Inline Patch for Themida

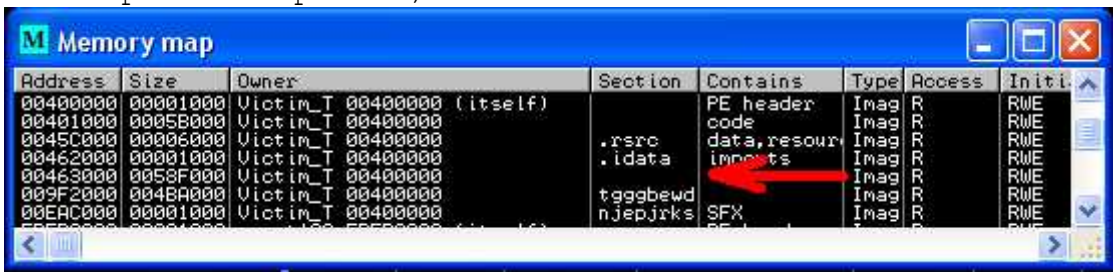
Автор: Maximus // Revenge CREW

[Введение]

В данной статье мы попытаемся с вами заинлайнить тестовую программу, накрытую коммерческим протектором Themida. Конечная цель инлайна будет запуск программы с ключом сгенерированным на другую машину. Так же я вам расскажу, как это сделать с забаненным ключом и с ключом, ограниченным по времени. В Архиве, кроме этого документа, я прилагаю защищенную и не защищенную программу. Исходники к ней и проект WinLicense. Итак, начнем’с

[Поиск Is_Registered двордов]

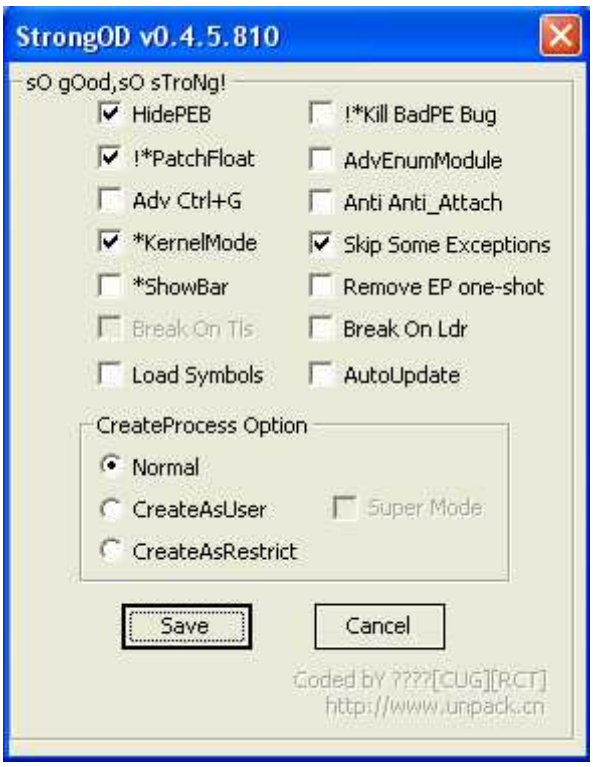
Основная исследовательская работа у нас будет происходить в секции VM Темиды. Она всегда третья, если считать с конца секций файла (отмечено красной стрелкой):



Я защитил приложение с максимально возможными опциями с виртуализацией 20% CISC VM машиной.

Для нормальной работы зарегистрированного приложения, протектор всегда пишет в отмеченную секцию два DWORD’а. Один DWORD говорит о том, зарегистрировано приложение или нет, а второй DWORD говорит о статусе ключа (забанен или нет, на наше железо или нет, просрочен ключ или нет). Зная эти два дворда можно запустить программу даже не имея ключа (например, для распаковки). Наша же задача, имея ключ, привязанный к железу направить программу по алгоритму, когда ключ работает на всех машинах. Для этого нам сначала надо найти DWORD отвечающий за статус ключа (второй DWORD).

Запустим OllyDebugger 1.10. Что бы обойти ультра антиотладку Themida, я использовал плагин StrongOD с следующими настройками:



Запускаем программу под отладчиком и получаем следующее сообщение:

Понятно, что в данный момент работает защищенный VM код Themida. Нам надо найти команду сравнения в нем. Хендл, который обрабатывает сравнение в Themida, выглядит так:

```
3BC8        CMP ECX,EAX
9C          PUSHFD
E9          JMP XXXXXXXX
```

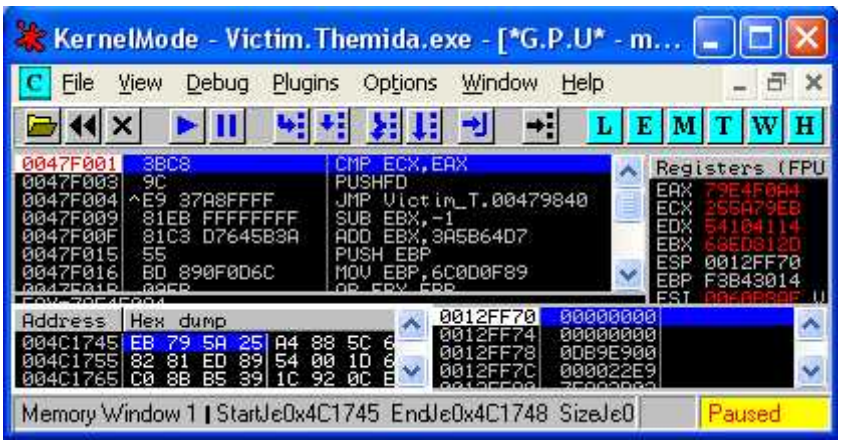
В RISC варианте будет выглядеть так:

```
3BCB        CMP ECX,EBX
9C          PUSHFD
E9          JMP XXXXXXXX
```

Следовательно делаем поиск по сигнатуре 3B C8 9C E9 в нашей исследуемой секции Темиды и находим место, куда мы поставим бряк:

```
0047F001    CMP ECX,EAX
0047F003    PUSHFD
0047F004    JMP 00479840
```

Нажимаем OK на сообщении о триале и жмем F9 до тех пор, пока у нас в регистрах EAX и ECX не появятся цифры отличные от нулей:



Обратите внимание, в EAX в данный момент у нас лежит правильный Is_Registered DWORD, а в ECX лежит наш неверный DWORD.

Далее, опять же в секции Темиды, ищем наш неверный DWORD и ставим на него хардварный бряк (адрес показан на скриншоте). На этом поиск дворда закончен.

Хочу отметить, что в новых версиях Темиды никаких сообщений может и не быть. Тогда обращение к Is_Registered DWORD можно найти уже после запуска программы, найдя по сигнатуре описанное сравнение (легко ловится после вызова Themida API функции WLRegGetStatus(ExtStatus)).

[Запуск программы с ключом сгенерированным на другой HWID]

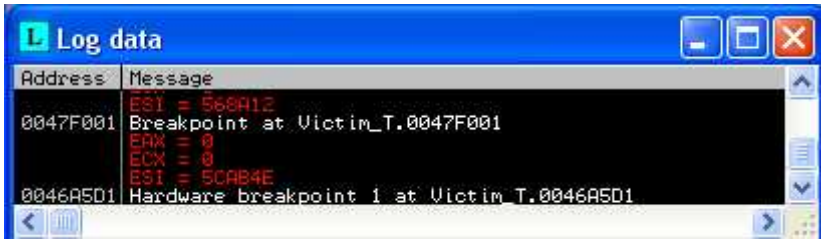
Теперь можно подложить наш ключик regkey.dat из папки WL в папку с программой-жертвой и перезапустить нашу программу. Давайте понажимаем F9 до тех пор, пока не увидим в нашей Is_Registered ячейке правильное значение:

Так как ключ у нас на другой HWID в какой-то момент в эту ячейку запишется неверное значение. Нам нужно этот момент отследить и направить алгоритм программы в нужное русло. Что бы отследить этот момент, напишем и запустим элементарный макрос (ленивые могут найти его в Victim\1.osc):

```
var addr
mov addr, 47F001 // Адрес нашей процедуры сравнения

bp addr
```

```
iteration:

esto
cmp eip, addr
jnz exit

log:
eval "EAX = {eax}"
log $RESULT, ""
eval "ECX = {ecx}"
log $RESULT, ""
eval "ESI = {esi}"
log $RESULT, ""
jmp iteration

exit:
ret
```

Выполним его и посмотрим в Log и увидим:



Это значит, что чтобы программа заработала, надо при заданном значении регистра ESI записать в ECX единицу и программа заработает. Для проверки наших слов исправим в макросе строки

```
cmp eip, addr
jnz exit
```

на

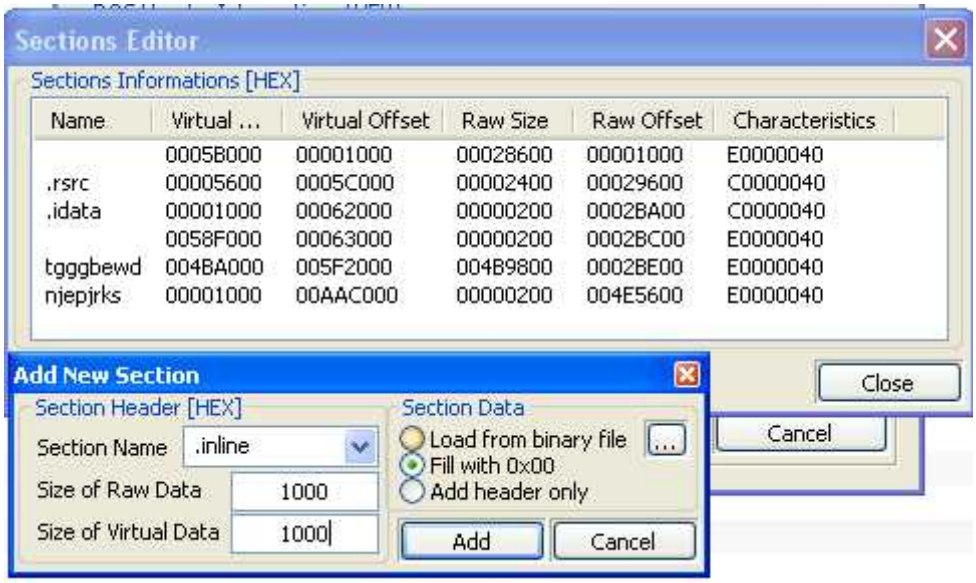
```
cmp esi, 5CAB4E   // проверка HWID
jz exit
```

перезапустим программу, опять дождаться когда в дворд у нас упадет правильное значение Is_Registered, и после того как отработает макрос снимем все бряки, запишем в ECX 1 и запустим программу, как видите программа запустилась.

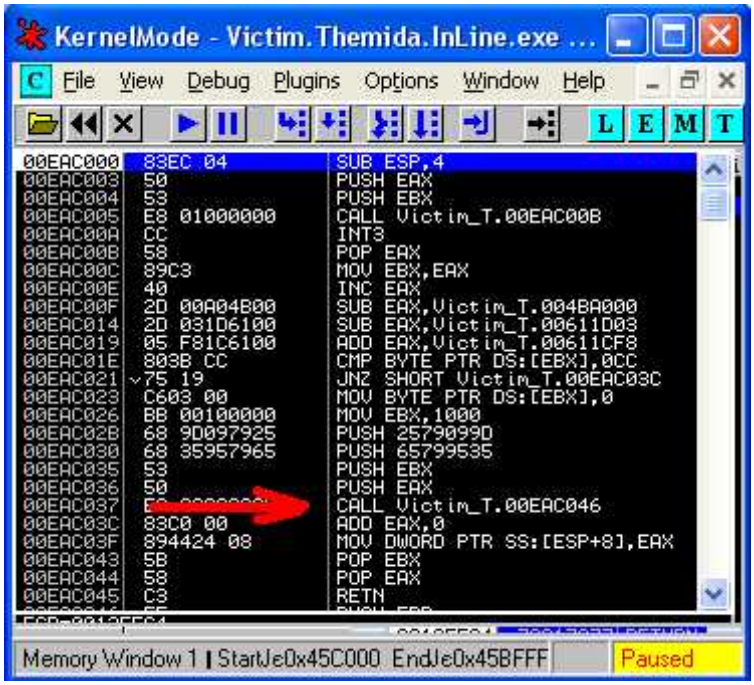
Похожий алгоритм изменения регистра EAX будет и при проверке забаненности ключа, и при проверке его времени действия.

[InLine Themida]

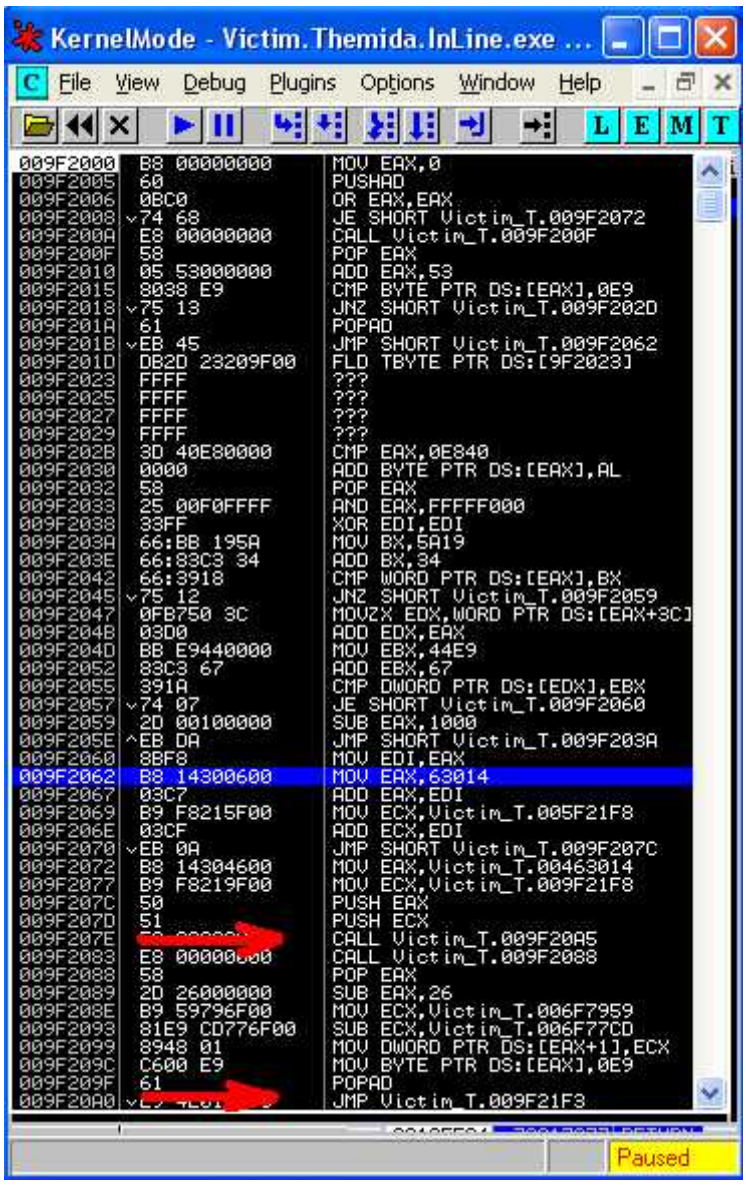
Место для патча нашли, теперь давайте инлайнить. Темида распаковывает себя в два захода. Сначала распаковывается загрузчик, а потом сама VM. Чтобы не мучиться с поиском байтов для патча, давайте добавим пустую секцию в конец файла размером 4 кб:



Если не использована защита от PE-сканера, то старт программы выглядит всегда одинаково



CALL что указан по стрелке распаковывает первый слой, вместо него мы вставим jmp на наш патч. После того как отработает CALL и мы выйдем из процедуры по команде RET мы провалимся во второй слой распаковки:



По первой красной стрелке происходит анпак ВМ, по второй красной стрелке мы сделаем заход в наш патч, что бы пропатчить наш CMP, где мы будем подменять значение регистра EAX. Итак начнем инлайнить:

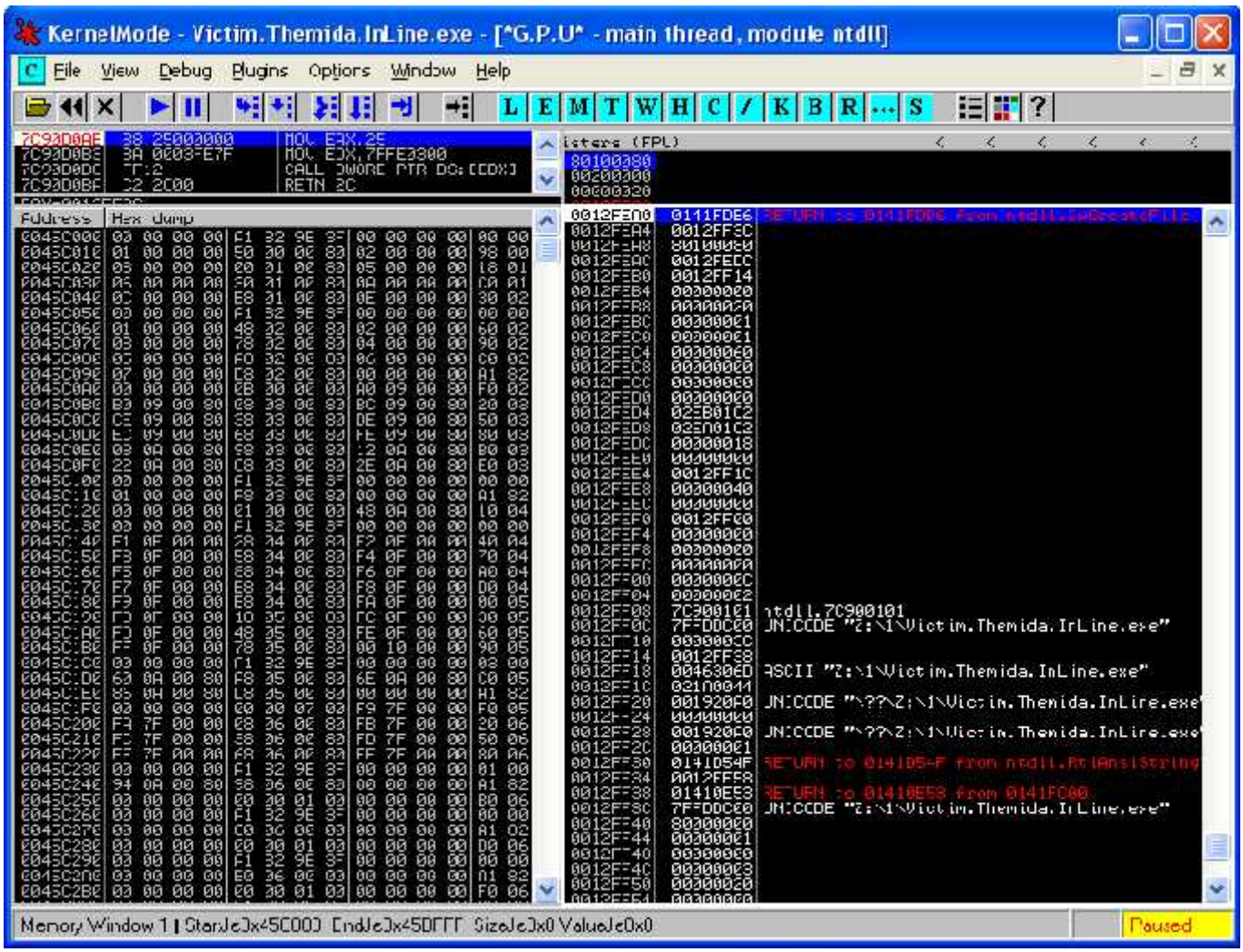
```
; Вход в наш In-Line патч, в добавленную секцию
      00EAC037    JMP 00EAD000

; Патч второго слоя
; Вызываем процедуру, которую мы затерли своим патчем, и распаковываем второй слой
      00EAD000    CALL 00EAC046

; Патчим второй слой так, что бы после распаковки VM управление программой
; вернулось к нам и мы смогли бы пропатчить наш CMP
; Для этого в место указанное второй красной стрелкой пишем код JMP 00EAD014
      00EAD005    MOV DWORD [9F20A1], 004BAF6F

; Возвращаем управление программе с прерванного места
      00EAD00F    JMP 00EAC03C

; Патч VM
; Патчим VM так, что бы после выполнения команды CMP управление программой
; вернулось к нам, и мы смогли бы в случае чего пропатчить значения регистров
```

```
; Для этого в первый JMP после команды CMP пишем значение,
; указывающее на наш патч (пишем код JMP 00EAD023)
      00EAD014   MOV DWORD [47F005],00A2E01A

; Возвращаем управление программе с прерванного места, послав программу по адресу
; затертым патчем
      00EAD01E   JMP 009F21F3

; Теперь вставляем нужные нам значения в регистры
; Проверяем нужно ли патчить или нет
00EAD023   CMP ESI, 005CAB4E
00EAD029   JNZ 00EAD034

; Если патчить нужно, патчим
00EAD02B   POPFD
00EAD02C   MOV ECX,1
00EAD031   CMP EAX,ECX
00EAD033   PUSHFD

; Возвращаем управление программе с прерванного места, послав программу по адресу
; затертым патчем
00EAD034   JMP 00479840
```

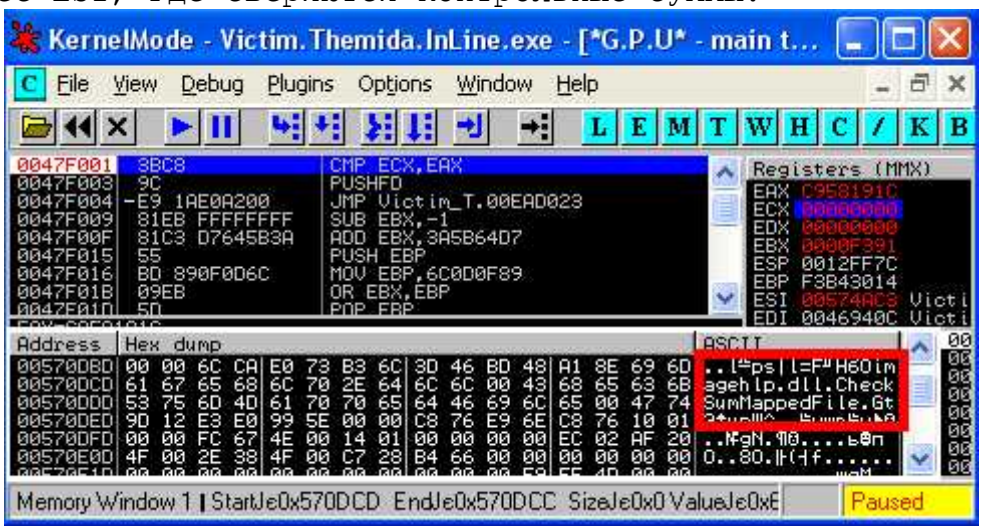
Запускаем программу и видим:

[Обход проверки изменения файла на диске]

Перезапустим программу и поставим бряк на ZwCreateFile. Немного повыполняв по F9 прогу, мы увидим в стеке название нашей программы:



Теперь делаем бряк на нашем старом знакомом CMP по адресу 47F001, и видим как в памяти разворачивается строка imagehlp.dll CheckSumMappedFile. Потом она выполняется, и после этого мы должны записать адрес ESI, где сверяются контрольные суммы:



Теперь все готово, что бы дописать наш инлайн патч:

```
; Патч CRC
; Проверяем нужно ли патчить или нет
00EAD034    CMP ESI, 00574AC3
00EAD03A    JNZ 00EAD042
```

```
; Если патчить нужно, патчим
00EAD03C   POPFD
00EAD03D   MOV ECX,EAX
00EAD03F   CMP EAX,ECX
00EAD041   PUSHFD

; Возвращаем управление программе с прерванного места, послав программу по адресу
; затертым патчем
      00EAD042   JMP 00479840
```

После этого запускаем прогу и видим:

[Exhibit]

Victim – папка с жертвой

WL – Папка с проектом WinLicense // Themida и ключом