Digital Security

JIT SPRAY

Алексей Синцов

Ведущий аудитор по ИБ Digital Security

Атаки на клиентов

Объект атаки:

- Браузеры
  - Internet Explorer
  - FireFox
  - Плагины/ActiveX
    - Банк-Клиенты
    - Бизнес-приложения
- Иное ПО
  - MS Office
  - Adobe Acrobat Reader
  - Adobe Flash

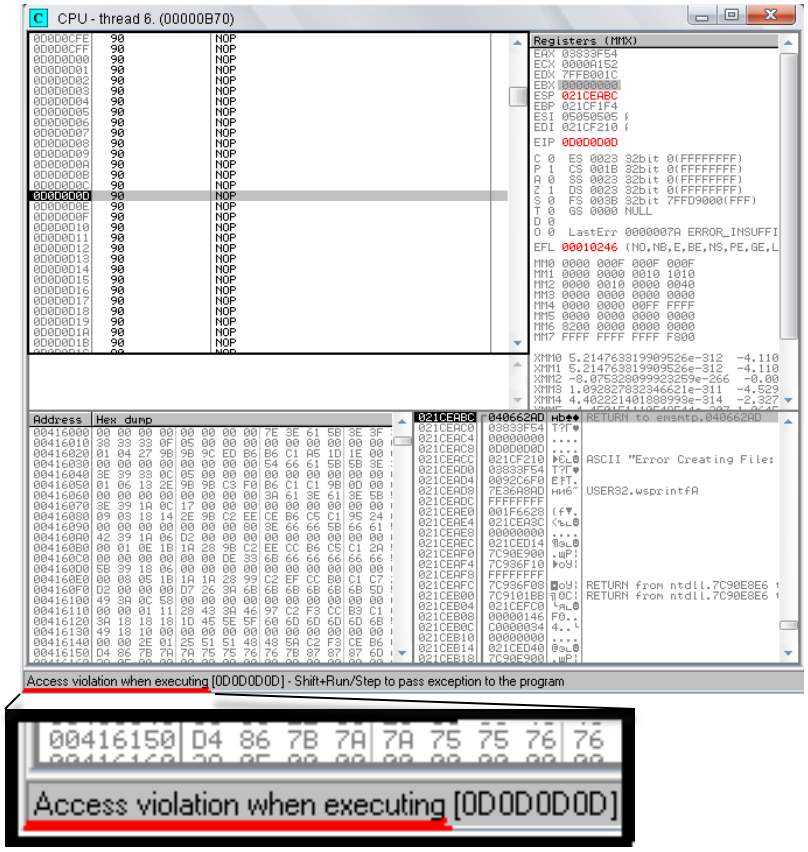
Источник атаки:

- HTML/JS
- SWF
- PDF
- DOC*

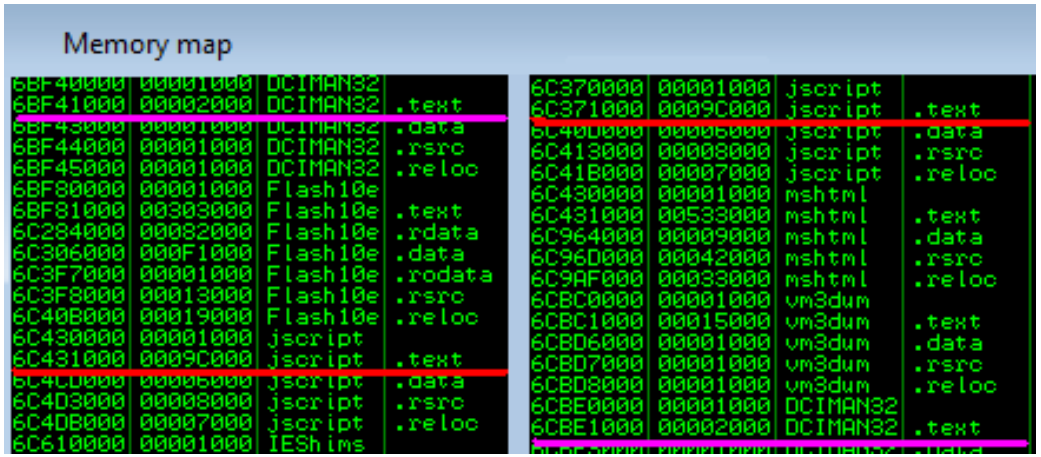
Эксплойт

Защита

DEP



ASLR



Атаки

DEP

- ROP – отключение DEP
- ROP – вызовы VirtualProtect/VirtualAlloc и memcpy

ASLR

- Поиск несовместимых DLL
- Использование связок уязвимостей

Pwn2own 2010:

- Взломан IE8 на Windows 7

http://vreugdenhilresearch.nl/Pwn2Own-2010-Windows7-InternetExplorer8.pdf

- Взломан FireFox 3.6 на Windows 7

- Эксплойт CVE-2010-0188

http://blog.metasploit.com/2010/03/latest-adobe-exploit-and-session.html

Не универсально

JIT SPRAY

Дионис Блазакис представил на BlackHat DC 2010 технику JIT SPRAY и способ вычисления адреса объекта в памяти:

http://www.semantiscope.com/research/BHDC2010/BHDC-2010-Paper.pdf

- Обход permanent DEP
- Обход ASLR

Нужен Flash

Вычисление адреса работает около 8-10 минут

Нет PoC

Не раскрыта тема шеллкода

JIT SPRAY: обходим DEP?

Инъекция команд через код ActionScript

```
var ret=(0x3C909090^0x3C909090^0x3C909090^0x3C909090);
```

```
0x1A1A0100: B89090903C MOV EAX,   3C909090
0x1A1A0105: 359090903C XOR EAX,   3C909090
0x1A1A010A: 359090903C XOR EAX,   3C909090
0x1A1A010F: 359090903C XOR EAX,   3C909090
```

Исполняемая память

JIT SPRAY: обходим DEP!

0x1A1A0100: B8**90**90903C MOV EAX, 3C**90**9090

0x1A1A0105: **35**9090903C **XOR EAX**, 3C909090

0x1A1A010A: **35**9090903C **XOR EAX**, 3C909090

0x1A1A010F: **35**9090903C **XOR EAX**, 3C909090

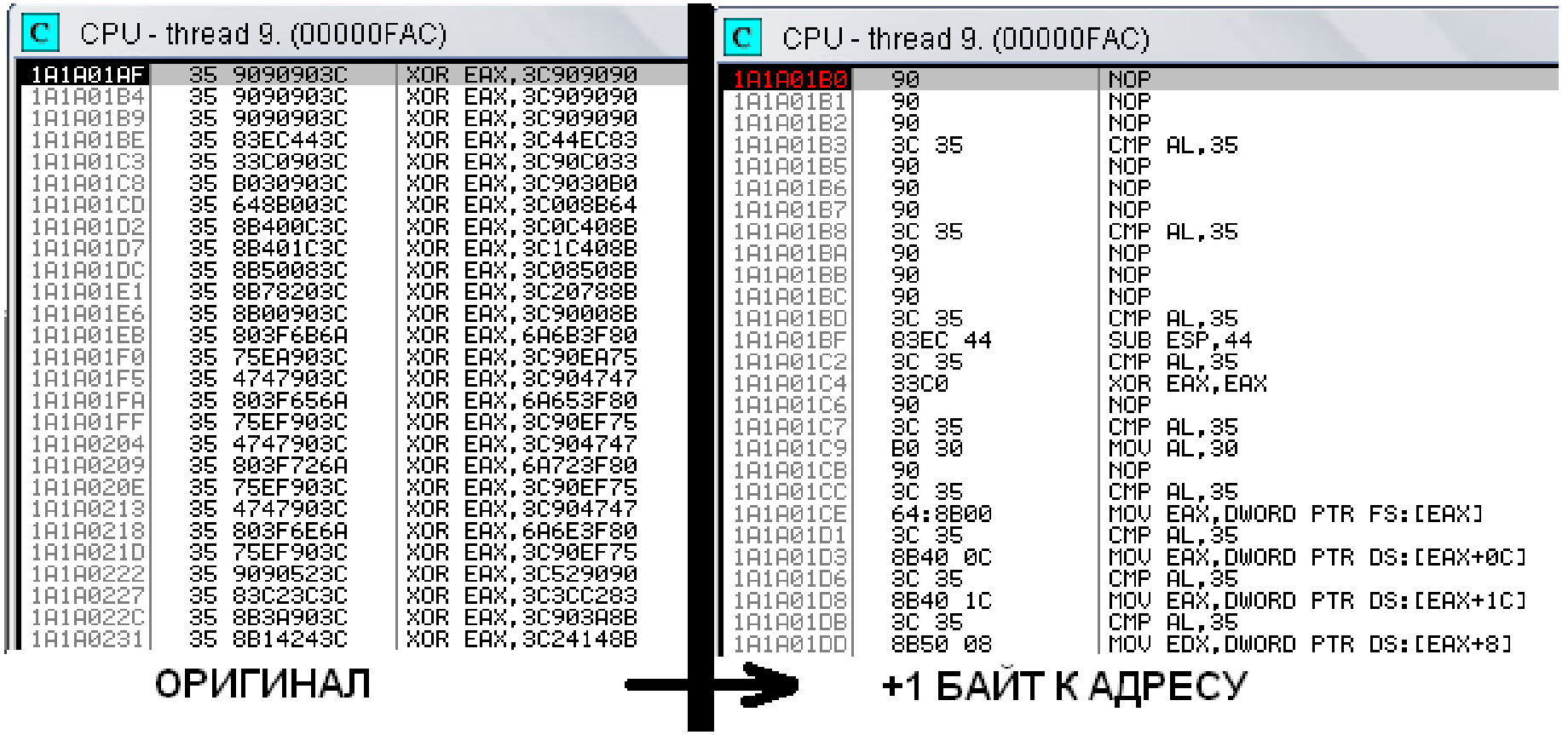
Исполняемая память

Смещаемся на 1 байт

0x1A1A0101: **90** **NOP**
0x1A1A0102: 90 NOP
0x1A1A0103: 90 NOP
0x1A1A0104: **3C35** **CMP AL, 35**
0x1A1A0106: 90 NOP
0x1A1A0107: 90 NOP
0x1A1A0108: 90 NOP
0x1A1A0109: **3C35** **CMP AL, 35**

Исполняемая память

JIT SPRAY: обходим DEP!



Обходим ASLR?

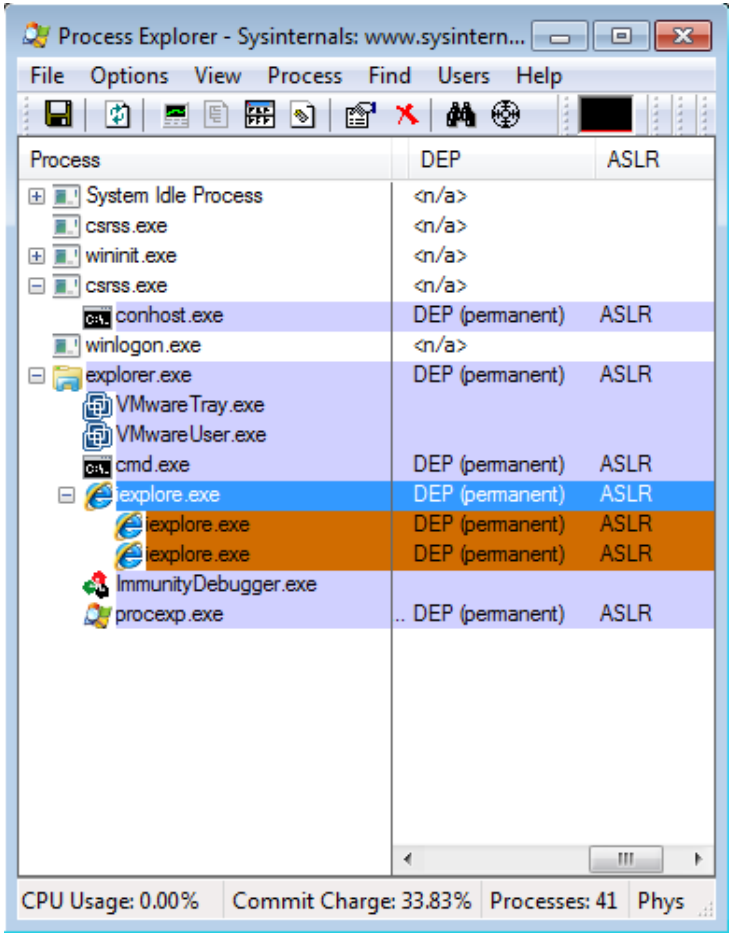
Время – деньги!

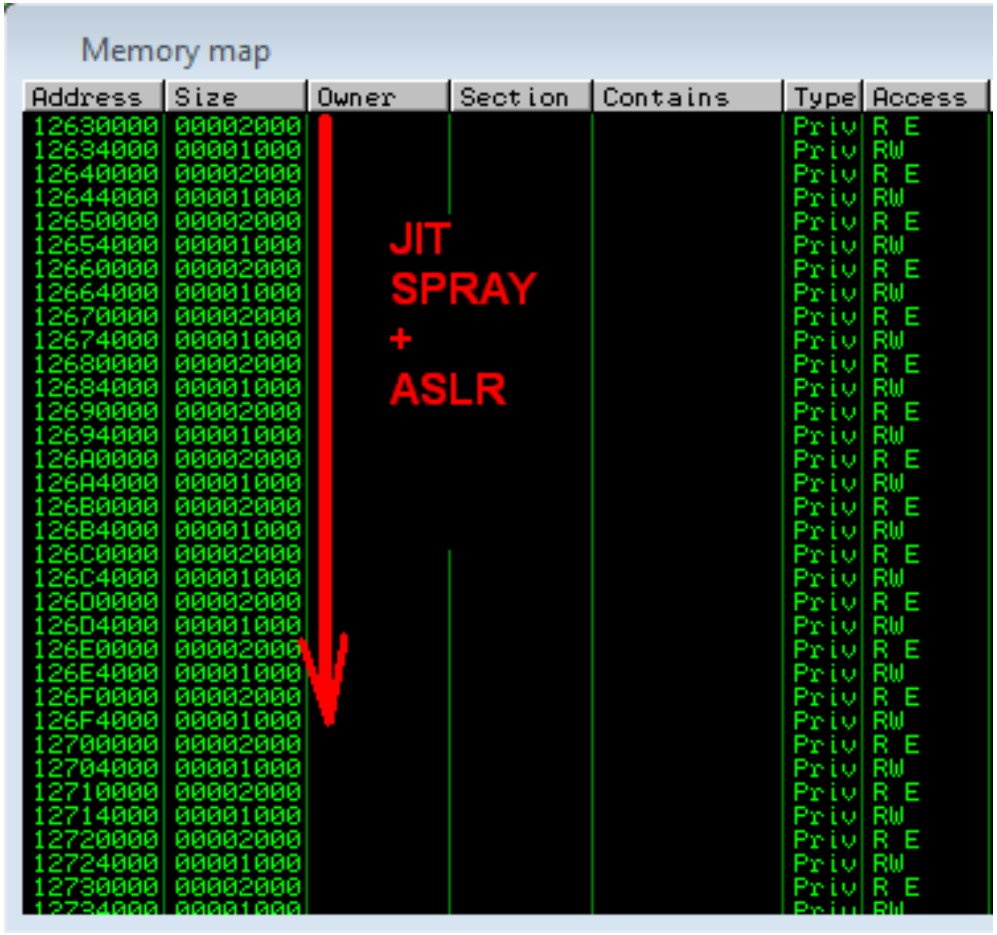
- Не будем вычислять адрес через утечки во Flash - долго
- Сделаем большое количество станиц в куче с JIT SPRAY
- Проанализируем память…

Выводы

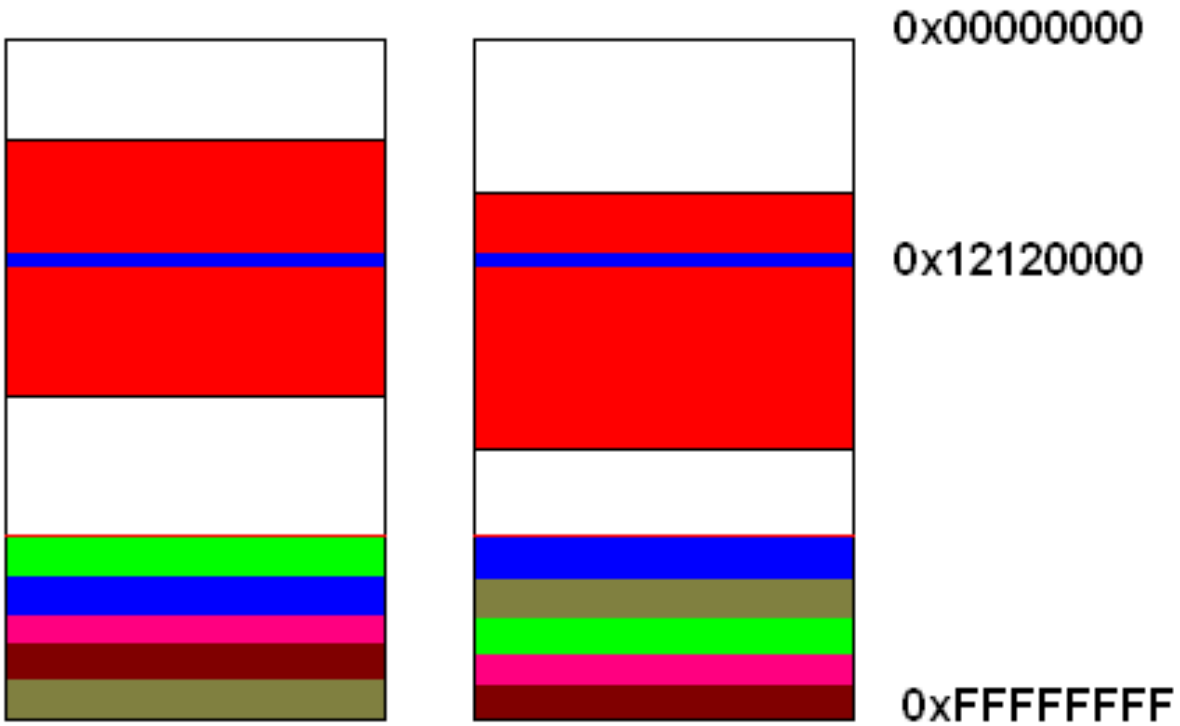
- ASLR в Windows 7 меняет базовые адреса модулей, но...
- Страницы с JIT SPRAY генерируются **ПОСЛЕДОВАТЕЛЬНО** в адресном пространстве, начиная с младших адресов.

Обходим ASLR!





Обходим ASLR!



JIT шеллкод - подготовка

Размер блока

От размера блока зависит сдвиг относительно следующего блока.

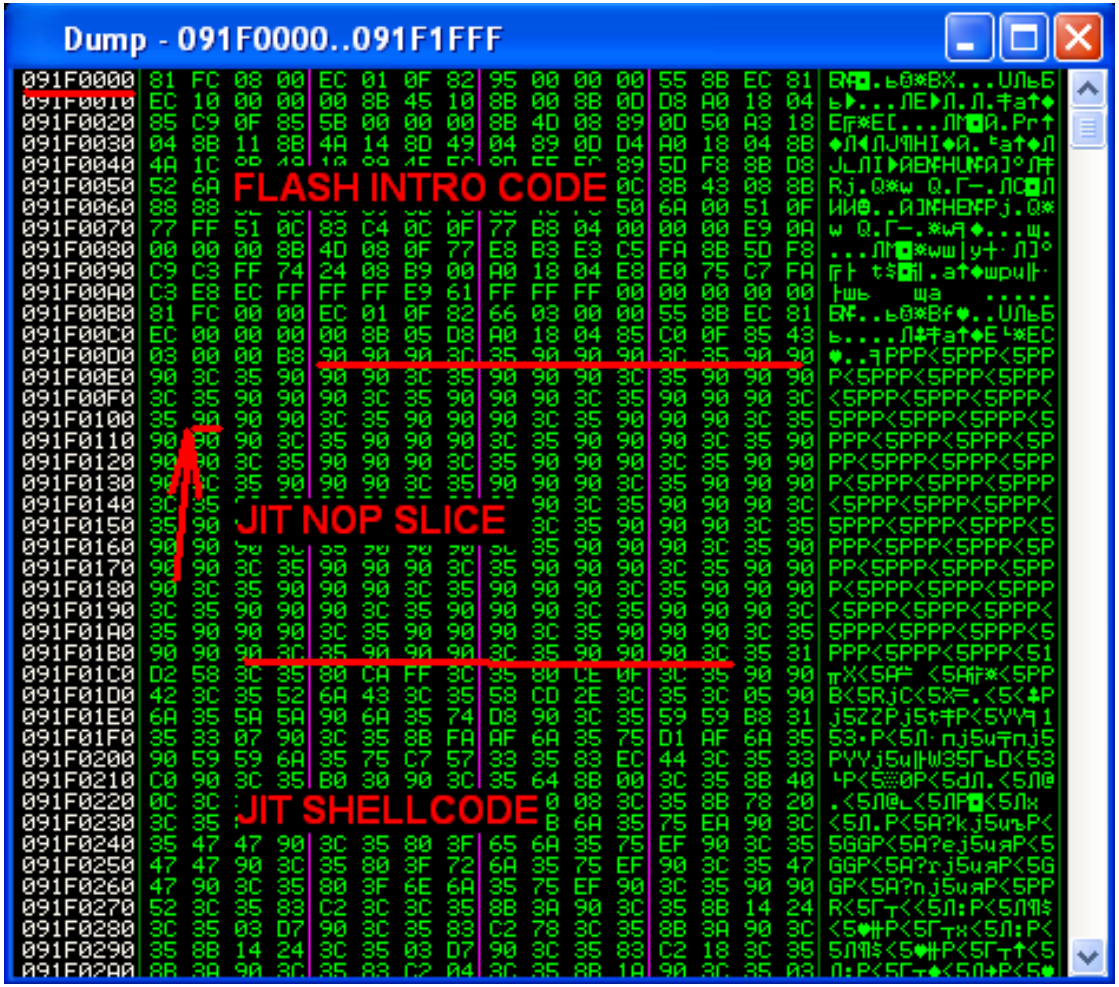
- 0xXXYY0000 – младшие разряды каждого блока – нули.
- Вводный код Flash – около 0x100 байт.
- Необходимо, чтобы сдвиг между исполняемыми блоками отличался на 0x00010000.
- Учитывая все это, велика вероятность попадания на шеллкод путём внедрения большого количества блоков. При тестировании – 100%. Аналогия с Heap Spray.

Боевая нагрузка

Дионис вычислял адрес боевого шеллкода в строке через утечку во Flash, передавал адрес в эксплойт. JIT шеллкод помечал память по полученному адресу как исполняемую и передавал управление коду из этой памяти.

- Долго
- Не универсально

Угадываем адрес - легко



JIT шеллкод - нагрузка

Egg-Hunter

- Размещаем боевой шеллкод в объекте Flash – в строке, а лучше в куче.
- JIT шеллкод ищет боевой шеллкод по метке.
- Помечает память с найденой меткой как исполняемую, передает управление.

Итог:

- Универсальнее (не только по адресу возврата, не только переполнение буфера)
- Надежнее (не зависим от старших разрядов, от тайм-аута Flash)
- Быстрее (поиск только в куче, поиск по известному сдвигу)

JIT шеллкод - howto

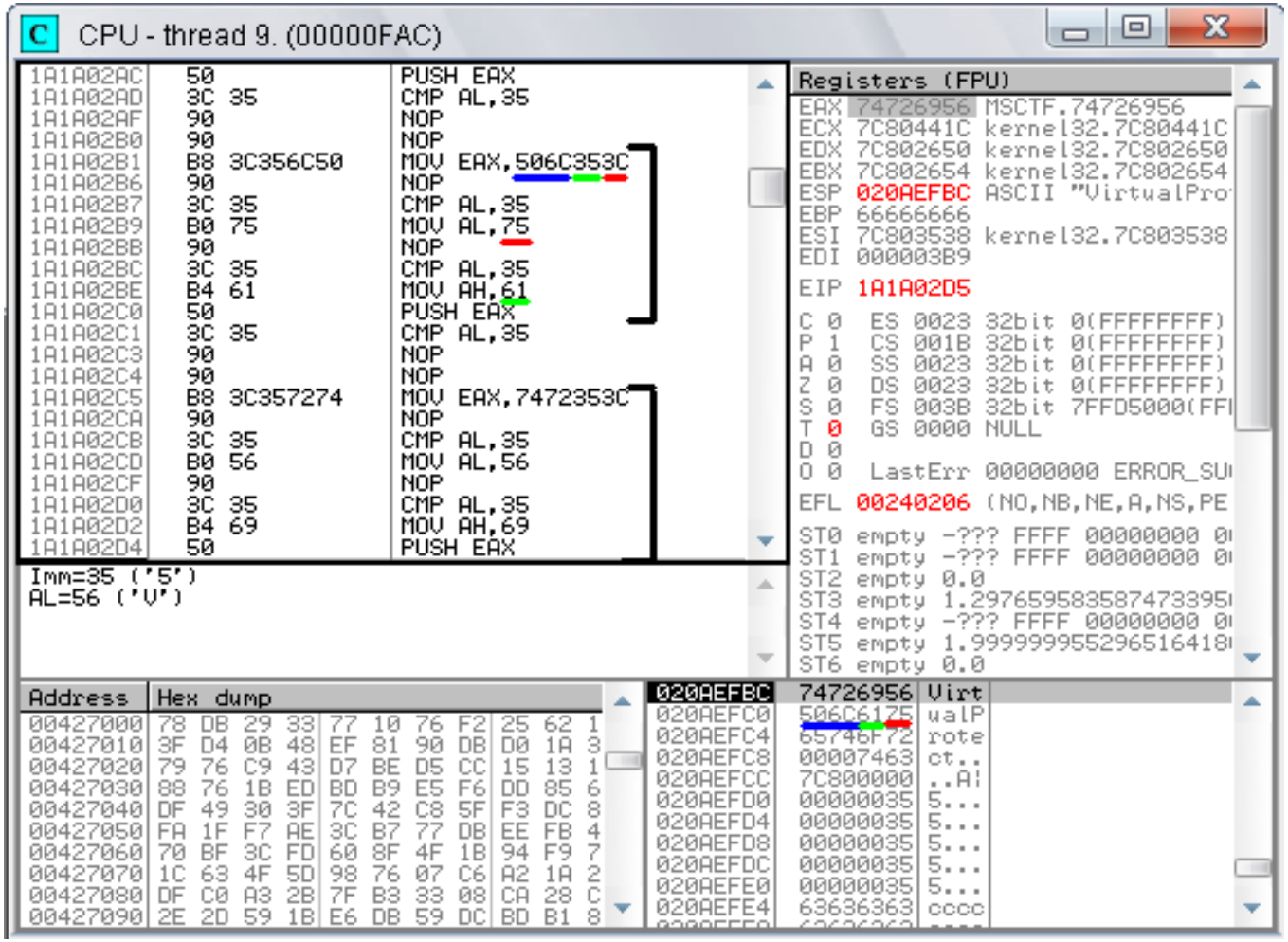
Проблемы:

- Старший байт не может быть больше 0x7F
- Сохранение Z флага при условном переходе

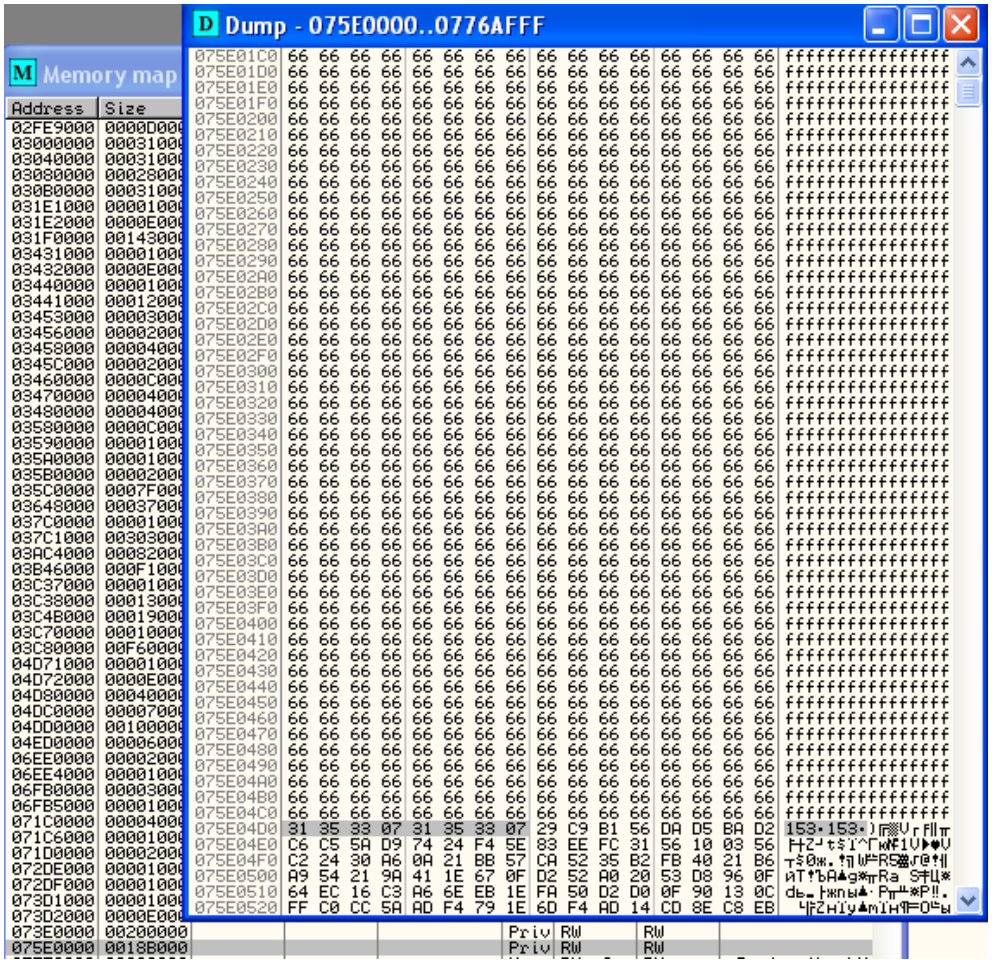
  Нельзя маскировать 0x35: CMP, SUB, ADD, OR …

  Единственное решение: **PUSH**, но усложняется работа со стеком
- Мы не можем работать полноценно с четырехбайтными аргументами
- Трудно работать с CALL

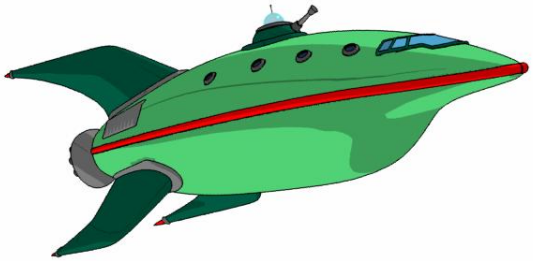
JIT шеллкод – работа с 4 байтами



JIT шеллкод - скорость



JIT SPRAY - эволюция



EGG-HUNTER STAGE 0 - < **30** сек

STAGE 0 - ~ **120** сек

STAGE 0 с утечкой памяти ~ **420** сек

Digital Security

Вопросы ?

a.sintsov@dsec.ru
dookie@inbox.ru