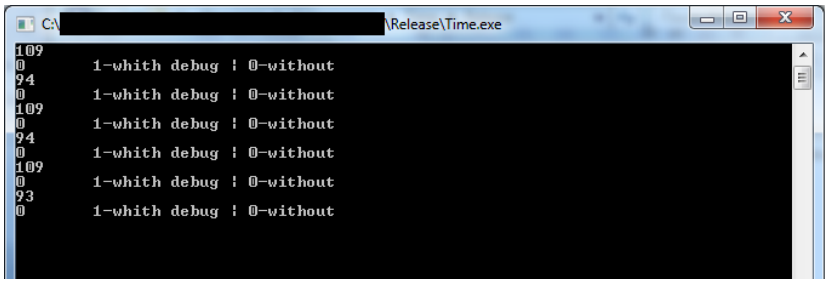

Рисунок 1 — Без отладчика

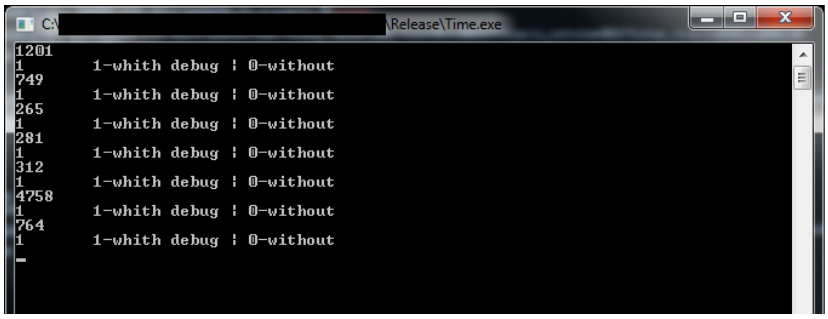

Рисунок 2 — С отладчиком

Если нам необходимо узнать, отлаживает ли кто-то нашу программу, то можно использовать функцию CheckRemoteDebuggerPresent(hProcess,&bDebug), где hProcess — наш процесс, а bDebug — переменная булевого типа, в которую функция занесёт результат проверки. Исходный код:

```
DWORD processID = GetCurrentProcessId();
 HANDLE hProcess = OpenProcess(PROCESS_QUERY_INFORMATION | PROCESS_VM_READ | PROCESS_SET_QUOTA,
        FALSE,processID);
BOOL bDebug;
CheckRemoteDebuggerPresent(hProcess,&bDebug);
```

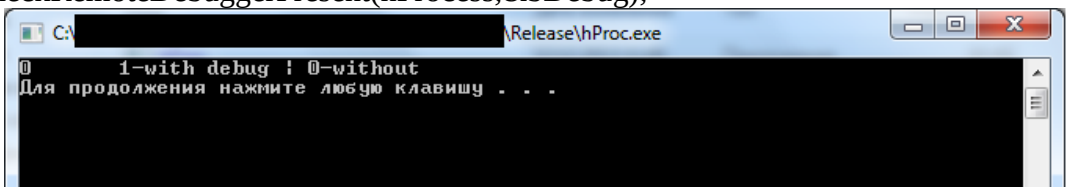

Рисунок 3 — Без отладчика

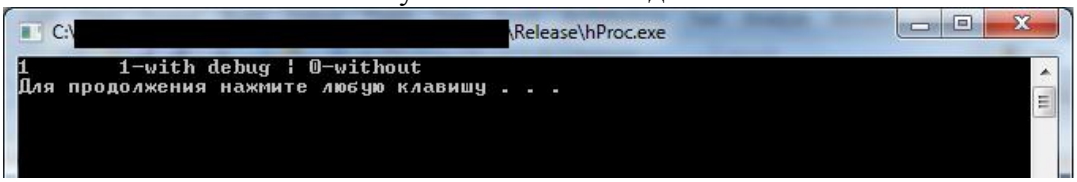

Рисунок 4 — С отладчиком

Как видим, значения разные в различных условиях исполнения. Применять такой метод удобнее, чем самим что-то проверять, но и опаснее, ведь, как говорилось выше, большинство API-функций сегодня можно перехватить.

Перспективы изучения

В рамках дальнейшего изучения реверсинга и защиты программ нужно изучить новые способы противодействия реверсингу, ознакомиться с другими программами в данной области информационных технологий.

Рынок и объём программного обеспечения сегодня растут с каждым днём, поэтому растёт число украденных программ и программ, которым необходима защита от взлома. Исходя из этого, сфера защиты программ тоже не стоит на месте: постоянно разрабатываются новые технологии взлома, а, следовательно, нужно изучать и совершенствовать и методы защиты.

Библиографический список

1. Панов, А.С. Реверсинг и защита программ от взлома. – СПб.: БХВ-Петербург, 2006. – 256 с: ил.
2. Электронный ресурс: http://www.anti-reversing.com
3. Электронный ресурс: http://www.xakep.ru/xa/?id=177