3. Загрузка ОС по iSCSI

Чтобы запустить Windows требуется вычитать загрузчик из MBR сектора и передать ему управление. Главное отличие от “оффлайн” запуска заключается в том, что вычитывать MBR сектор мы будем с удаленного диска по iSCSI протоколу.

В такой постановке у нас есть сервер, раздающий диск с предустановленной ОС и клиент, который подключается к серверу, получает доступ к нужному SAN диску и вычитывает загрузчик.

В роли сервера мы будем использовать **Tgtd**, выступающего в роли iSCSI таргета.

Клиентом же является специальный загрузчик ОС, **iPXE**, который способен выступать в роли iSCSI инициатора.

Ядру операционной системы, после передачи ему управления, потребуется повторно установить соединение с iSCSI таргетом для того, чтобы примонтировать диск, с которого он был загружен. Для этого требуется предварительно сохранить информацию про iSCSI соединение, которое устанавливал iPXE.
Эта информация хранится внутри специальной ACPI таблицы - **iBFT**, которую запишет iPXE в системную память перед тем, как прочитать MBR с iSCSI диска.

iBFT содержит в себе:
информацию про инициатора, таргета и сетевую карту(MAC, Bus Device Function)

В дальнейшем в подразумевается, что мы используем Centos для настройки и работы с Tgtd, iPXE.

4. Установка и настройка Tgtd

Установка:
sudo yum install -y epel-release
sudo yum install -y scsi-target-utils

Настройка:
Допустим у нас есть блочное устройство /dev/sda, которое мы собираемся раздавать по iSCSI, для этого требуется описать новый таргет и LUN в tgtd. Конфигурационные файлы tgtd находятся в директории /etc/tgt/, главным является targets.conf в котором указаны директивы include, подключающие остальные конфиги, к примеру,
/etc/tgt/conf.d/sample.conf, который мы и будем править.

в iSCSI вместо SCSI ID используются **IQN** (iSCSI qualified name)
формата:
iqn.{гггг-мм}.{инвертированное имя домена}:{имя сервера}

внутри sample.conf опишем новый таргет с IQN
iqn.2021-02.org.example:storage1-sys.prof

<target iqn.2021-02.org.example:storage1-sys.prof>

backing-store /dev/sda

</target>

backing-store /dev/sda - описание раздела, который мы будем раздавать. В данном случае, мы описываем LUN0.

запуск tgtd:

sudo systemctl start tgtd

sudo systemctl enable tgtd

5. Установка и настройка iPXE

Мы будем скачивать iPXE и добавлять новую запись в Grub

Скачиваем загрузчик

wget http://boot.ipxe.org/ipxe.lkrn -O /boot/ipxe.lkrn

Добавляем скрипт, который исполнится после загрузки iPXE

cat << EOF >> /boot/ipxe.conf

#!ipxe

dhcp

sanboot iscsi:192.168.0.108:::0:iqn.2021-02.org.example:storage1-sys.prof

EOF

dhcp - автоматическая настройка сетевых адаптеров через dhcp.

sanboot - запуск ОС с iscsi диска.

Здесь *192.168.0.108* - ip адрес таргета.

0 - номер LUN,

iqn.2021-02.org.example:storage1-sys.prof - IQN имя таргета.

В приведённом примере мы получим доступ и запустим загрузчик с /dev/sda, описанного в предыдущем разделе о настройке tgtd.
Добавляем новую опцию в Grub:

```
cat << EOF >> /etc/grub.d/40_custom
menuentry "iPXE" {
    linux16 /boot/ipxe.lkrn
    initrd16 /boot/ipxe.conf
}
EOF
```

6. Описание успешного запуска Windows, начиная с iPXE до передачи управления ядру ОС.

В этом разделе в деталях описывается успешный запуск Windows с момента запуска загрузчика iPXE до передачи управления ядру ОС. Это описание призвано упростить понимание проблемы, к которой мы придём в последующих разделах.

В данной работе ряд комментариев по внутренней структуре реестра и ядра Windows будут даны исходя из изучения утекших в сеть исходных кодов.

Комментарии по коду iPXE даны на момент коммита
02280dc642907b908f4b5c7e0d82d8ad1d51d574
В официальном репозитории: https://github.com/ipxe/ipxe

I. Мы загрузились в iPXE, запустилась команда
> sanboot iscsi:192.168.0.108:::0:iqn.2021-02.org.example:storage1-sys.prof

внутри iPXE происходит следующий процесс:
Мы переходим в uriboot, в котором происходит процесс запуска загрузчика ОС:
1. вызывается san_hook(...) ->int13_hook(...)
В нем регистрируется информация о блочном устройстве, устанавливается соединение с san диском, переписывается обработчик **INT13**, программного прерывания BIOS для доступа к жесткому диску. Таким образом, будет перехватываться чтение san диска загрузчиком ОС.

2. вызывается san_describe(..)->int13_describe(..)->acpi_install(int13_install)
В нём заполняется новая ACPI таблица и помещается в системную память.
В нашем случае, эта таблица - **iBFT**.

3. вызывается san_boot(...) -> int13_boot(...)
в котором мы непосредственно вычитываем MBR и передаем управление загрузчику Windows.

II. Загрузчик Windows при помощи INT13 читает загрузочный диск, ищет BootMgr и передаёт ему управление.

III. BootMgr:
Нас интересует часть кода в *minkernel\boot\environ\app\osloader\osloader.c*
В OslpLoadAllModules(...)

1. Считываются с загрузочного диска все необходимые модули и Boot Critical драйвера в память.

2. Вызывается OslpGetBootDriverFlags(...), в которой будет найдена **iBFT** и выставлен флаг CM_SERVICE_NETWORK_BOOT_LOAD, благодаря которому в дальнейшем будут загружены драйвера для сетевой карты, описанной в iBFT.

3. Передается управление ядру Windows.

IV. Происходит перечисление PCI устройств для PnPManager. Драйвера создают логические устройства. Формируются структуры, которые в дальнейшем будут добавлены в реестр.

V. Остаток загрузки

7. Воспроизведение BSOD

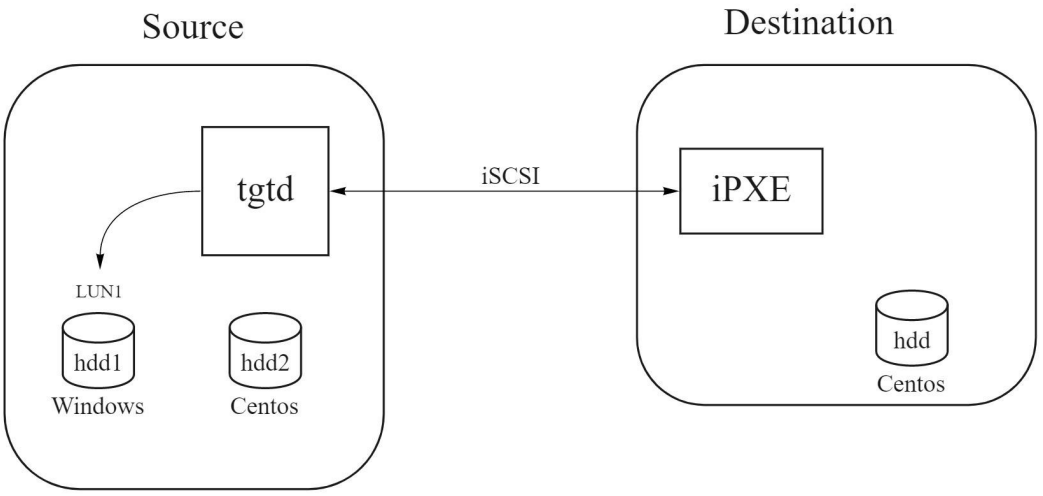

Рис. 1 (Окружение)

VMware ESXi 6.7.0

Две виртуальные машины (Source,Destination)

Source:

сетевая карта: e1000

ip: 192.168.0.108

2 hdd

1. Проинсталлируем Windows 2016 на hdd1

2. Запустим Windows

При запуске ОС произойдёт перечисление PCI устройств VM Source, в том

числе сетевой карты e1000. Эта информация сохранится в реестре.
Таким образом, Windows будет “знать” про подключенную сетевую карту e1000.
Больше загружаться с hdd1 мы не станем. В дальнейшем мы будем только раздавать этот диск по iSCSI.
3. Проинсталлируем Centos на hdd2 и сделаем hdd2 загрузочным.
4. Добавим и настроим tgtd. Опишем hdd1 как LUN1 в конфиг файле tgt.

Destination:
сетевая карта: e1000
1 hdd

1. Проинсталлируем Centos на hdd
2. Скачиваем iPXE, создаём конфиг в котором добавим запуск ОС с hdd1 Source VM.
3. Добавляем опцию в Grub с загрузкой iPXE.
В дальнейшем будем использовать только запуск iPXE, а Centos трогать не будем.

Если сейчас запустить iPXE, Windows успешно запустится с iSCSI диска.

Теперь поменяем сетевой адаптер на Destination VM с e1000 на e1000e и вновь попробуем запустить Windows с iSCSI диска. Мы столкнемся со следующим поведением:
1. Появление логотипа Windows на несколько минут.
2. BSOD (inaccessible boot device).

8. Исправление BSOD

Проблема, которую мы решаем, формулируется следующим образом:
Какие минимальные изменения на Source VM требуется совершить для успешного запуска Windows с iSCSI диска ?

BSOD “Inaccessible Boot Device” возникает тогда, когда мы меняем сетевую карту у *Destination*. Для нас важно то, что новая сетевая карта ни разу не встречалась в Windows при перечислении PCI устройств.

В контексте запуска по iSCSI ОС считает, что две сетевые карты совпадают, если:

1. У них совпадает модель.
2. У них совпадает BDF (Bus Device Function)
3. Они подключены к одному и тому же родительскому PCI устройству, у которого совпадает BDF.

Таким образом, если у двух сетевых карт совпадают модель, BDF, но они подключены к разным слотам - с точки зрения ОС это две разные карты.

Это определение является ключевым для понимания того, как исправить BSOD. После ряда модификаций реестра была выявлена следующая закономерность: Windows успешно запускается с iSCSI диска при соблюдении двух условий:

1. В Windows присутствует драйвер новой сетевой карты *Destination*.
2. Сетевая карта у *Destination* должна совпадать с сетевой картой, описанной ранее при перечислении PCI устройств. Таким образом, на момент запуска по iSCSI в реестре должна присутствовать вся информация, заполненная драйверами во время предыдущих успешных запусков ОС с локального диска.

Ниже будет продемонстрировано, какие действия нужно совершить для успешного запуска по iSCSI с новой сетевой картой e1000e. Все дальнейшие модификации Windows будут происходить из-под запущенной Centos на *Source*. Для доступа к файловой системе, мы примонтируем диск с Windows. Модификация SYSTEM hivex файла будет происходить при помощи библиотеки libhivex.

Нам потребуется добавить в реестр всю необходимую информацию по e1000e, для этого удобно создать еще одну тестовую виртуальную машину, добавить на неё единственную сетевую карту e1000e, проинсталлировать Windows Server 2019, и копировать все нужные параметры из её реестра.

Эта сетевая карта будет совпадать с картой на *Destination*. Связано это с тем, что BDF,Slot,Родительское PCI устройство сетевой карты ESXi VM детерминировано задаются исходя из порядка добавления карты. Таким образом, если создать две разные виртуальные машины, добавить в них одну и ту же первую сетевую карту- они будут совпадать.
Отличия на уровне MAC адресов, как уже было упомянуто выше, Windows интересовать не будет.

9. Описание новой сетевой карты в реестре

Ниже будут кратко описаны ветки реестра, которые обязательны к заполнению для устранения BSOD. В дальнейшем, мы будем брать информацию для заполнения этих веток реестра из inf-файла драйвера и тестовой ВМ.

Несмотря на то, что большая часть информации будет шаблонной и скопирована в реестр “как-есть”, ряд ключей, именуемых далее runtime-данными, потребуется заполнить самостоятельно в момент их добавления. Эти runtime-данные можно поделить на 2 типа:

1. Сгенерированные случайным образом, например - GUID
2. Зависящие от сетевой карты на *Destination*, к примеру - BDF

ControlSet001\Services

В данном разделе описываются все существующие в ОС сервисы.

Windows Service - асинхронная программа, схожая по своей концепции с Unix демонами.

В Windows за каждым драйвером закрепляется свой отдельный сервис, в связи с этим, После добавления нового драйвера на файловую систему, потребуется описать в этом разделе новый сервис, который будет регулировать его работу.

ControlSet001\ENUM\PCI

Новая сетевая карта на *Destination* - это подключенное PCI устройство, которое должно быть описано в этой ветке.

ControlSet001\Class

Для упрощения установки, конфигурации, и инициализации в Windows все схожие устройства сгруппированы в специальные классы. Каждый такой класс имеет свой уникальный GUID. Например, новая сетевая карта относится к классу “Network Adapter” с GUID *4d36e972-e325-11ce-bfc1-08002be10318.*

В данном разделе описывается информация про то, как инициализировать описанное ранее PCI устройство. Например, отвечающий за это устройство драйвер.

ControlSet001\Control\NetworkSetup2\Interfaces

В данном разделе описываются сетевые интерфейсы - “точки соединения” между компьютером и Ethernet, Wi-fi и пр., через которые будут отправляться данные.

С точки зрения ОС, сетевой интерфейс можно рассматривать как логическую сетевую карту. Сетевой интерфейс сопоставляется либо с реальным физическим устройством - сетевой картой, либо с программным, как например в случае с IPv4 “loopback” интерфейсом 127.0.0.1 или его аналогом :::1 для IPv6.

Эта точка соединения описывается своим ip адресом и используемыми протоколами(IPv4, IPv6, Tcpip и др)

SlotPersistentInfo

Для сетевых карт VMware, которые подключены к VMware Root Express Port, требуется явно описать связь между сетевой картой и устройством, к которому она подключена. Эта информация будет описана в параметре *SlotPersistentInfo*. Полное название соответствующего ключа будет дано в разделе, посвященному этому параметру.

Многие runtime-данные ссылаются друг на друга. Ниже можно наглядно увидеть эту связь.

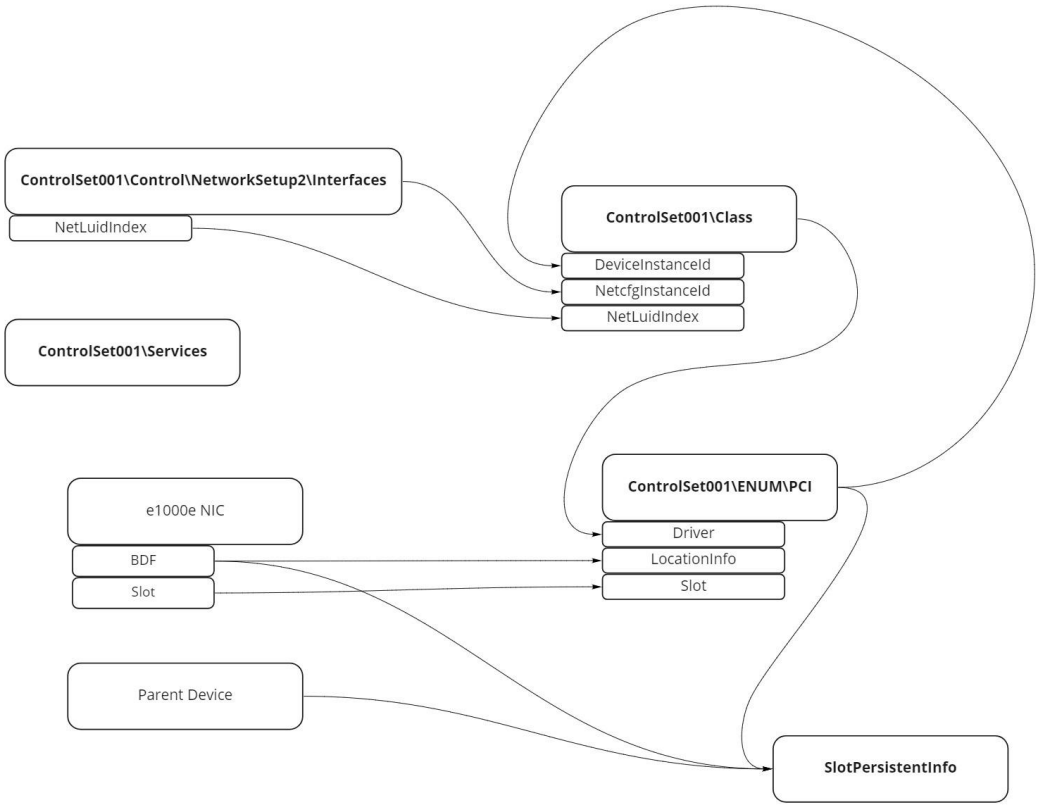

Рис.2 (Взаимосвязь runtime значений)

9.1 INF файл

Часть информации, которая будет добавлена в реестр, будет взята из inf-файла драйвера сетевой карты, поэтому имеет смысл остановиться на нём подробнее и описать, что в нём находится.

inf-файл содержит в себе набор секций и директив, которые будут исполнены во время установки драйвера или при добавлении нового устройства, которым может управлять этот драйвер.

Для каждого устройства описывается своя собственная секция, именуемая дальше *DDInstall*, с которой начинается процесс установки драйвера. Эта секция содержит в себе директивы для инсталляции файлов драйвера, добавления информации про новое устройство и\или про сам драйвер в реестр.

Сопоставление названия *DDInstall* секции и устройства происходит при помощи *DeviceID,* уникального идентификатора модели PCI устройства, который используется Windows PnP менеджером.

DeviceID имеет разные форматы, нас же будут интересовать следующие:

1. *PCI\\VEN_v(4)&DEV_d(4)*
2. *PCI\\VEN_v(4)&DEV_d(4)&SUBSYS_s(4)n(4)&REV_r(2)*

Здесь (N) - N разрядное hex число.

v(4) - PCI SIG ID вендора устройства,

d(4) - ID устройства, специфичный для вендора,

s(4) - ID подсистемы, специфичный для вендора,

n(4) - PCI SIG ID вендора подсистемы,

r(2) - номер ревизии,

s(2) - код сабкласса,

примеры таких *DeviceID* для *e1000e*:

PCI\VEN_8086&DEV_10D3

PCI\VEN_8086&DEV_10D3&SUBSYS_07D015AD&REV_00

Найдём на примере e1000e нужные нам секции:

внутри Inf-файла находим

```
[Intel.NTamd64.6.2]
; DisplayName              Section      DeviceID
; -----------              -------      --------
%E10DENC.DeviceDesc%       = E10DE,     PCI\VEN_8086&DEV_10DE
%E10DENC.DeviceDesc%       = E10DE,     PCI\VEN_8086&DEV_10DE&SUBSYS_10DE8086
```

Для устройства с DeviceID *PCI\VEN_8086&DEV_10DE*(нашей карты) название секции *DDInstall* - *E10DE*

Таким образом, будут исполняться директивы, описанные в секциях [*E10DE], [E10DE.Services], [E10DE.HW]*

```
;-------------------------------------------------------------------------
; Intel(R) 82567LM-3 Gigabit Network Connection
;
[E10DE]
Characteristics    = 0x84 ; NCF_HAS_UI | NCF_PHYSICAL
BusType            = 5    ; PCI
AddReg             = e1c.reg, TcpSeg.reg, JumboPacket.reg
AddReg             = Copper.reg, Copper1000.reg
AddReg             = RSS.reg, RSS2Q.reg, RSSAdvanced.reg
CopyFiles          = win8.CopyFiles
*IfType            = 6    ; IF_TYPE_ETHERNET_CSMACD
*MediaType         = 0    ; NdisMedium802_3
*PhysicalMediaType = 14 ; NdisPhysicalMedium802_3



[E10DE.Services]
AddService = e1iexpress, 2, win8.Service, win8.EventLog

[E10DE.HW]
Include            = machine.inf
Needs              = PciIoSpaceNotRequired.HW
AddReg             = MSI.reg
```

Как правило, директивы описывают связку действия и секции, в которой находятся параметры для исполнения.

В дальнейшем, мы воспользуемся inf-файлом e1000e для добавления информации в следующие разделы реестра:

- ***ControlSet001\Services***
- ***ControlSet001\Class***
- ***ControlSet001\Control\NetworkSetup2\Interfaces***

Подробное описание того, что будет добавлено в данные разделы будет описано в соответствующих разделах данной работы. А пока что опишем директивы *AddReg* и *AddService,* которые мы встретим в дальнейшем.

AddReg:

Данная директива добавляет ключи и параметры в реестр. Формат этой директивы:

AddReg=add-registry-section[,add-registry-section] ...

Где add-registry-section - это название секции, которая описывает добавляемые в реестр параметры. Таким образом,

AddReg = e1c.reg, TcpSeg.reg, JumboPacket.reg

запишет в реестр информацию, описанную в секциях *e1c.reg, TcpSeg.reg*, *JumboPacket.reg.*

Каждая секция, на которую ссылается директива *AddReg,* имеет следующий формат:

[add-registry-section]

reg-root, [subkey],[value-entry-name],[flags],[value][,[value]]

reg-root, [subkey],[value-entry-name],[flags],[value][,[value]]

...

Здесь:

reg-root - это корень, относительно которого будут добавлены ключи (subkey). Этот параметр может принимать следующие значения:

1. HKCR - “HKEY_CLASSES_ROOT”
2. HKCU - “HKEY_CURRENT_USER”
3. HKLM - “HKEY_LOCAL_MACHINE”
4. HKU - “HKEY_USERS”
5. HKR - Относительный путь, который зависит от названия секции, в которой записана директива AddReg.

   В случае, если название секции - это *DDInstall*, то
   HKR - это *Driver Software Key*, который равен
   ControlSet001\Class\{SetupClassGUID}\{id}.
   Заполнению этого ключа посвящен раздел 9.5 данной работы

   Если название секции - это DDInstall.HW, то
   HKR - это *Device Key, который равен*
   ControlSet001\ENUM\PCI\{DeviceId}\{InstanceId}\Device Parameters

subkey - опциональный параметр, название ключа относительно корня, в который будут добавлены параметры и их значения. В случае, если ключ не был создан до момента исполнения директивы, он может быть создан.

value-entry-name - опциональный параметр, название параметра в ключе. Если параметр не был создан до момента исполнения директивы, он может быть создан.

flags - опциональный параметр, битовая маска, описанная в виде hex числа, определяет тип добавляемого параметра а также определяет поведение директивы AddReg. Например:

0x00000000 (FLG_ADDREG_TYPE_SZ) - Тип добавляемого параметра REG_SZ

0x00000004 (FLG_ADDREG_DELVAL) - Удаляет описанный subkey или value-entry-name в subkey.

value - значение для value-entry-name

Таким образом, директива "*AddReg = e1c.reg"*, описанная в DDInstall секции с названием [E10DE],

[E10D3]

AddReg=e1c.reg

...

[e1c.reg]

HKR, Ndi\Interfaces, UpperRange, 0, "ndis5"

...

запишет в *Software Key следующие ключи и их значения:*

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class\4d36e972-e325-11ce-bfc1-08002be10318\{ID}\Ndi\Interfaces]

UpperRange="ndis5"

...

AddService:

Данная директива добавляет описание нового сервиса в *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\{ServiceName}*.

Формат этой директивы:

AddService= ServiceName, [flags] ,service-install-section

[,event-log-install-section[,[EventLogType][,EventName]]

А все параметры, добавляемые в этот ключ берутся из секций service-install-section, которые имеют тот же самый формат, что и секции *add-registry-section* у *AddReg*

9.2 Добавление boot critical драйвера

Драйвер e1000e состоит из двух файлов: **e1i63x64.sys** и **net1ix64.inf**
sys-файл кладём в *%SystemRoot%\ System32\drivers*
inf-файл в *%SystemRoot%\INF*

Остается описать сервис в *ControlSet001\Services\{имя сервиса}* , который будет управлять этим драйвером. В дальнейшем будем предполагать, что текущий ControlSet - это ControlSet001

Информацию, которую нужно добавить в реестр, находим в inf-файле, в директиве AddService:
Для этого находим нужную секцию по DeviceID, для e1000e - это *E10DE*

```
[E10DE.Services]
AddService = e1iexpress, 2, win8.Service, win8.EventLog
```

отсюда *e1iexpress* - имя сервиса
win8.Service, win8.EventLog - секции, в которых описывается информация, добавляемая в реестр для этого сервиса.

В конечном итоге в реестр будет добавлен ключ
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\e1iexpress]
полное описание которого можно посмотреть в *Приложении А* к данной работе.

9.3. Описываем новый сетевой интерфейс

В Windows при добавлении нового интерфейса генерируется новый случайный GUID, который в дальнейшем будет ассоциироваться с ним, поэтому, будем считать, что для нашего интерфейса был сгенерирован GUID = *{6C123746-1E93-41BA-99869D7A7C76AA45}*

От нас требуется описать сетевой интерфейс в *ControlSet001\Control\NetworkSetup2\Interfaces\{GUID}\Kernel*

Таким образом, мы будем заполнять ветку *ControlSet001\Control\NetworkSetup2\Interfaces\{6C123746-1E93-41BA-9986 9D7A7C76AA45}\Kernel*

В которой описывается параметры интерфейса, например:

- IfType, REG_DWORD - Тип интерфейса
- CurrentAddress, REG_BINARY - Текущий MAC адрес
- ProtocolList, REG_MULTI_SZ - Используемые протоколы (Tcpip, Tcpip6 и др.)

Для заполнения этой ветки, мы воспользуемся нашей тестовой ВМ и возьмем оттуда шаблонные значения. Для этого потребуется найти среди всех интерфейсов, тот у которого IfDescr совпадает с названием устройства, взятого из inf-файла. Это название определяется всё из того же *DeviceID*

*%**E10D3NC.DeviceDesc**% = E10D3, PCI\VEN_8086&DEV_10D3*

...

E10D3NC.DeviceDesc = "Intel(R) 82574L Gigabit Network Connection"

Таким образом, нужно найти GUID, у которого
ifDescr = *"Intel(R) 82574L Gigabit Network Connection"*

Шаблонные значения для данной ветки можно найти в *Приложении А*

Нам достаточно скопировать все шаблонные значения. Среди них можно встретить противоречивые, например, *CurrentAddress,* определяющий MAC адрес сетевой карты, такие параметры Windows исправит сама после успешного запуска. В данной работе не проверялось, что будет при отсутствии этих параметров на момент запуска по iSCSI.

Добавляем runtime-данные:

IfAlias - уникальное текстовое название сетевого интерфейса, например: “подключение по локальной сети 25”. Таким образом, можем сгенерировать его случайным образом, главное, чтобы он не совпадал с названием других интерфейсов, описанных в реестре.

NetLuidIndex, согласно MSDN - это специальный идентификатор, который может быть использован NDIS и интерфейс провайдерами, для того, чтобы отличить интерфейсы одинакового типа. В связи с этим, он должен быть уникальным в рамках локальной машины.

NDIS провайдер алоцирует этот идентификатор через
NdisIfAllocateNetLuidIndex.
А освобождает при помощи *NdisIfFreeNetLuidIndex.* Таким образом, где-то должна храниться информация про занятые идентификаторы.

Было найдено, что в Windows 7 эта информация содержалась в в реестре:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NDIS\IfTypes\{ifType}
Здесь ifType - тип интерфейса, который указан в шаблонных значениях

Внутри этого ключа содержится параметр
IfUsedNetLuidIndices RG_BINARY,
который является битовой маской, в которой отмечаются занятые идентификаторы.

Максимальный размер этой маски - 256 байтов. Таким образом, всего значений для маски 2^16-1. Размер *IfUsedNetLuidIndices* выясняется из сообщения об ошибке [8],

!!! cci: NdisCoinst: NcipAllocateNetLuidIndex failed with error 0x5aa
!!! dvi: CoInstaller 1: failed(0x000005aa)!
!!! dvi: Error 1450: Insufficient system resources exist to complete the requested service.

когда не получилось добавить новый интерфейс в связи с тем, что количество свободных идентификаторов иссякло.

Ключ и параметр всё еще присутствуют в Windows ядро которых базируется на ядре Windows10, но явно не используются: *IfUsedNetLuidIndices* не изменяется при добавлении сетевых адаптеров.

Поэтому, в нашем случае можно поступить следующим путём: Можно обратить внимание на то, что *NdisIfAllocateNetLuidIndex* для самого первого интерфейса принимает значение 0x80000(2^15), а все последующие интерфейсы просто инкрементируют это значение. Поэтому, чтобы получить *IfUsedNetLuidIndices* просто соберём все значения этого

идентификатора среди интерфейсов, у которых тип совпадает с нашим ifType и выставим в ***NetLuidIndex*** следующий свободный.

9.4. Описываем новое PCI устройство

Нам требуется описать нашу новую сетевую карту в
ControlSet001\Enum\PCI

В общем случае ключ выглядит так:
ControlSet001\Enum\PCI\'DeviceId'\'InstanceId'

InstanceId - это уникальный идентификатор PCI устройства, который сохраняется при перезагрузки машины. Однако, он не обязан сохраняться, если устройство будет подключено в другой слот. Формат зависит от конкретного устройства, которое он описывает. В нашем случае, нас интересует 3 разных контекста:

1. InstanceId у VMWare Express Root Port:
   В обычном случае, InstanceID выглядит следующим образом: 3&18d45aa6&0&B0
   где 3&18d45aa6 - одинаковая часть у всех VMWare Express Root Port в реестре.

   3 - некий 'level'
   18d45aa6 - хеш, который мог быть вычислен на основе device id + соли от 'serial id' устройства. Генерация хеша неизвестна.

   0 - номер шины

B0 = 0b10110000 = 0b10110(22) << 3 | 0b0 = 0 , Device << 3 | Function

Таким образом, для генерации InstanceId у VMware Express Root Port достаточно знать его BDF.

2. Сетевая карта, подключенная к VMware Express Root Port, в таком случае InstanceId - это мак адрес, дополненный нулевым байтом вначале и в конце: 00**0C29FFFF75722**900

3. Сетевая карта, подключенная к PCI Bridge. у VMware это только сетевая карта e1000. Формат InstanceId полностью неизвестен. Выяснить его можно последовательным включением N сетевых карта e1000.
   Имеем:
   BDF = InstanceID
   02:02:00 = 4&b70f118&0&1088
   02:03:00 = 4&b70f118&0&1888
   02:04:00 = 4&b70f118&0&2088
   02:05:00 = 4&b70f118&0&2888
   02:06:00 = 4&b70f118&0&3088
   02:07:00 = 4&b70f118&0&3888
   Таким образом, InstanceID однозначно определяется BDF.

Полный DeviceId нашей карты:
VEN_8086&DEV_10D3&SUBSYS_07D015AD&REV_00

Вставляем шаблонные значения в наш ключ:
ControlSet001\Enum\PCI\VEN_8086&DEV_10D3&SUBSYS_07D015AD&REV_00\000C29FFFF75722900
Полный список шаблонных значений приведен в *Приложении А* к данной работе.

Правим runtime значения

1. **Driver** = {4d36e972-e325-11ce-bfc1-08002be10318}\0004 - Это ссылка на ветку, которую мы добавим в разделе *9.5* данной работы.

2. **LocationInformation** =
   "@System32\drivers\pci.sys,#65536;PCI bus %1, device %2, function %3; **(11,0,0)**"
   (11,0,0) - это BDF нашей сетевой карты

3. **UINumber** = 192
   это PCI\PCI-e Slot, к которому подключена сетевая карта.

9.5. Описываем связь между PCI устройством и драйвером

От нас требуется описать связь между PCI устройством и его драйвером. Для этого, нужно определить, к какому классу устройств относится наша сетевая карта. Узнать это можно всё в том же inf-файле

; Copyright (c) 2006-2016, Microsoft

[Version]
Signature = "$Windows NT$"
Class = Net
ClassGUID = {4d36e972-e325-11ce-bfc1-08002be10318}

Таким образом, от нас требуется добавить новый ключ в
ControlSet001\Control\Class\{4d36e972-e325-11ce-bfc1-08002be10318}\0004

Откуда взялось число 0004?

ключи в *Class\{4d36e972-e325-11ce-bfc1-08002be10318}*
выглядят следующим образом:
Class\{4d36e972-e325-11ce-bfc1-08002be10318}\0001
Class\{4d36e972-e325-11ce-bfc1-08002be10318}\0002
Class\{4d36e972-e325-11ce-bfc1-08002be10318}\0003
0004 - следующий свободный порядковый номер

Значения ключа берутся из inf-файла, из секции DDInstall

В конечном итоге, добавится ключ:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class\{4d36e972-e325-11ce-bfc1-08002be10318}\0004]

Шаблонные значения можно глянуть в *Приложении А* к данной работе.

Добавляем runtime-данные:

1. **NetCfgInstanceId** = {6C123746-1E93-41BA-9986-9D7A7C76AA45}
   это ссылка на сетевой интерфейс, который мы добавляли ранее в разделе 9.3

2. **DeviceInstanceID**=
   PCI\VEN_8086&DEV_10D3&SUBSYS_07D015AD&
   REV_00\000C29FFFF75722900
   это ссылка на нашу карту в ENUM\PCI, которую мы добавляли в разделе 9.4

3. **NetLuidIndex** = 0x80002
   это индекс сетевого интерфейса, который мы добавляли в разделе 9.3

9.6. Связываем новую сетевую карту с родительским PCI устройством

Сетевые карты, которые подключены в VMware Express Root Port обладают следующими особенностями в реестре:

1. их InstanceID - это MAC адрес, дополненный нулевыми байтами.
2. для них требуется создать SlotPersistentInfo ключ в родительском PCI устройстве.

Достоверно неизвестно, для чего используется SlotPersistentInfo.
Возможно следующее:
1. это некий “внутренний кеш” PnP Manager’а
2. он напрямую связывает родительское устройство с дочерним.

сами Microsoft нигде не описывали предназначение данного ключа.

В интернете встречается всего-лишь один контекст с упоминанием SlotPersistentInfo: В марте 2018 года Microsoft выпустила патч, который ломал работу с VMware картами vmxnet3 на Windows Server 2008. [6] Хотфикс заключался в том, что запускался VB скрипт, который удалял любой найденный SlotPersistentInfo из реестра.

Примечательно следующее:
Проблема возникла на виртуальных машинах VMware, на таких же проблемных сетевых картах, которые подключены к VMware Express Root Port.

Поэтому, в случае, если карта подключена к Root Port'у, мы:

1. используем InstanceID = MAC с дополненными нулевыми байтами
2. создаём SlotPersistentInfo у её родительского порта.

Описываем SlotPersistentInfo:

DeviceID у VMware Express Root Port следующий:
VEN_15AD&DEV_07A0&SUBSYS_07A015AD&REV_0

таким образом, мы будем описывать
ControlSet001\Enum\PCI\VEN_15AD&DEV_07A0&SUBSYS_07A015AD&REV_01\3&18d45aa6&0&B0 \Device Parameters\SlotPersistentInfo
Почему ключ имеет такой вид - будет рассказано ниже.

В ENUM\PCI этого порта нам требуется описать ключ InstanceID нашего Root Port по "обычным правилам" В общем случае, level&hash в InstanceID не обязан совпадать для всех дочерних ключей у Enum\Pci\'DeviceID'\ .Вероятна ситуация, когда в рамках одного DeviceID у нас присутствуют следующие InstanceID

Enum\Pci\'DeviceID'\3&18d45aa6&0&B0
Enum\Pci\'DeviceID'\3&18d45aa6&0&B2
Enum\Pci\'DeviceID'\2&15f42ba1&0&00

Почему нас это волнует?

Вычислять самим level&hash, который генерируется PnP Manager'ом по внутренним правилам, не стоит. Зареверсить его генерацию можно. Да и с точки зрения обратной совместимости алгоритм генерации не будет меняться, однако, нас спасает следующий факт: level&hash совпадают у

всех дочерних к DeviceID Root Port'а ключей. поэтому, их можно смело брать из любого
ControlSet001\Enum\PCI\VEN_15AD&DEV_07A0&SUBSYS_07A015AD&REV_01\level&hash&bus&device_function

Для нашего случая, level&hash - **3&18d45aa6**
BDF нашего родительского VMware Root Express Port **0,22,0**
Таким образом,
Bus = 0
DeviceFunction = B0
B0 = (0b10110(22) << 3 | 0b0(0) << 3)

и сам InstanceID = **3&18d45aa6&0&B0**

Поэтому, SlotPersistentInfo мы будем создавать в
ControlSet001\Enum\PCI\VEN_15AD&DEV_07A0&SUBSYS_07A015AD&REV_01\3&18d45aa6&0&B0 \Device Parameters\SlotPersistentInfo

Что из себя представляет SlotPersistentInfo?

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\PCI\VEN_15AD&DEV_07A0&SUBSYS_07A015AD&REV_01\3&18d45aa6&0&B0\Device Parameters\SlotPersistentInfo]
"DEV_00&FUN_00"=hex:c0,f8,d4,41,86,80,d3,10,00,00,00,02,ad,15,d0,07,00,00,5b,41,88,d2,ff,ff,29,72,75,ff,ff,29,0c,00

DEV%02d&FUN%02d К примеру,
DEV00&FUN00, (DEV - device, FUN - function.)

Это device function нашей сетевой карты, ссылку на которую мы в данный момент делаем. А конкретно, ссылка на
PCI\VEN_8086&DEV_10D3&SUBSYS_07D015AD&
REV_00\000C29FFFF75722900

Таким образом, в нашем случае это поле имеет следующее название:
DEV_00&FUN_00 (т.к. BDF сетевой карты 11,0,0)

А вот внутри DEV00&FUN00 будут лежать уже бинарные данные.

Пример:
C0 F8 D4 40 86 80 D3 10
00 00 00 02 AD 15 D0 07
00 00 B2 D9 88 E1 FF FF
29 72 75 FF FF 29 0C 00

Можно обратить внимание на то, что последние байты относятся к MAC адресу сетевой карты.
А сами данные лежат в Big Endian.

C0 F8 D4 40	можем занулить, опустим	
86 80	8086	: VEN_8086
D3 10	10D3	: DEV_10D3
00 00 00 02	02	: REV_02
AD 15	15AD	
D0 07	07D0	: SUBSYS_07D015AD
00 00 B2 D9 88 E1 FF FF	можем занулить, опустим	
29 72 75 FF FF 29 0C 00	0C29FFFF757229	: MAC(InstanceID)

Таким образом, выясняется, что это просто сериализованная структура, вид которой следующий:

```
struct {
    uint32_t a //опустим этот параметр.
    uint16_t Vendor
    uint16_t Device
    uint8_t Revision
    uint8_t b // опустим этот параметр
    uint8_t SubClass
    uint8_t BaseClass
    uint16_t SubVendor
    uint16_t SubSystem
    uint8_t c // опустим параметр + несколько байт мусора из-за
выравнивания
    uint64_t MAC// InstanceID нашей карты (MAC)
}
```

Предполагается, что это просто прямая ссылка на Enum\PCI нашей карты. К тому же, описанный здесь MAC адрес не является MAC адресом карты у Destination, что только дополнительно подчёркивает, что это просто ссылка на ключ в реестре.

Теперь у нас есть всё, что требуется для создания *ControlSet001\Enum\PCI\VEN_15AD&DEV_07A0&SUBSYS_07A015AD&REV_01\3&18d45aa6&0&B0 \Device Parameters\SlotPersistentInfo*

Создаем и заполняем SlotPersistentInfo в реестре Source VM.

Еще раз про DeviceInstanceID:

Для наглядности, вспомним наш DeviceInstanceID карты:

VEN_8086&DEV_10D3&SUBSYS_07D015AD&
REV_00\000C29FFFF75722900

Нас смущали дополнительные нулевые байты у MAC

*00**0C29FFFF75722900***

Теперь понятно, что дополнительный нулевой байт в самом начале объясняется тем, что в мак адрес лежит в 8 байтовой переменной. Однако, всё ещё непонятно, откуда появился последний нулевой байт. Предполагается, что это связано с фиксированным максимальным размером у InstanceID.

Приложение A:

Шаблонные значения в реестре

ControlSet001\Services

[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\e1iexpress]

"Type"=dword:00000001

"Start"=dword:00000003

"ErrorControl"=dword:00000001

"Tag"=dword:00000010

"ImagePath"=hex(2):5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,00,

74,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,64,00,72,

00,69,00,76,00,65,00,72,00,73,00,5c,00,65,00,31,00,69,00,36,00,33,00,78,00,

36,00,34,00,2e,00,73,00,79,00,73,00,00,00

"DisplayName"="@net1ix64.inf,%e1iExpress.Service.DispName%;Intel(R) PRO/1000 PCI Express Network Connection Driver I"

"Group"="NDIS"

"Owners"=hex(7):6e,00,65,00,74,00,31,00,69,00,78,00,36,00,34,00,2e,00,69,00,6e,

00,66,00,00,00,00,00

"BootFlags"=dword:00000001

"NdisMajorVersion"=dword:00000006

"NdisMinorVersion"=dword:0000001e

"DriverMajorVersion"=dword:0000000c

"DriverMinorVersion"=dword:0000000f

ControlSet001\Control\NetworkSetup2\Interfaces

[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\NetworkSetup2\Interfaces\{6C123746-1E93-41BA-99869D7A7C76AA45}\Kernel]

"IfType"=dword:00000006

"MediaType"=dword:00000000

"PhysicalMediaType"=dword:0000000e

"IfAlias"="Ethernet0 2"

"NetLuidIndex"=dword:00008002

"Characteristics"=dword:00000084

"IfDescr"="Intel(R) 82574L Gigabit Network Connection"

"ProtocolList"=hex(7):52,00,44,00,4d,00,41,00,4e,00,44,00,4b,00,00,00,54,00,63,

00,70,00,69,00,70,00,00,00,4d,00,73,00,4c,00,6c,00,64,00,70,00,00,00,54,00,

63,00,70,00,69,00,70,00,36,00,00,00,72,00,73,00,70,00,6e,00,64,00,72,00,00,

00,52,00,61,00,73,00,50,00,70,00,70,00,6f,00,65,00,00,00,4e,00,64,00,69,00,

73,00,75,00,69,00,6f,00,00,00,6c,00,6c,00,74,00,64,00,69,00,6f,00,00,00,00,

00

"FilterList"=hex:20,78,fd,3b,5c,d6,1b,4c,9f,ea,98,3a,01,96,39,ea,00,00,59,d6,

f4,b5,aa,7d,65,45,8e,41,be,22,0e,d6,05,42,00,00,60,64,0d,b7,35,36,42,4d,b8,

66,b8,ab,1a,24,45,4c,00,00

"CurrentAddress"=hex:00,0c,29,75,72,29

"PermanentAddress"=hex:00,0c,29,75,72,29

ControlSet001\ENUM\PCI

[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\PCI\VEN_8086&DEV_10D3&SUBSYS_07D015AD&REV_00\000C29FFFF75722900]

"DeviceDesc"="@net1ix64.inf,%e10d3nc.devicedesc%;Intel(R) 82574L Gigabit Network Connection"

"LocationInformation"="@System32\\drivers\\pci.sys,#65536;PCI-шина %1, устройство %2, функция %3;(11,0,0)"

"Capabilities"=dword:00000016

"UINumber"=dword:000000c0

"ContainerID"="{77065ad9-c5e8-11eb-a98a-806e6f6e6963}"

"HardwareID"=hex(7):50,00,43,00,49,00,5c,00,56,00,45,00,4e,00,5f,00,38,00,30,

00,38,00,36,00,26,00,44,00,45,00,56,00,5f,00,31,00,30,00,44,00,33,00,26,00,

53,00,55,00,42,00,53,00,59,00,53,00,5f,00,30,00,37,00,44,00,30,00,31,00,35,

00,41,00,44,00,26,00,52,00,45,00,56,00,5f,00,30,00,30,00,00,00,50,00,43,00,

49,00,5c,00,56,00,45,00,4e,00,5f,00,38,00,30,00,38,00,36,00,26,00,44,00,45,

00,56,00,5f,00,31,00,30,00,44,00,33,00,26,00,53,00,55,00,42,00,53,00,59,00,

53,00,5f,00,30,00,37,00,44,00,30,00,31,00,35,00,41,00,44,00,00,00,50,00,43,

00,49,00,5c,00,56,00,45,00,4e,00,5f,00,38,00,30,00,38,00,36,00,26,00,44,00,

45,00,56,00,5f,00,31,00,30,00,44,00,33,00,26,00,43,00,43,00,5f,00,30,00,32,

00,30,00,30,00,30,00,30,00,00,00,50,00,43,00,49,00,5c,00,56,00,45,00,4e,00,

5f,00,38,00,30,00,38,00,36,00,26,00,44,00,45,00,56,00,5f,00,31,00,30,00,44,

00,33,00,26,00,43,00,43,00,5f,00,30,00,32,00,30,00,30,00,00,00,00,00

"CompatibleIDs"=hex(7):50,00,43,00,49,00,5c,00,56,00,45,00,4e,00,5f,00,38,00,

30,00,38,00,36,00,26,00,44,00,45,00,56,00,5f,00,31,00,30,00,44,00,33,00,26,

00,52,00,45,00,56,00,5f,00,30,00,30,00,00,00,50,00,43,00,49,00,5c,00,56,00,

45,00,4e,00,5f,00,38,00,30,00,38,00,36,00,26,00,44,00,45,00,56,00,5f,00,31,

00,30,00,44,00,33,00,00,00,50,00,43,00,49,00,5c,00,56,00,45,00,4e,00,5f,00,

38,00,30,00,38,00,36,00,26,00,43,00,43,00,5f,00,30,00,32,00,30,00,30,00,30,

00,30,00,00,00,50,00,43,00,49,00,5c,00,56,00,45,00,4e,00,5f,00,38,00,30,00,

38,00,36,00,26,00,43,00,43,00,5f,00,30,00,32,00,30,00,30,00,00,00,50,00,43,

00,49,00,5c,00,56,00,45,00,4e,00,5f,00,38,00,30,00,38,00,36,00,00,00,50,00,

43,00,49,00,5c,00,43,00,43,00,5f,00,30,00,32,00,30,00,30,00,30,00,30,00,26,

00,44,00,54,00,5f,00,30,00,00,00,50,00,43,00,49,00,5c,00,43,00,43,00,5f,00,

30,00,32,00,30,00,30,00,30,00,30,00,00,00,50,00,43,00,49,00,5c,00,43,00,43,

00,5f,00,30,00,32,00,30,00,30,00,26,00,44,00,54,00,5f,00,30,00,00,00,50,00,

43,00,49,00,5c,00,43,00,43,00,5f,00,30,00,32,00,30,00,30,00,00,00,00,00

"ConfigFlags"=dword:00000000

"ClassGUID"="{4d36e972-e325-11ce-bfc1-08002be10318}"

"Service"="e1iexpress"

"Driver"="{4d36e972-e325-11ce-bfc1-08002be10318}\\0004"

"Mfg"="@net1ix64.inf,%intel%;Intel Corporation"

"FriendlyName"="Intel(R) 82574L Gigabit Network Connection"

ControlSet001\Class

[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Class\{4d36e972-e325-11ce-bfc1-0800 2be10318}\0004]

"DriverDesc"="Intel(R) 82574L Gigabit Network Connection"

"ProviderName"="Microsoft"

"DriverDateData"=hex:00,80,be,18,ce,8e,d1,01

"DriverDate"="4-5-2016"

"DriverVersion"="12.15.22.6"

"InfPath"="net1ix64.inf"

"InfSection"="E10D3"

"IncludedInfs"=hex(7):6d,00,61,00,63,00,68,00,69,00,6e,00,65,00,2e,00,69,00,6e,

00,66,00,00,00,70,00,63,00,69,00,2e,00,69,00,6e,00,66,00,00,00,00,00

"MatchingDeviceId"="PCI\\VEN_8086&DEV_10D3"

*"*RSSProfile"="4"*

*"*IfType"=dword:00000006*

*"*MediaType"=dword:00000000*

*"*PhysicalMediaType"=dword:0000000e*

"BusType"="5"

"Characteristics"=dword:00000084

*"*FlowControl"="3"*

*"*TransmitBuffers"="512"*

*"*ReceiveBuffers"="256"*

*"*TCPChecksumOffloadIPv4"="3"*

*"*TCPChecksumOffloadIPv6"="3"*

*"*UDPChecksumOffloadIPv4"="3"*

*"*UDPChecksumOffloadIPv6"="3"*

*"*IPChecksumOffloadIPv4"="3"*

"LogLinkStateEvent"="51"

"WaitAutoNegComplete"="2"

"ITR"="65535"

*"*InterruptModeration"="1"*

*"*PriorityVLANTag"="3"*

*"*LsoV2IPv4"="1"*

*"*LsoV2IPv6"="1"*

*"*JumboPacket"="1514"*

*"*SpeedDuplex"="0"*

"MasterSlave"="0"

"AdaptiveIFS"="0"

*"*RSS"="1"*

*"*NumRssQueues"="2"*

*"*MaxRssProcessors"="8"*

*"*RssBaseProcNumber"="0"*

*"*NumaNodeId"="65535"*

*"*RssMaxProcNumber"="63"*

"IfTypePreStart"=dword:00000006

"NetworkInterfaceInstallTimestamp"=hex(b):63,82,77,41,f5,59,d7,01

"InstallTimeStamp"=hex:e5,07,06,00,06,00,05,00,0a,00,1a,00,21,00,6f,00

"DeviceInstanceID"="PCI\\VEN_8086&DEV_10D3&SUBSYS_07D015AD&REV_00\\000C29FFFF7
5722900"

"ComponentId"="PCI\\VEN_8086&DEV_10D3"

"NetCfgInstanceId"="{87A5D378-A417-44C5-8AB1-3C9876B9B785}"

"NetLuidIndex"=dword:00008002

Приложение Б:
Настройка сетевого адаптера в реестре.

От нас требуется добавить логическое подключение сети и настроить новый сетевой интерфейс, который мы добавили в разделе 9.3.

Эти пункты не связаны между собой, но для удобства чтения расположены в одном приложении.

Логическое подключение:

Это то самое “подключение сети”, которое находится в Панель управления\Сеть и Интернет\Сетевые Подключения”

От нас требуется добавить ключ
ControlSet001\Control\Network\’Device Class’\’GUID’\Connection

В нашем случае:
ControlSet001\Control\Network*\4D36E972-E325-11CE-BFC1-08002BE10318*6C123746-1E93-41BA-99869D7A7C76AA45\Connection

в этот ключ нужно добавить 2 поля:

1. Name [string] - названия подключения(My Internet Connection #1)
2. PnPInstanceId [string] - DeviceInstanceId карты, пример: PCI\VEN_8086&DEV_10D3&SUBSYS_07D015AD& REV_00\000C29FFFF75722900

Настройка нового сетевого адаптера:

Нам нужно настроить добавленный нами сетевой интерфейс

Параметры интерфейса (static\dynamic, dhcp и др) лежат в *HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\ 'GUID'*

В нашем случае:
HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\ 6C123746-1E93-41BA-99869D7A7C76AA45

Все параметры этого ключа вполне стандартные и описаны на MSDN в документации по Tcpip, Tcpip6 [7]

Ниже приведён пример такого ключа.

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{6c12
3746-1e93-41ba-99869d7a7c76aa45}]
"EnableDHCP"=dword:00000001
"Domain"=""
"NameServer"=""
"DhcpIPAddress"="192.168.0.106"
"DhcpSubnetMask"="255.255.255.0"
"DhcpServer"="192.168.0.1"
"Lease"=dword:00001c20
"LeaseObtainedTime"=dword:60cf0537
"T1"=dword:60cf1347
"T2"=dword:60cf1dd3
"LeaseTerminatesTime"=dword:60cf2157
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000
"DhcpNameServer"="192.168.0.1"
"DhcpDefaultGateway"=hex(7):31,00,39,00,32,00,2e,00,31,00,36,00,38,00,2e,00,30,\
  00,2e,00,31,00,00,00,00,00
"DhcpSubnetMaskOpt"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,\
  00,35,00,35,00,2e,00,30,00,00,00,00,00
```

"DhcpInterfaceOptions"=hex:fc,00,00,00,00,00,00,00,00,00,00,00,00,00,00,47,
13,cf,60,79,00,00,00,00,00,00,00,00,00,00,00,00,00,00,47,13,cf,60,2f,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,47,13,cf,60,2e,00,00,00,00,00,00,
00,00,00,00,00,00,00,00,00,00,47,13,cf,60,2c,00,00,00,00,00,00,00,00,00,00,
00,00,00,00,47,13,cf,60,2b,00,00,00,00,00,00,00,00,00,00,00,00,00,00,47,
13,cf,60,21,00,00,00,00,00,00,00,00,00,00,00,00,00,00,47,13,cf,60,1f,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,47,13,cf,60,0f,00,00,00,00,00,00,
00,00,00,00,00,00,00,00,00,47,13,cf,60,06,00,00,00,00,00,00,00,04,00,00,00,
00,00,00,00,57,21,cf,60,c0,a8,00,01,03,00,00,00,00,00,00,00,04,00,00,00,00,
00,00,00,57,21,cf,60,c0,a8,00,01,01,00,00,00,00,00,00,00,04,00,00,00,00,00,
00,00,57,21,cf,60,ff,ff,ff,00,33,00,00,00,00,00,00,00,04,00,00,00,00,00,00,
00,57,21,cf,60,00,00,1c,20,36,00,00,00,00,00,00,00,04,00,00,00,00,00,00,00,
57,21,cf,60,c0,a8,00,01,35,00,00,00,00,00,00,00,01,00,00,00,00,00,00,00,57,
21,cf,60,05,00,00,00
"DhcpGatewayHardware"=hex:c0,a8,00,01,06,00,00,00,b0,be,76,a8,88,6e
"DhcpGatewayHardwareCount"=dword:00000001