Устройство загружаемого кода

Прямой перенос кода из исходного приложения может быть сопряжен с определенными трудностями. В общем случае, код, перенесенный в том же виде, как он существует в приложении, будет неработоспособен в электронном ключе. Поэтому код должен быть модифицирован и оптимизирован для выполнения на платформе CORTEX-M3. Желательно, чтобы этот код был написан заново и реализовывал функции, которых в ранних версиях приложения не было, либо эти функции должны быть видоизменены.

Прототип функции **main()** объявляется следующим образом:

```
DORD main(
DWORD dwInDataLng, DWORD dwOutDataLng, DWORD dwP1)
```

Параметры **dwInDataLng** и **dwOutDataLng** устанавливают количество данных, считываемых из буфера ввода и возвращаемых в буфер вывода. Параметр **dwP1** используется для передачи кода подфункции загружаемого кода.

Параметр **dwP1** передается функции **GrdCodeRun()** и его можно получить в загруженном коде в виде третьего параметра функции **main** (функции, которая первой получает управление в С-коде):

```
DWORD func1(dwInDataLng, dwOutDataLng)
{
// Логика работы 1:
return 101;
}
DWORD func2(dwInDataLng, dwOutDataLng)
{
// Логика работы 2:
return 102;
}
DWORD func3(dwInDataLng, dwOutDataLng)
{
// Логика работы 3:
return 103;
}

DWORD main(DWORD dwInDataLng, DWORD dwOutDataLng, DWORD dwP1) {
switch (dwP1)
{
case 0x01:
return func1(dwInDataLng, dwOutDataLng);
case 0x02:
return func2(dwInDataLng, dwOutDataLng);
case 0x03:
return func3(dwInDataLng, dwOutDataLng);
case 0x04:
// ...
default:
return -1;
}
}
```

По возможности, переменные лучше объявлять глобально (хотя это и противоречит принципам функционального программирования), а не в теле функции, чтобы не передавать данные через стек. Этим экономится память стека и увеличивается быстродействие.

В загружаемом коде можно создать глобальные переменные, содержимое которых не будет обнуляться между вызовами. Такие переменные требуется объявлять с макросом **NO_INIT**:

```
DWORD buffer[100] NO_INIT
```

Эти переменные являются аналогами статических переменных.

Польза от них может заключаться в возможности запоминания некоторых состояний загружаемого кода. Это делает анализ «черного ящика» гораздо более сложным.
Выход из приложения можно осуществлять следующим образом:

- Возврат из main при помощи **return**. Код возврата будет помещен в параметр **dwRetCode** функции **GrdCodeRun()**
- Вызов функции **GcaExit()**. Ей также передается код возврата

Кроме того, принудительное завершение приложения происходит в следующих случаях:

- Наступление таймаута времени выполнения загружаемого кода (3 секунды)
- Попытка выполнения приложением недопустимого действия (обращение к недопустимым адресам памяти и т.д.)

В примерах в качестве кода возврата с ошибкой используется значение -1. Для упрощения отладки можно возвращать значения макроса **_LINE_** или пользоваться вызовом **GcaExit(0, _LINE_)**. Этот способ поможет определить строку, на которой произошел выход из приложения. Оставлять в конечных версиях возвраты в данном виде нежелательно, так как это может дать дополнительную информацию для злоумышленника.

Для отладки можно, к примеру, использовать следующий макрос:

```
#define ASSERT(cond){if(cond)GcaExit(0,_LINE_);}
```

ASSERT(x != 0); // Если x!=0, осуществит возврат из программы с указанием номера строки, в которой вставлен **ASSERT.**
За вычисления с плавающей точкой отвечает библиотека **libm** из комплекта **GCC.** Полное описание математических функций, доступных в ней, можно найти в документации к данной библиотеке. Для каждой функции имеется 2 варианта: обычный, для вычислений с двойной точностью (тип double), а также с приставкой «f», для вычислений с половинной точностью (тип float).