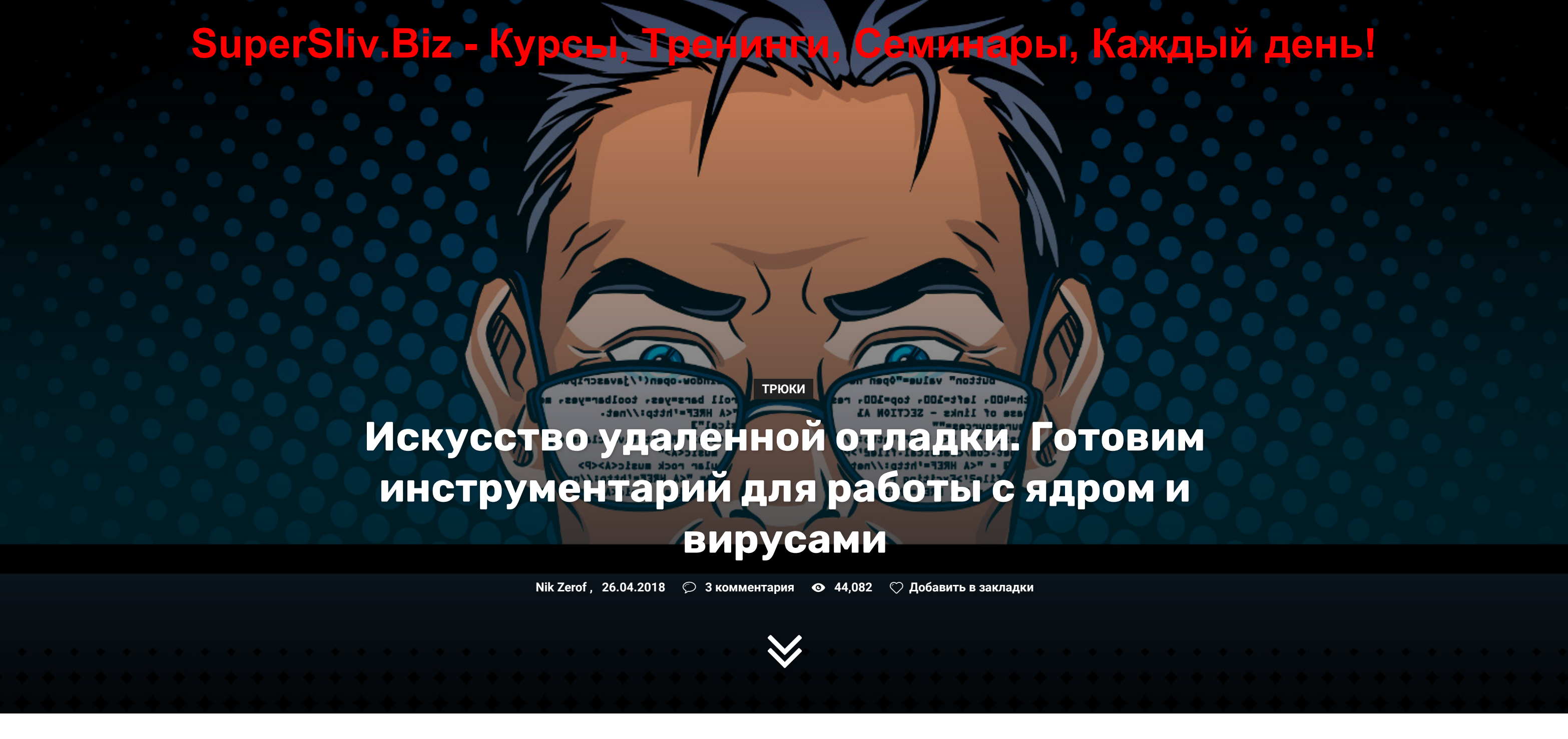


ТРЮКИ

Искусство удаленной отладки. Готовим инструментарий для работы с ядром и вирусами

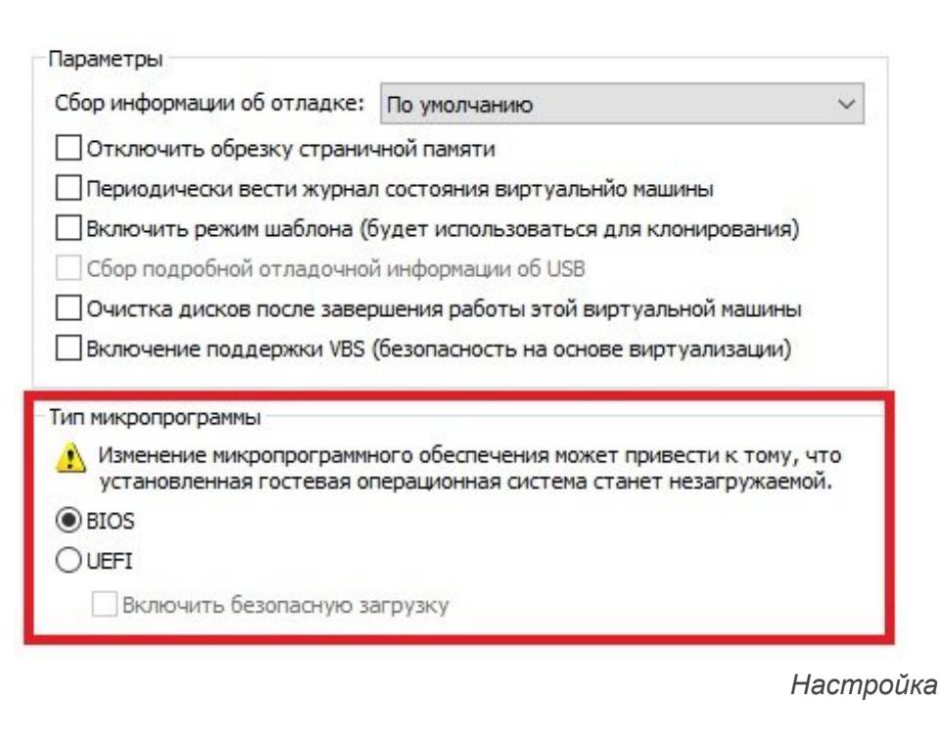
Nik Zerof , 26.04.2018 3 комментария 44,082 Добавить в закладки

Содержание статьи

Необходимость в отладке программ, запущенных внутри виртуалки, может возникнуть, когда ты пишешь компонент ядра, драйвер или же занимаешься вирусной аналитикой и не хочешь заразить основную машину. Существует несколько инструментов, которые позволяют это сделать. Настроить их с первого раза может быть непросто, так что давай посмотрим, какими они бывают и как с ними обращаться.

Установка и настройка виртуальной машины

Чтобы начать наши эксперименты, необходимо установить саму виртуальную среду. Кто-то предпочитает VirtualBox, но мы будем использовать VMware Workstation, потому что VirtualBox с некоторыми инструментами удаленной отладки дружит несколько хуже и требует дополнительной настройки. С VMware таких проблем нет. В качестве целевой операционной системы мы будем использовать Windows 10 x64 LTSB. Настоятельно рекомендуется сразу же создать общую папку для удобной переброски файлов с хостовой ОС на гостевую. Также следует выбрать тип микропрограммы BIOS, а не UEFI. Это делается для совместимости с некоторыми отладочными компонентами, которые мы будем использовать.



Параметры
Сбор информации об отладке: По умолчанию
☐ Отключить обрезку страничной памяти
☐ Периодически вести журнал состояния виртуальнйо машины
☐ Включить режим шаблона (будет использоваться для клонирования)
☐ Сбор подробной отладочной информации об USB
☐ Очистка дисков после завершения работы этой виртуальной машины
☐ Включение поддержки VBS (безопасность на основе виртуализации)
Тип микропрограммы
Изменение микропрограммного обеспечения может привести к тому, что установленная гостевая операционная система станет незагружаемой.
● BIOS
○ UEFI
☐ Включить безопасную загрузку

Настройка виртуальной машины

INFO

Во время отладки на компьютере зачастую запущено несколько требовательных к объему оперативной памяти приложений: сама виртуальная машина, компилятор, браузер. Для комфортной работы понадобится как минимум 8 Гбайт.

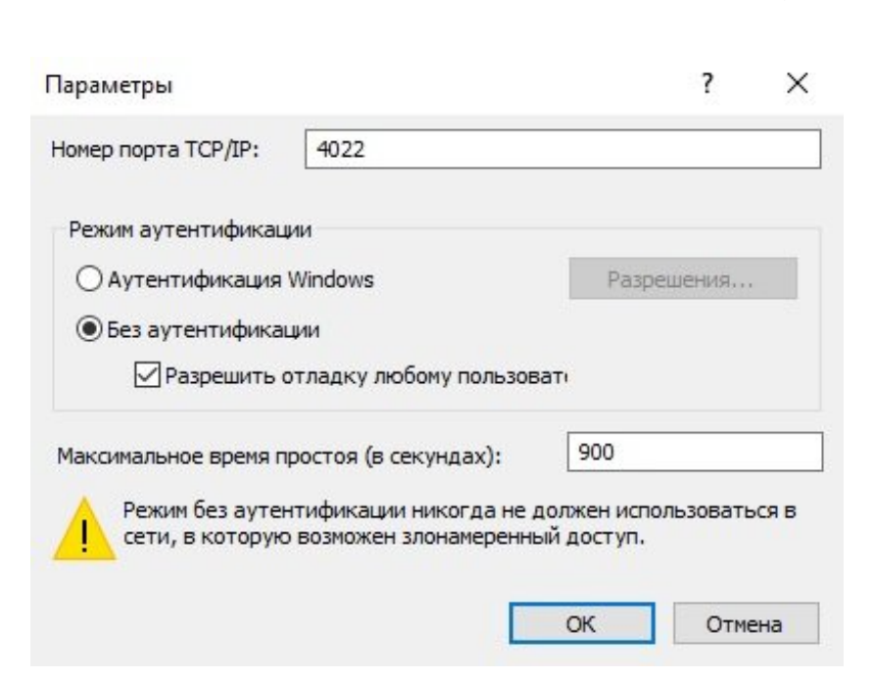
Насколько безопасно выполнять вирусную аналитику в виртуальной машине?

- ○ Совершенно безопасно. Виртуалка обеспечивает изоляцию вредоносных процессов
- ○ Потенциально опасно. Малварь может выбраться за пределы ВМ
- ○ Опасность есть, но с удаленным отладчиком наготове она минимальна

Удаленная отладка в Microsoft Visual Studio

Популярная IDE Microsoft Visual Studio также поддерживает средства удаленной отладки. Чтобы ими воспользоваться, тоже понадобится виртуальная машина, с сервером отладки Visual Studio Remote Tools. Обрати внимание: он должен подходить к той версии Visual Studio, из которой ты хочешь подключаться.

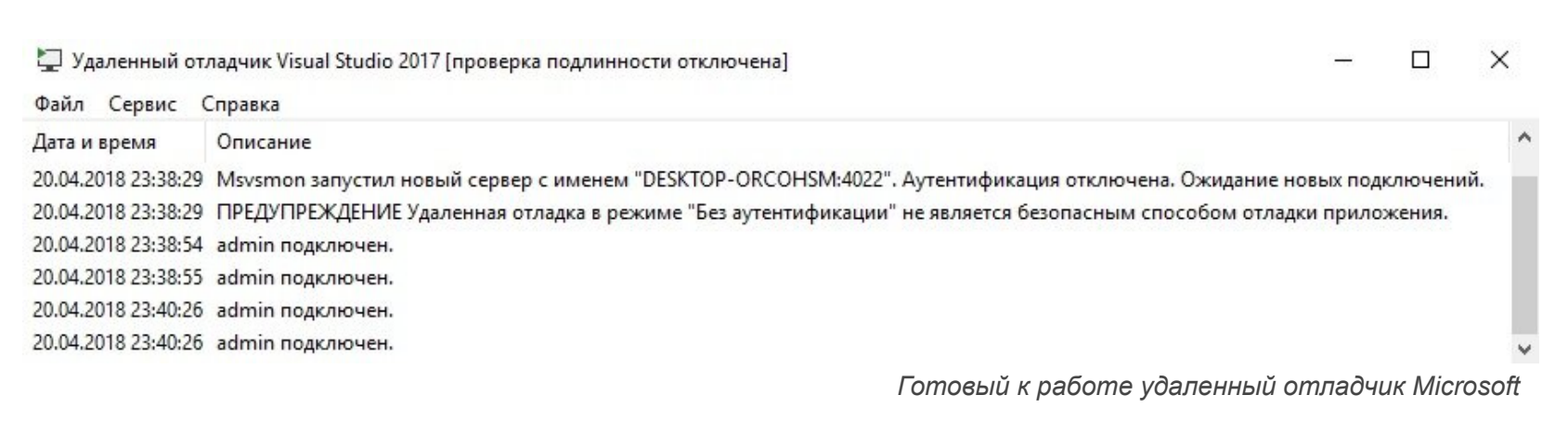
Также я рекомендую отключить аутентификацию на сервере отладки, чтобы всякий раз не вводить учетные данные пользователя. Для этого зайди в «Сервис → Параметры», выбери режим «Без аутентификации» и включи чекбокс «Разрешить отладку любому пользователю».



Параметры
Номер порта TCP/IP: 4022
Режим аутентификации
○ Аутентификация Windows
● Без аутентификации
☑ Разрешить отладку любому пользоват
Максимальное время простоя (в секундах): 900
Режим без аутентификации никогда не должен использоваться в сети, в которую возможен злонамеренный доступ.

Настройка Visual Studio Remote Tools

Сервер запущен, настроен и слушает порт 4022 на предмет подключения отладчика.

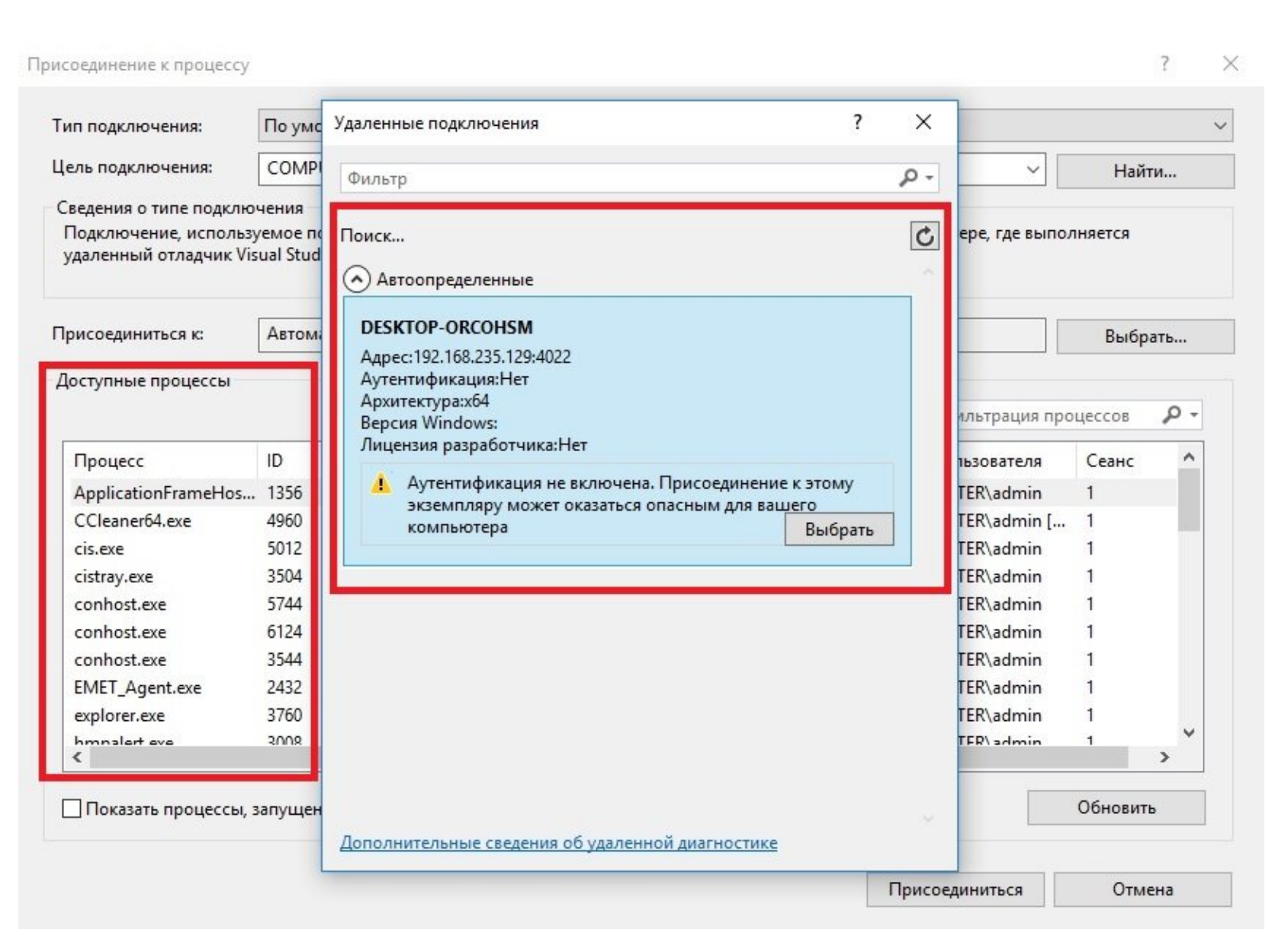


```
Удаленный отладчик Visual Studio 2017 [проверка подлинности отключена]
Файл  Сервис  Справка
Дата и время         Описание
20.04.2018 23:38:29  Msvsmon запустил новый сервер с именем "DESKTOP-ORCOHSM:4022". Аутентификация отключена. Ожидание новых подключений.
20.04.2018 23:38:29  ПРЕДУПРЕЖДЕНИЕ Удаленная отладка в режиме "Без аутентификации" не является безопасным способом отладки приложения.
20.04.2018 23:38:54  admin подключен.
20.04.2018 23:39:05  admin подключен.
20.04.2018 23:40:26  admin подключен.
20.04.2018 23:40:26  admin подключен.
```

Готовый к работе удаленный отладчик Microsoft

Теперь переходим в Visual Studio для настройки проекта под удаленную отладку. Для начала нам нужно открыть свойства проекта и перейти на вкладку «Отладка». В строке «Удаленная команда» нужно задать путь до отлаживаемого файла, в пунктах «Рабочий каталог» и «Каталог развертывания» указать каталог с отлаживаемым файлом. Также необходимо задать имя удаленного сервера (его можно взять в окне сервера отладки в виртуальной машине) и выбрать тип подключения «Без аутентификации». Теперь заходим в «Диспетчер конфигураций» и ставим чекбокс в столбце «Развертывание» напротив нашего проекта.

Теперь все готово к удаленной отладке, которую можно начать, выбрав в меню «Отладка» пункт «Начать отладку» либо нажав F5. Visual Studio, кстати, позволяет подсоединяться к удаленному процессу в виртуальной машине. Для этого нужно выбрать в меню «Отладка» «Присоединиться к процессу» и в появившемся окне — удаленный сервер и процесс.



DESKTOP-ORCOHSM
Адрес:192.168.235.129:4022
Аутентификация:Нет
Архитектура:x64
Версия Windows:
Лицензия разработчика:Нет
Аутентификация не включена. Присоединение к этому экземпляру может оказаться опасным для вашего компьютера

Процесс	ID
ApplicationFrameHos...	1356
CCleaner64.exe	4960
cis.exe	5012
cistray.exe	3504
conhost.exe	5744
conhost.exe	6124
conhost.exe	3544
EMET_Agent.exe	2432
explorer.exe	3760

Настройка Visual Studio Remote Tools

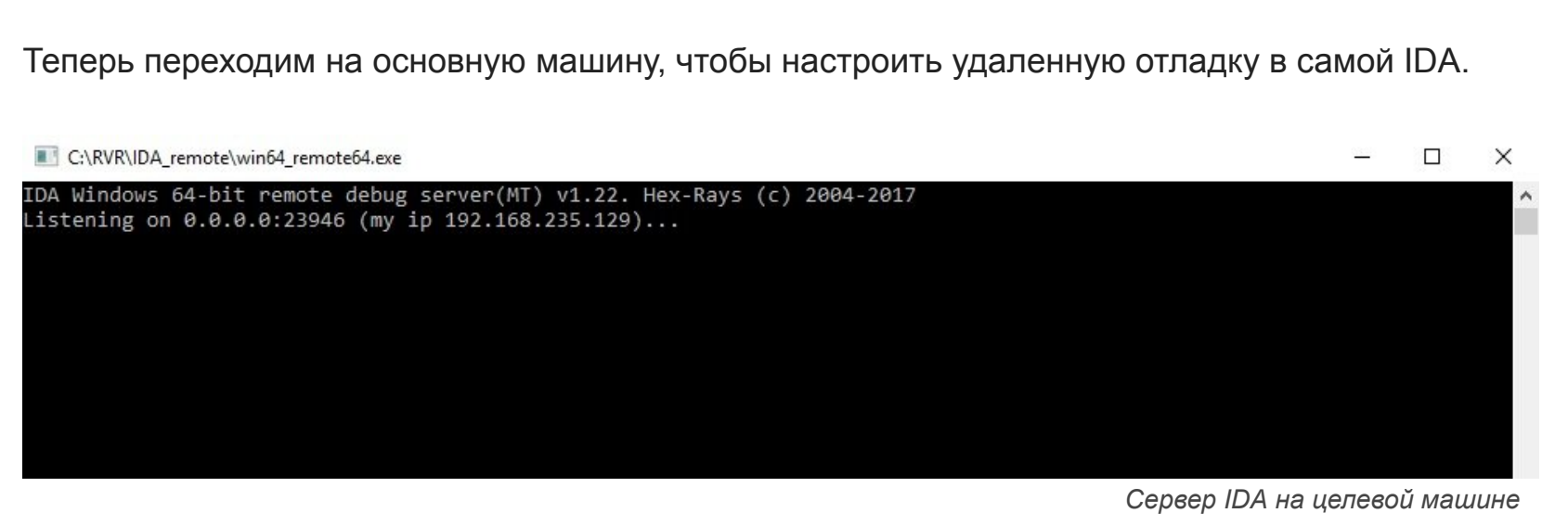
Ура! Visual Studio полностью готова к удаленной отладке.

Удаленная отладка с IDA Starter

С некоторого времени популярный и весьма мощный дизассемблер IDA Pro стал распространяться бесплатно для частного использования. Разумеется, речь идет о его урезанной версии — IDA Starter. Основная функция этого инструмента — статический анализ, но, помимо этого, он умеет удаленно отлаживать приложения. Сейчас мы разберемся, как его настроить для удаленной отладки.

Итак, в корневой папке IDA ты найдешь каталог `dbgsrv`, внутри которого есть несколько серверов под разные ОС и архитектуры процессора. Если ты собираешься отлаживать 64-разрядные приложения, то на гостевой ОС необходимо запустить файл `win64_remote64.exe`, предварительно скопировав его в виртуальную машину. После запуска он сообщит нам IP отладочного сервера и порт, через который происходит отладка. Если необходимо запустить этот сервер с возможностью авторизации, просто добавь параметр `-P` и пароль при запуске сервера отладки.

Теперь переходим на основную машину, чтобы настроить удаленную отладку в самой IDA.

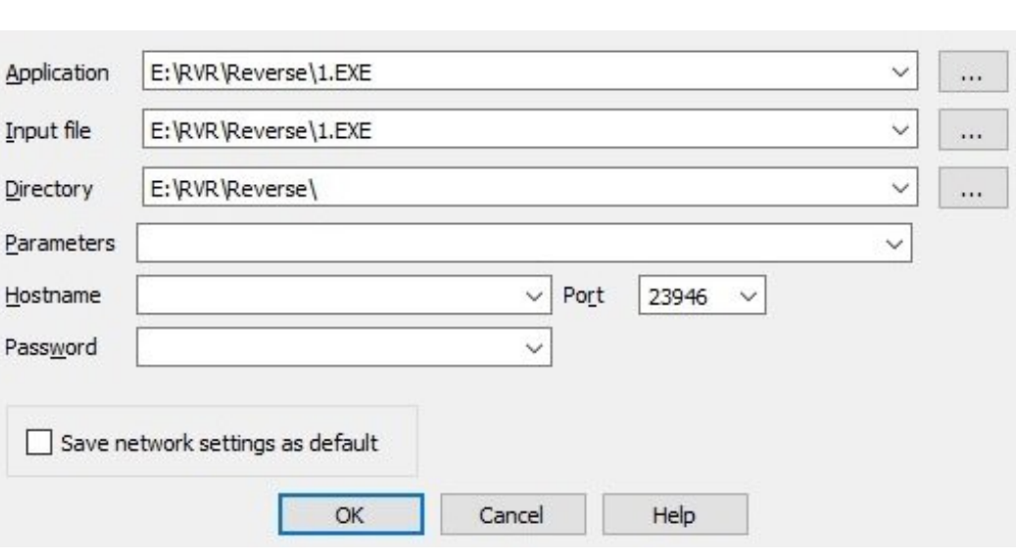


```
C:\RVR\IDA_remote\win64_remote64.exe
IDA Windows 64-bit remote debug server(MT) v1.22. Hex-Rays (c) 2004-2017
Listening on 0.0.0.0:23946 (my ip 192.168.235.129)...
```

Сервер IDA на целевой машине

Для настройки отладчика в IDA открывай меню Debugger → Select debugger или жми F9. Теперь в списке доступных отладчиков выбирай Remote Windows Debugger. Появится окно, предупреждающее о рисках дебага, отвечай утвердительно, если понимаешь их. 🙂

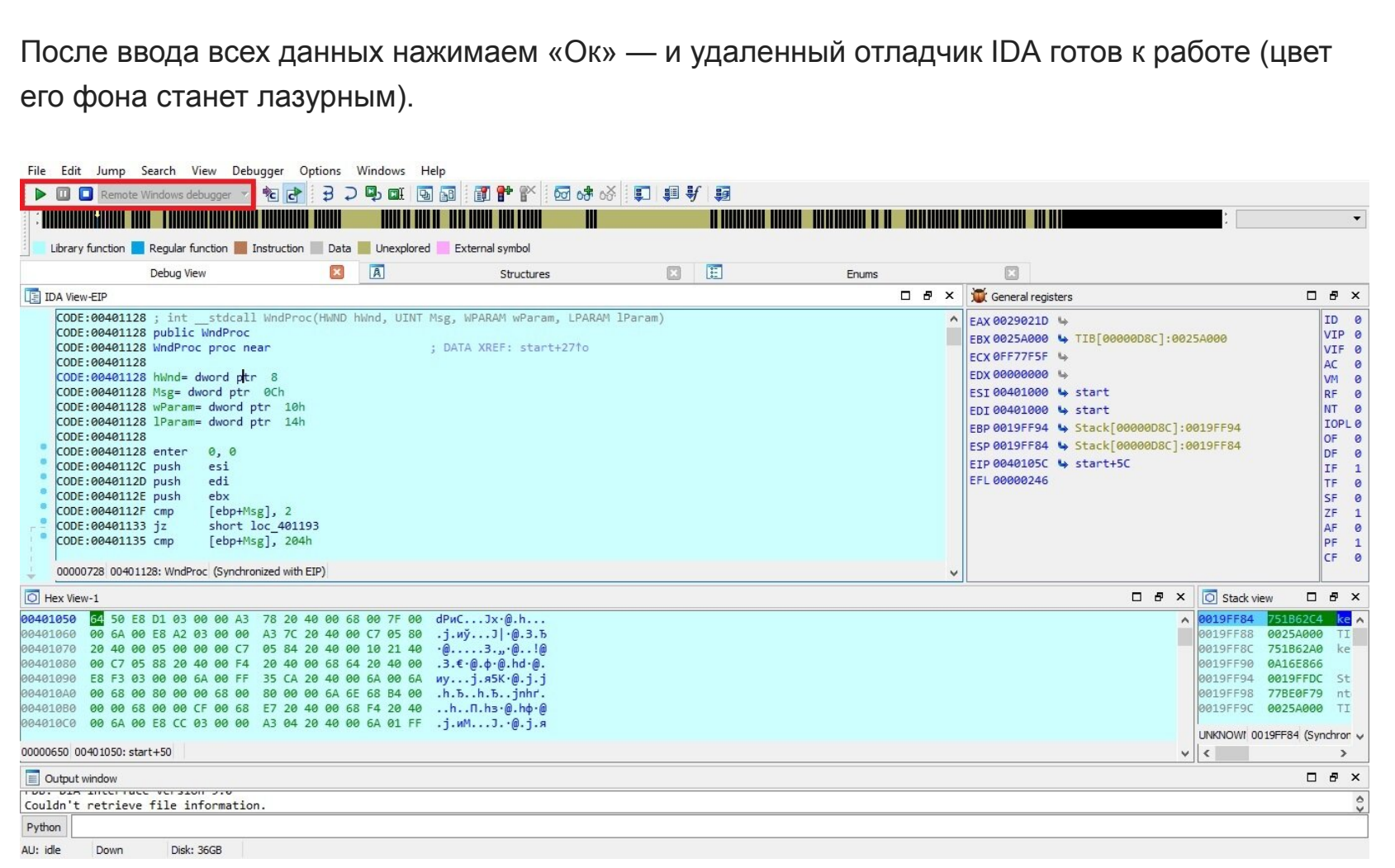
Теперь появится окно настройки удаленного отладчика, где следует задать путь к отлаживаемому файлу и рабочей папке, а также IP и порт сервера отладки на удаленной машине. Всю эту информацию можно получить в виртуальной машине, которую мы уже настроили.



Application E:\RVR\Reverse\1.EXE
Input file E:\RVR\Reverse\1.EXE
Directory E:\RVR\Reverse\
Parameters
Hostname
Port 23946
Password
☐ Save network settings as default

Настройка параметров подключения удаленного отладчика IDA

После ввода всех данных нажимаем «Ок» — и удаленный отладчик IDA готов к работе (цвет его фона станет лазурным).



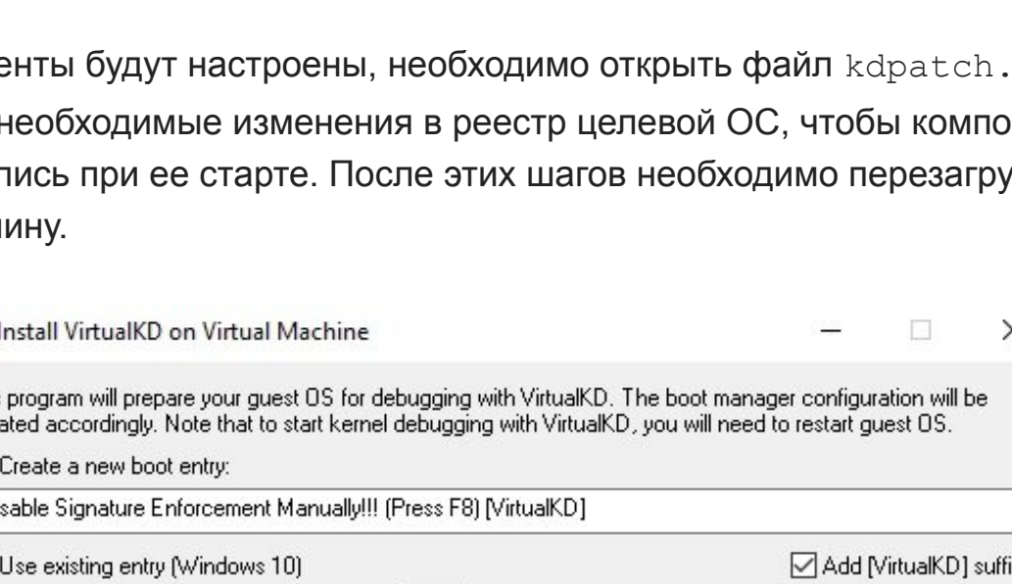
Удаленный отладчик IDA в работе

Настройка WinDbg/VirtualKD

Чтобы ускорить удаленную отладку с WinDbg, был создан специальный инструмент-надстройка — VirtualKD. Он значительно упрощает привязку WinDbg к виртуальной машине и автоматизирует некоторые настройки целевой операционки. Чтобы VirtualKD работал, в основной ОС уже должен стоять WinDbg и в нем должны быть прописаны пути для отладочных символов.

После распаковки архива с VirtualKD в его корневой папке появится папка под названием `target`. Ее необходимо скопировать на целевую ОС, после этого запустить файл `vminstall.exe`. Это основной установщик компонентов VirtualKD на целевую ОС, он копирует необходимые файлы и настраивает ОС для работы в режиме отладки. В нем необходимо включить чекбоксы Set VirtualKD boot entry as default и Replace kdcom.dll, после этого нажать кнопку Install.

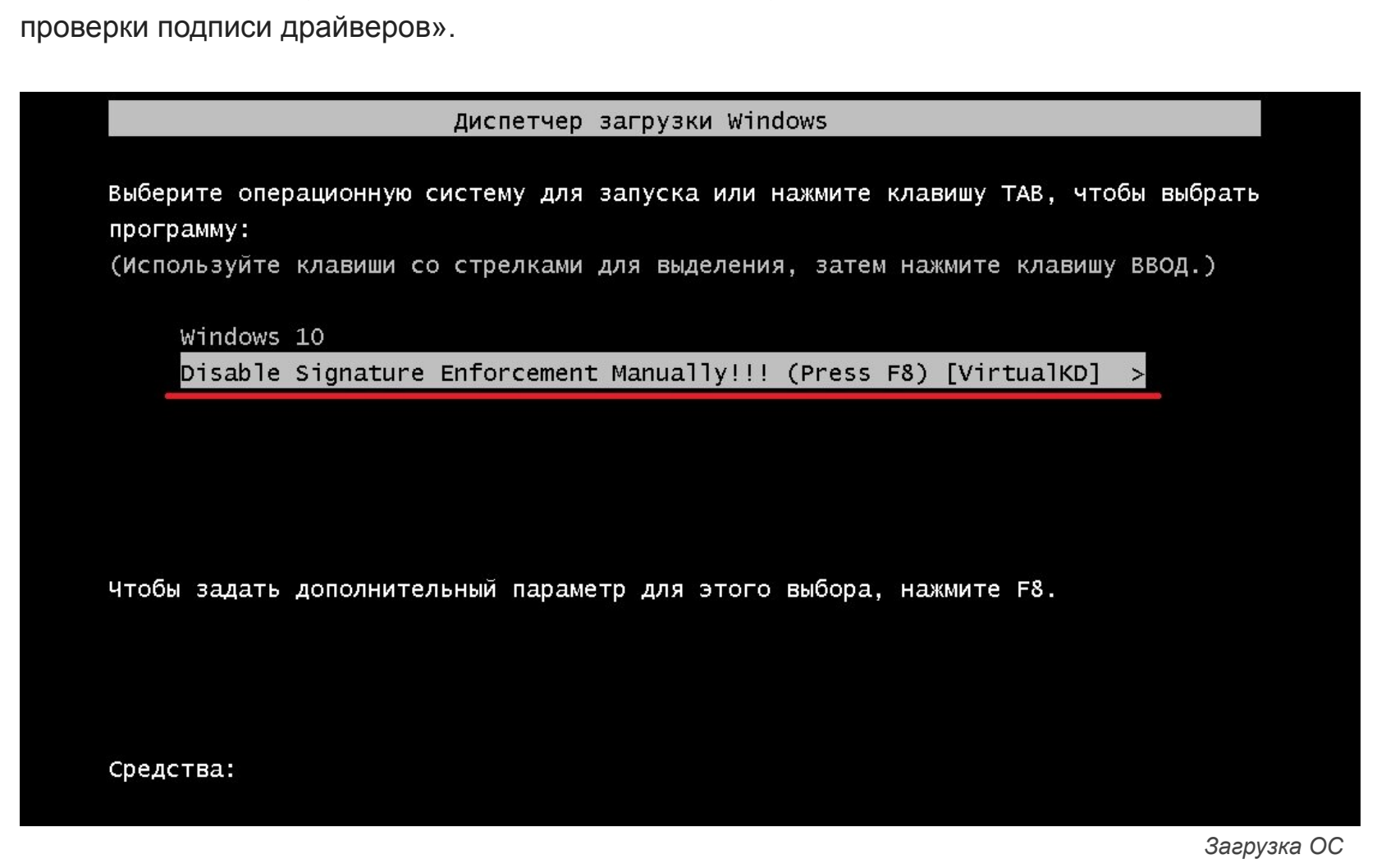
Когда все компоненты будут настроены, необходимо открыть файл `kdpatch.reg` из папки target. Он внесет необходимые изменения в реестр целевой ОС, чтобы компоненты агента VirtualKD запускались при ее старте. После этих шагов необходимо перезагрузить виртуальную машину.



Install VirtualKD on Virtual Machine
This program will prepare your guest OS for debugging with VirtualKD. The boot manager configuration will be updated accordingly. Note that to start kernel debugging with VirtualKD, you will need to restart guest OS.
● Create a new boot entry: Disable Signature Enforcement Manually!!! (Press F8) [VirtualKD]
○ Use existing entry (Windows 10)
☑ Add [VirtualKD] suffix
☑ Set VirtualKD boot entry as default with 30 second timeout.
☑ Replace kdcom.dll (required on Windows 10+)
☐ Install VisualDDK Launch Monitor to:

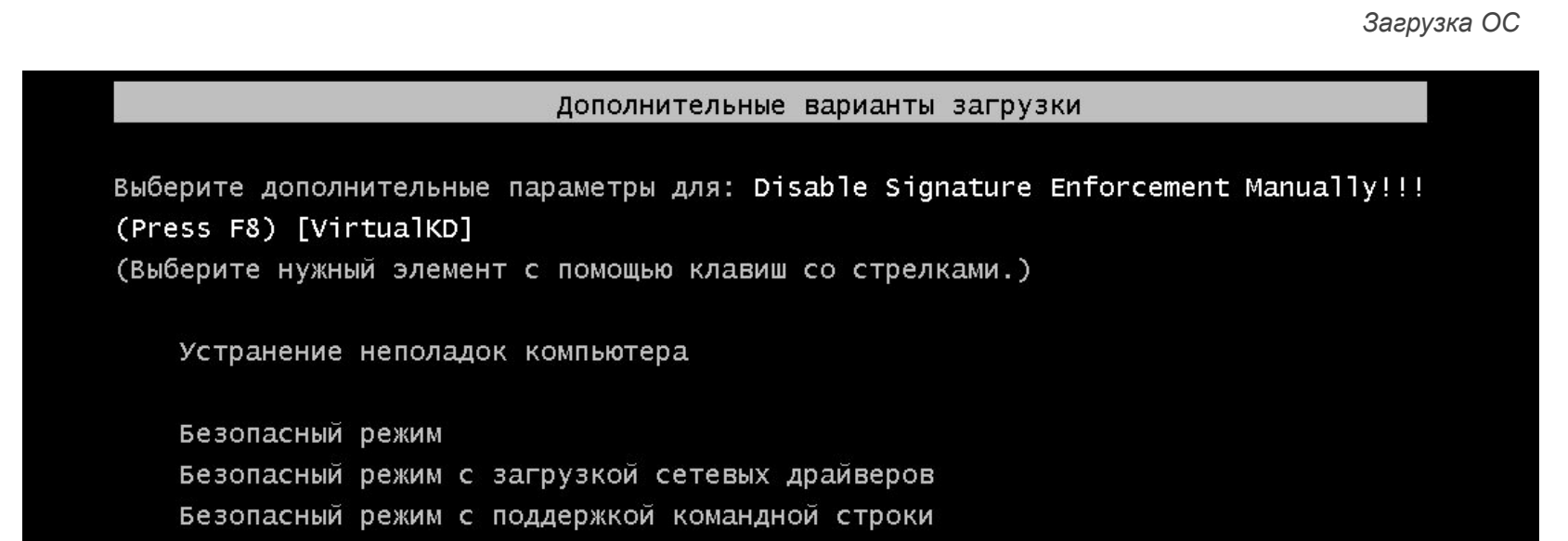
Установка VirtualKD на целевой машине

При загрузке гостевой ОС появится диалог, в котором следует выбрать Disable Signature Enforcement Manually и нажать F8. Теперь выбирай пункт «Отключение обязательной проверки подписи драйверов».



```
                    Диспетчер загрузки Windows

Выберите операционную систему для запуска или нажмите клавишу TAB, чтобы выбрать
программу:
(Используйте клавиши со стрелками для выделения, затем нажмите клавишу ВВОД.)

    Windows 10
    Disable Signature Enforcement Manually!!! (Press F8) [VirtualKD]  >


Чтобы задать дополнительный параметр для этого выбора, нажмите F8.


Средства:
```

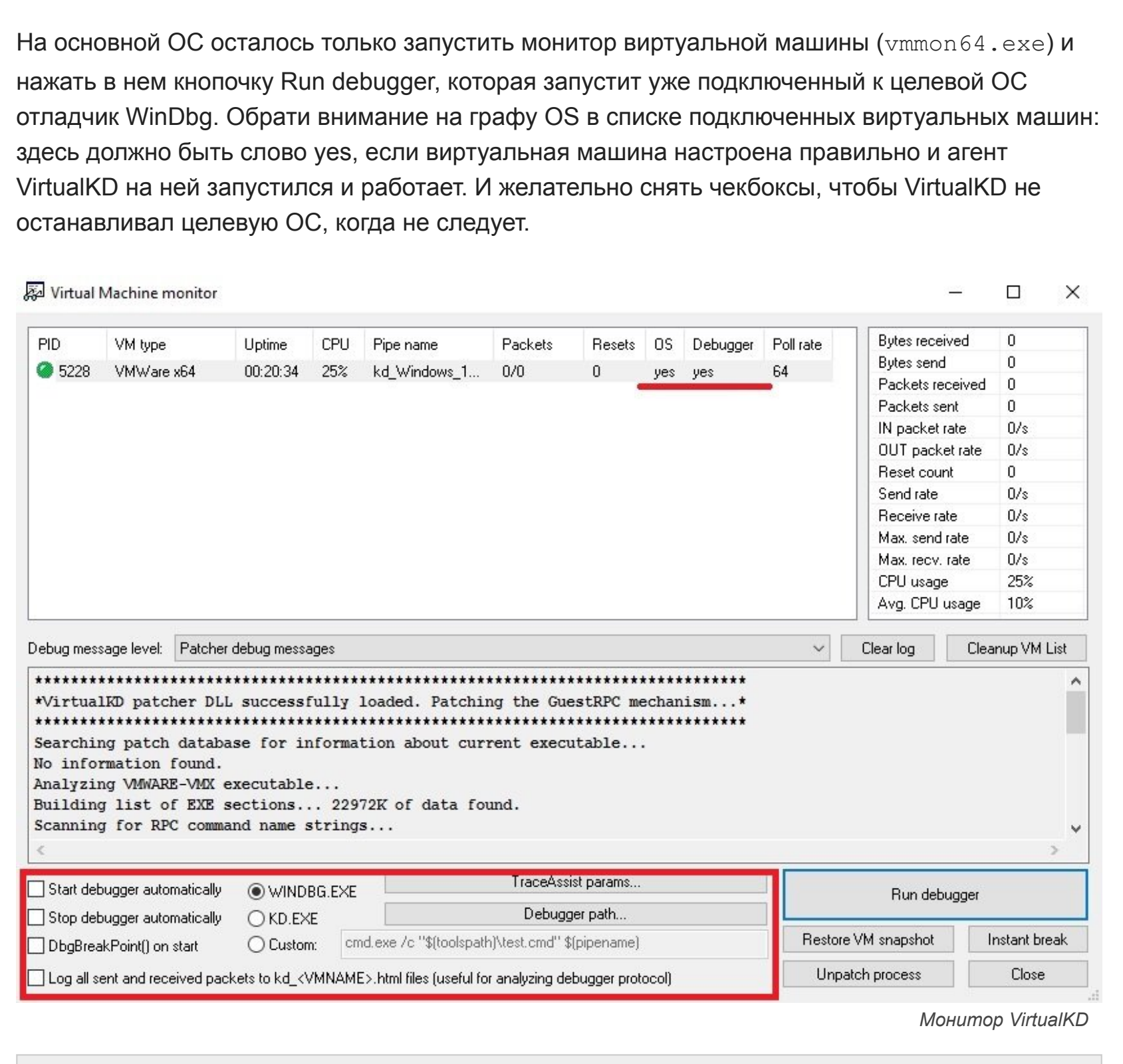
Загрузка ОС



```
                    Дополнительные варианты загрузки

Выберите дополнительные параметры для: Disable Signature Enforcement Manually!!!
(Press F8) [VirtualKD]
(Выберите нужный элемент с помощью клавиш со стрелками.)

    Устранение неполадок компьютера

    Безопасный режим
    Безопасный режим с загрузкой сетевых драйверов
    Безопасный режим с поддержкой командной строки

    Ведение журнала загрузки
    Включение видеорежима с низким разрешением
    Режим отладки
    Отключить автоматическую перезагрузку при отказе системы
    Отключение обязательной проверки подписи драйверов
    Отключение раннего запуска антивредоносного драйвера

    Обычная загрузка Windows
```

Отключение обязательной проверки подписи драйверов

На основной ОС осталось только запустить монитор виртуальной машины (`vmmon64.exe`) и нажать в нем кнопочку Run debugger, которая запустит уже подключенный к целевой ОС отладчик WinDbg. Обрати внимание на графу OS в списке подключенных виртуальных машин: здесь должно быть слово yes, если виртуальная машина настроена правильно и агент VirtualKD на ней запустился и работает. И желательно снять чекбоксы, чтобы VirtualKD не останавливал целевую ОС, когда не следует.



PID	VM type	Uptime	CPU	Pipe name	Packets	Resets	OS	Debugger	Poll rate
5228	VMWare x64	00:20:34	25%	kd_Windows_1...	0/0	0	yes	yes	64

Bytes received	0
Bytes send	0
Packets received	0
Packets sent	0
IN packet rate	0/s
OUT packet rate	0/s
Reset count	0
Send rate	0/s
Receive rate	0/s
Max. send rate	0/s
Max. recv. rate	0/s
CPU usage	25%
Avg. CPU usage	10%

Debug message level: Patcher debug messages

```
**********************************************************************
*VirtualKD patcher DLL successfully loaded. Patching the GuestRPC mechanism...*
**********************************************************************
Searching patch database for information about current executable...
No information found.
Analyzing VMWARE-VMX executable...
Building list of EXE sections... 22972K of data found.
Scanning for RPC command name strings...
```

☐ Start debugger automatically ● WINDBG.EXE
☐ Stop debugger automatically ○ KD.EXE
☐ DbgBreakPoint() on start ○ Custom: cmd.exe /c "$(toolspath)\test.cmd" $(pipename)
☐ Log all sent and received packets to kd_<VMNAME>.html files (useful for analyzing debugger protocol)

Монитор VirtualKD

Что может помешать удаленной отладке исполняемого файла?

- ○ Детект запуска в виртуалке
- ○ Использование UEFI
- ○ Процессор без поддержки аппаратной виртуализации
- ○ Ответы 1 и 2
- ○ Ответы 2 и 3

Анализ аварийного дампа при помощи WinDbg

Итак, инструментарий для удаленной отладки на все случаи жизни у нас подготовлен и настроен. Но если с IDA Pro или Visual Studio ты без труда сможешь начать работать, то с отладкой в режиме ядра могут возникнуть определенные сложности. Давай посмотрим, что делать, если запуск драйвера обернулся BSOD и нам надо в этом разобраться.

Предположим, что мы запустили виртуальную машину, загрузили в нее наш драйвер и не забыли включить отладчик через монитор VirtualKD. Посылаем драйверу запрос IOCTL, чтобы он выполнил нужный блок кода, который и привел к BSOD. После этого виртуальная машина остановилась, и отладчик сообщил о сбое, предложив исполнить команду `!analyze -v`. Выполняем ее, и WinDbg показывает анализ дампа сбоя, из которого мы будем получать основную информацию. Давай разберем самые значимые для нас поля.

Где искать дампы

Если мы отлаживаем собственный драйвер, то знаем, когда может произойти сбой, и держим наготове отладчик, подключенный к виртуальной машине. Но если ОС падает в BSOD неожиданно, нам требуется вручную загрузить файл аварийного дампа в отладчик, чтобы понять, что вызвало сбой. Windows создает файл полного аварийного дампа автоматически и называет его `MEMORY.DMP`. Он располагается в корневой папке ОС, а мини-дамп ты найдешь в папке minidump.

Приглашение WinDbg начинается с баннера Bugcheck Analysis и содержит ссылку на команду `!analyze -v`, которая загрузит крешдамп в отладчик и разберет его. Выполняем команду и видим примерно следующее (крешдамп большой, я покажу его начало).

```
SYSTEM_SERVICE_EXCEPTION (3b)
An exception happened while executing a system service routine.
Arguments:
Arg1: 00000000c0000005, Exception code that caused the bugcheck
Arg2: fffff801033e1722, Address of the instruction which caused the bugcheck
Arg3: ffff890143778d20, Address of the context record for the exception that caused the
Arg4: 0000000000000000, zero.

Debugging Details:
------------------
DUMP_CLASS: 1
DUMP_QUALIFIER: 0
BUILD_VERSION_STRING:  14393.2068.amd64fre.rs1_release.180209-1727
DUMP_TYPE:  0
BUGCHECK_P1: c0000005
BUGCHECK_P2: fffff801033e1722
BUGCHECK_P3: ffff890143778d20
BUGCHECK_P4: 0
EXCEPTION_CODE: (NTSTATUS) 0xc0000005
...
```

Крешдамп сообщает об ошибке `SYSTEM_SERVICE_EXCEPTION (3b)`. MSDN говорит, что она происходит из-за исключения, вызванного при переходе из непривилегированного кода (режим пользователя) в привилегированный (режим ядра). В глаза бросается строчка `EXCEPTION_CODE`, которая показывает ошибку `NTSTATUS`, о ней ты можешь прочесть по ссылке. Получается, произошло нарушение доступа, `STATUS_ACCESS_VIOLATION`. Читаем дамп дальше.

```
CONTEXT:  ffff890143778d20 -- (.cxr 0xffff890143778d20)
rax=0000000000000000 rbx=ffffb104c7a5c850 rcx=0000000000000000
rdx=0000000040000010 rsi=0000000000000001 rdi=ffffb104c6ed0ab0
rip=fffff801033e1722 rsp=ffff890143779720 rbp=0000000000000002
r8=ffffb104c82daa0c  r9=ffff8901437797d8 r10=0000000000000000
r11=ffff890143779758 r12=0000000000000000 r13=ffffb104c6b2b970
r14=ffffb104c7a5c850 r15=0000000000000000
```

Здесь мы видим состояние регистров процессора на момент сбоя.

```
DEFAULT_BUCKET_ID:  WIN8_DRIVER_FAULT
BUGCHECK_STR:  0x3B
PROCESS_NAME:  loader.exe
```

Поле `DEFAULT_BUCKET_ID` говорит нам о категории сбоя, `BUGCHECK_STR` показывает сокращенный код сбоя, `PROCESS_NAME` указывает на название процесса, к которому был подключен процессор на момент BSOD. В данном случае процесс `loader.exe` послал запрос IOCTL к драйверу, который и обрушил систему.

```
FOLLOWUP_IP:
testdrv!MmGetSystemAddressForMdlSafe+12
fffff801`033e1722 0fbf400a        movsx   eax,word ptr [rax+0Ah]
```

В этой части отладчик указывает на конкретное место сбоя в формате `[модуль]![функция+смещение]`. Здесь модуль называется `testdrv`, функция `MmGetSystemAddressForMdlSafe` и смещение 12. Далее идет ассемблерный код места сбоя.

```
MODULE_NAME: testdrv
IMAGE_NAME:  testdrv.sys
```

Здесь `IMAGE_NAME` указывает на название файла, а `MODULE_NAME` — на название объекта.

Мы рассмотрели основные поля аварийного дампа, но не все, потому что некоторые из них дублируются, другие предоставляют избыточную информацию. В любом случае, изучив вывод команды `!analyze -v`, ты соберешь немало информации. Конечно, это не все: поиск ошибок в драйверах отнимает немало сил и времени даже у опытных системных программистов, но это хороший старт в отладке и трассировке системного кода.

Каким отладчиком ты уже пользовался?

- ○ IDA
- ○ WinDbg
- ○ Встроенным в MS Visual Studio
- ○ OllyDbg или Immunity Debugger
- ○ Другим (укажи в комментариях)

Заключение

Мы рассмотрели основной инструментарий удаленной отладки, настроили виртуальную машину и попытались разобрать аварийный дамп операционной системы. На самом деле тема отладки приложений весьма обширна, и сложно уместить все в одной статье, но задать нужное направление для самостоятельного изучения можно. Надеюсь, что этот материал поможет тебе в работе.