

Лохматим сервер Cisco WSA S690 (Cisco UCS C240 M4 Server) : Получаем root-доступ к AsyncOS, захватываем управление IMC, устанавливаем альтернативную ОС

23.09.2022

62 Просмотров



Программно-аппаратное решение **Cisco WSA S690** с аппаратной точки зрения строится на базе серверной платформы **Cisco UCS C240 M4**, а с программной точки зрения - на базе предустановленной ОС **AsyncOS**. Владельцы такого решения, оставшиеся без поддержки вендора, и желающие использовать серверную платформу под альтернативные задачи, могут столкнуться с невозможностью использовать сторонние загрузчики инсталляторов операционных систем. В этой статье мы рассмотрим ряд манипуляций, которые позволят нам решить данную проблему.

Подготовка и получение полного доступа к файловой системе AsyncOS

В попытках получения доступа к модулю управления **Cisco IMC**, опираясь на предыдущий опыт (<https://blog.it-kb.ru/2022/09/06/how-to-take-control-of-the-cisco-imc-bmc-and-disable-secure-boot-in-bios-on-cisco-wsa-s695-ucs-c240-m5-server/>) и "Server Installation and Service Guide" (https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/c/hw/C240M4/install/C240M4/replace.html#89328), мы некоторое время безуспешно пытались работать с аппаратной частью сервера на уровне манипуляций с джамперами и DIP-переключателями на материнской плате. Опыты с частично документированным переключателем **SW8** и недокументированным **SW6** не дали желаемого результата. Поэтому было решено подойти к проблеме получения доступа к IMC с другой стороны – со стороны **AsyncOS**.

Процесс изучения возможностей получения полного доступа к AsyncOS был начат со сбора информации об известных уязвимостях ОС. При поверхностном изучении вопроса в открытом доступе мне не удалось найти никаких интересных технических подробностей по той или иной уязвимости, или обнаружить каких-либо готовых эксплоитов, которые бы подходили под нашу версию AsyncOS. Однако в ходе перелопачивания этой информации глаз зацепился за один любопытный проект [github/ss23/cisco-ironport-appliances-service](https://github.com/ss23/cisco-ironport-appliances-service) (<https://github.com/ss23/cisco-ironport-appliances-service>), где описывался принцип получения доступа через сервисный аккаунт "**enablediag**" к старым версиям AsyncOS 8 ветки. Исключительно из спортивного интереса мы скомпилировали исходник проекта в исполняемый файл и проверили его на нашей AsyncOS 12.5.2 ... Не сработало... Оно и не удивительно, столько лет прошло и вендор, наверняка, сделал из этой истории правильные выводы. Но сама идея меня заинтересовала и начались эксперименты в этом направлении. И как оказалось в дальнейшем, направление было выбрано верное и вендор, таки снова наступил на те же грабли и оставил нам возможности для манёвра.

Далее мы рассмотрим сформировавшийся в ходе экспериментов вариант получения **root**-доступа к **AsyncOC 12.5.2** для возможности дальнейшего прямого конфигурирования контроллера IMC непосредственно из этой ОС.

В нашем случае предполагается, что сервер WSA S690 уже ранее был в работе по прямому назначению, то есть AsyncOS на нём уже первично настроена и имеет подключение к локальной сети через порт "**MGMT 1**" с выделенными IP адресом, где доступны такие протоколы подключения к AsyncOS, как HTTP/SSH.

Выделяем новый статический IP адрес, который будет в дальнейшем назначен контроллеру IMC. В нашем примере это будет адрес 10.2.2.5.

Подключаем патч-кордом порт "**RPC**" к порту коммутатора, на котором в дальнейшем будет работать выделенный IP адрес для IMC.



Изучаем старый документ "Cisco Secure Email Gateway / Support FAQ / Enable service account (enablediag) via console on the ESA/WSA/SMA" (<https://www.cisco.com/c/en/us/support/docs/security/email-security-appliance/200151-enable-service-account-on-the-esa-wsa-sm.html>) и предполагаем, что это будет работать и у нас. Проверяем...

По протоколу **SSH** подключаемся к интерфейсу управления WSA (на порт "MGMT 1") с учётной записью "**enablediag**" и паролем стандартного администратора AsyncOS.

```
PuTTY
login as: enablediag
Keyboard-interactive authentication prompts from server:
| enablediag@wsal.holding.com's password:
End of keyboard-interactive prompts from server
AsyncOS 12.5.2 for Web build 011

Welcome to the Cisco S690 Web Security Appliance

Available Commands:
help -- View this text.
quit -- Log out.
service -- Enable or disable access to the service system.
network -- Perform emergency configuration of the diagnostic network interface.
clearnet -- Resets configuration of the diagnostic network interface.
ssh -- Configure emergency SSH daemon on the diagnostic network interface.
clearssh -- Stop emergency SSH daemon on the diagnostic network interface.
tunnel -- Start up tech support tunnel to IronPort.
print -- Print status of the diagnostic network interface.

S/N 700D10A000A0-FCH0000A000
Service Access currently disabled.
wsal.holding.com>
```

Как видим, вход успешно выполнен и перед нами открывается специальное меню с ограниченным набором команд.

Выполняем команду "**service**", чтобы запустить процедуру включения и настройки сервисного доступа к системе.

Утвердительно отвечаем на вопрос о том, хотим ли мы включить сервисный доступ к системе – "**Y**".

На запрос о назначении секретной строки выбираем вариант "**2**" и в следующем запросе вводим произвольную строку, соответствующую простейшим требованиям сложности. В нашем примере используется строка "Cisco1234567890". Запоминаем эту строку, а также серийный номер системы, указанный ниже. В нашем случае это "700D10A000A0-FCH0000A000". Эти данные нам понадобятся в дальнейшем.

```
PuTTY
wsal.holding.com> service

Service Access is currently disabled. Enabling this system will allow an
IronPort Customer Support representative to remotely access your system
to assist you in solving your technical issues. Are you sure you want
to do this? [Y/N]> Y

A random seed string is required to initialize secure communication with Cisco
Customer Support.

1. Generate a random string to initialize secure communication (recommended)
2. Enter a random string
[1]> 2
[]> Cisco1234567890

Service access has been ENABLED. Please provide the string:

Cisco1234567890

to your Cisco Customer Support representative.

S/N 700D10A000A0-FCH0000A000
Service Access currently ENABLED (0 current service logins)
wsal.holding.com>
```

В итоге мы должны увидеть сообщение о том, что на системе теперь включен специальный режим с удалённым доступом для сервисных целей. Здесь нам больше делать нечего, поэтому завершаем сессию командой "**quit**".

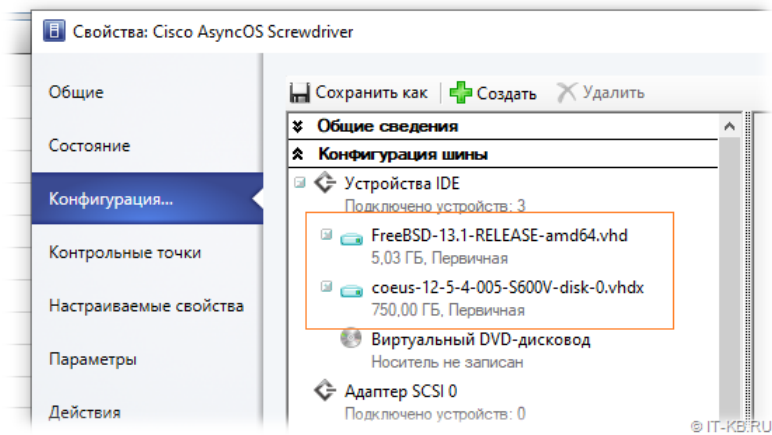
Дальше задача немного усложняется, так как нам потребуется виртуальный аплайнс WSA (**Secure Web Appliance Virtual**), аналогичный или близкий по версии к нашей версии AsyncOS. В нашем примере будет использоваться самая последняя версия аплайнса из 12 ветки: 12.5.4(MD) 26-Apr-2022 (<https://software.cisco.com/download/home/284806698/type/282975114/release/12.5.4>).

Скачиваем с сайта Cisco архив с виртуальным диском WSA под нужную платформу виртуализации, но создавать виртуальную машину пока не торопимся.

Дело в том, что нас интересует не сам по себе запуск виртуального аплайнса, как такового, а запуск любой виртуальной системы, умеющей беспрепятственно работать с файловой системой, используемой в *BSD дистрибутивах – **UFS**. Так как именно эту файловую систему имеют разделы виртуального аплайнса WSA. И наша задача состоит в том, чтобы в загруженной виртуальной ОС мы могли спокойно монтировать дисковые разделы UFS из виртуального диска аплайнса WSA и работать с данными на этих разделах.

Поупражнявшись некоторое время с разными Live-дистрибутивами на базе **FreeBSD** я так и не нашёл для себя удобного и адекватно работающего инструмента для работы в своей среде виртуализации **Hyper-V**. Поэтому было принято решение развернуть чистую VM актуальной версии **FreeBSD 13.1**, используя готовый виртуальный диск VHD, доступный по ссылке /VM-IMAGES/13.1-RELEASE/amd64/Latest/ (<https://download.freebsd.org/ftp/releases/VM-IMAGES/13.1-RELEASE/amd64/Latest/>).

Итак, создаём VM Hyper-V Gen1 с парой vCPU, 6GB ОЗУ и цепляем к ней два виртуальных диска. Первым диском, с которого будет загружаться VM, выставляем диск с оригинальной FreeBSD, а вторым подключаем диск от виртуального аплайнса WSA.



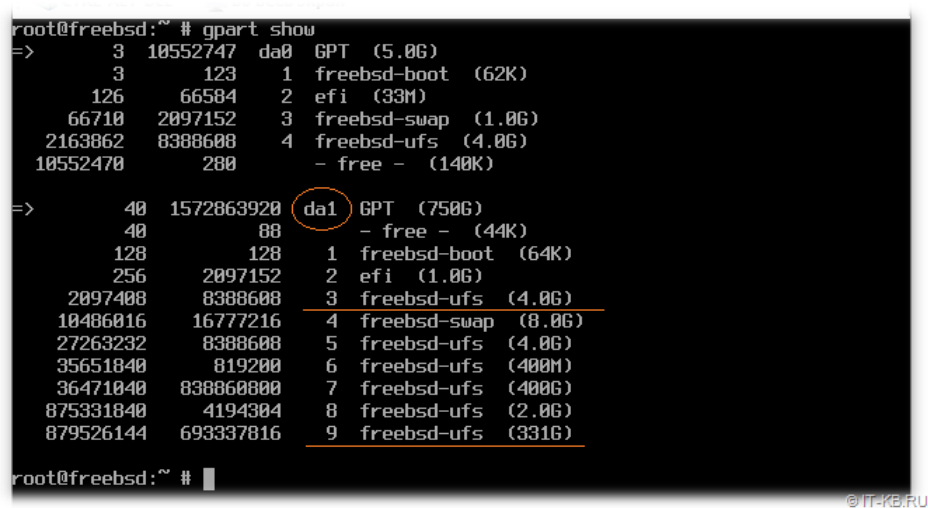
Чтобы с этой системой было удобно работать в дальнейшем и не заморачиваться с ручной настройкой статического IP адреса, сетевой адаптер виртуальной машины можем сразу подключить к виртуальной сети с DHCP-сервером.

Включаем виртуальную машину, ждем загрузки основной системы FreeBSD, входим в неё с правами root. При первом входе мы можем задать пароль для root и при необходимости включить SSH-сервер для удалённого доступа к системе:

```
# echo 'sshd_enable="YES"' >> /etc/rc.conf
# sysrc sshd_enable=YES
# service sshd start
# service sshd status
# netstat -nat | grep LISTEN
```

Далее получим информацию обо всех обнаруженных дисках и дисковых разделах:

```
# geom disk list
# gpart show
```



Как видим, второй виртуальный диск с данными аплайнса WSA размером в 750G у нас в системе определён как **"da1"**. На этом диске нам особо интересны два раздела – **3** и **9**. На третьем разделе размещена корневая система AsyncOS, а на девятом разделе размещены дополнительные файлы, которые нам потребуются.

Создаём в нашей базовой ОС временные каталоги и монтируем в них соответствующие разделы:

```
# mkdir /mnt/wsa-p3
# mkdir /mnt/wsa-p9
# mount -o rw /dev/da1p3 /mnt/wsa-p3
# mount -o rw /dev/da1p9 /mnt/wsa-p9
```

Монтирование разделов должно пройти без ошибок.

```

root@freebsd:~ #
root@freebsd:~ # mkdir /mnt/wsa-p3
root@freebsd:~ # mkdir /mnt/wsa-p9
root@freebsd:~ # mount -o rw /dev/da1p3 /mnt/wsa-p3
root@freebsd:~ # mount -o rw /dev/da1p9 /mnt/wsa-p9
root@freebsd:~ #

```

© IT-KB.RU

Теперь, когда нужные нам разделы смонтированы, нам потребуется выполнить длинную команду. По сути это вызов с помощью **python** специальной библиотеки **gen_pass.so**, в которую мы отправим в качестве входных параметров ранее придуманную нами секретную строку ("Cisco1234567890") и серийный номер WSA ("700D10A000A0-FCH0000A000"). Конечная команда в нашем примере будет выглядеть следующим образом:

```

env LD_LIBRARY_PATH=/mnt/wsa-p3/usr/local/lib:/mnt/wsa-p3/lib:/mnt/wsa-p3/usr/lib PYTHONPATH="/mnt/wsa-p9/python-eggs/phoebe_extensions-1.0.0_000-py2.6_10_amd64_nothr-freebsd-10.4-RELEASE-amd64.ipoe-tmp" SERIAL_NUMBER=700D10A000A0-FCH0000A000 /mnt/wsa-p3/usr/local/bin/python -c 'import sys, gen_pass; print gen_pass.gen_pass(sys.argv[1]);' Cisco1234567890

```

В этой команде серийный номер WSA вводим в переменную "SERIAL_NUMBER", а секретную строку вводим в самом конце команды. В результате выполнения этой команды мы должны получить от библиотеки gen_pass.so ответ в виде некоторой последовательности символов.

```

root@freebsd:~ #
root@freebsd:~ # env LD_LIBRARY_PATH=/mnt/wsa-p3/usr/local/lib:/mnt/wsa-p3/lib:/mnt/wsa-p3/usr/lib PYTHONPATH="/mnt/wsa-p9/python-eggs/phoebe_extensions-1.0.0_000-py2.6_10_amd64_nothr-freebsd-10.4-RELEASE-amd64.ipoe-tmp" SERIAL_NUMBER=700D10A000A0-FCH0000A000 /mnt/wsa-p3/usr/local/bin/python -c 'import sys, gen_pass; print gen_pass.gen_pass(sys.argv[1]);' Cisco1234567890
jNaVmnKS8n2XAHjK
root@freebsd:~ #
root@freebsd:~ #

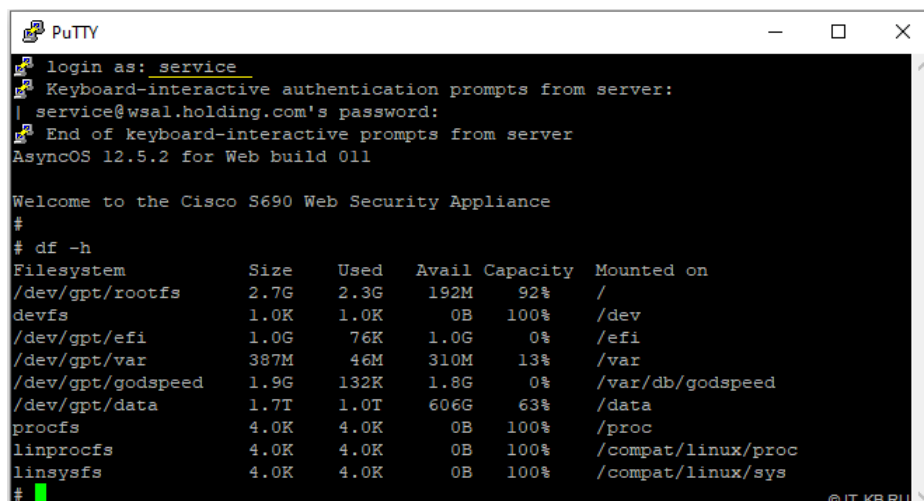
```

© IT-KB.RU

Это и есть тот самый пароль, с которым подразумевается подключение специалистов службы поддержки Cisco к нашему серверу WSA после включения режима **"service"** из под пользователя **"enablediag"**.

Завершаем сеанс работы в виртуальной машине с FreeBSD, так как она свою функцию выполнила и больше нам не нужна.

Теперь подключаемся по **SSH** к нашему серверу **WSA** и при запросе аутентификации вводим логин **"service"** и пароль полученный выше.



```

PuTTY
login as: service
Keyboard-interactive authentication prompts from server:
| service@wsal.holding.com's password:
End of keyboard-interactive prompts from server
AsyncOS 12.5.2 for Web build 011

Welcome to the Cisco S690 Web Security Appliance
#
# df -h
Filesystem      Size   Used  Avail Capacity  Mounted on
/dev/gpt/rootfs  2.7G   2.3G   192M    92%      /
devfs            1.0K   1.0K    0B    100%    /dev
/dev/gpt/efi     1.0G    76K    1.0G     0%    /efi
/dev/gpt/var     387M    46M   310M    13%    /var
/dev/gpt/godspeed 1.9G   132K   1.8G     0%  /var/db/godspeed
/dev/gpt/data    1.7T   1.0T   606G    63%    /data
procfs           4.0K   4.0K    0B    100%    /proc
linprocfs        4.0K   4.0K    0B    100%  /compat/linux/proc
linsysfs         4.0K   4.0K    0B    100%  /compat/linux/sys
#

```

© IT-KB.RU

Как видим, мы получили полный доступ к файловой системе AsyncOS и теперь нам доступны все средства и инструменты, которые имеются в её арсенале.

Если в нашем распоряжении больше одного сервера WSA S690 с подобной ОС AsyncOS 12.5, то развёрнутая на FreeBSD виртуальная "песочница" и выполнение указанной выше команды позволит нам получить root-доступ для каждого такого сервера. Для этого в команде достаточно лишь менять переменную с известным нам серийным номером WSA и заведомо известную нам секретную строку.

Настройка контроллера CIMC с помощью IPMI

Ранее мы уже упоминали (<https://blog.it-kb.ru/2022/09/06/how-to-take-control-of-the-cisco-imc-bmc-and-disable-secure-boot-in-bios-on-cisco-wsa-s695-ucs-c240-m5-server/>) о том, что специфичное конфигурирование порта **"RPC"** в стандартной оболочке AsyncOS выполняется командой **"remotepower"**, поэтому давайте присмотримся к этому процессу повнимательней.

Понять, что происходит с контроллером IMC в тот момент, когда выполняется команда "remotepower", нам поможет анализ найденного скрипта, генерируемого утилитами в этой ОС. Найти этот скрипт оказалось довольно просто:

```
# find /* -name "*remotepower*"
/data/link/tmp/remotepower.sh
```

Когда в команде "remotepower" выполняются шаги по включению порта "RPC", то в скрипт попадает следующая последовательность команд:

```
# less /data/link/tmp/remotepower.sh

echo Configuring IPMI
ipmitool raw 0x36 0x03 0x01
sleep 300
ipmitool raw 0x36 0x50 0x01
sleep 15
ipmitool raw 0x36 0x29 0x02
sleep 15
ipmitool raw 0x36 0x14 0x00 0x00 0x00
sleep 15
ipmitool chassis policy previous
sleep 15
ipmitool raw 0x36 0x56 1
sleep 15
ipmitool raw 0x36 0x52 0x00
sleep 15
ipmitool lan set 1 ipsrc static
sleep 5
ipmitool lan set 1 ipaddr 10.2.2.5
sleep 5
ipmitool lan set 1 netmask 255.255.252.0
sleep 5
ipmitool lan set 1 defgw ipaddr 10.2.2.1 255.255.252.0
sleep 5
ipmitool lan set 1 access on
sleep 5
ipmitool lan set 1 auth USER "MD5"
sleep 5
ipmitool lan set 1 auth OPERATOR "MD5"
sleep 5
ipmitool lan set 1 auth ADMIN "MD5"
sleep 5
ipmitool lan set 1 auth CALLBACK "MD5"
sleep 5
ipmitool lan set 1 arp respond on
sleep 5
ipmitool lan set 1 arp generate on
sleep 5
ipmitool user set name 1 petya
sleep 5
ipmitool user set password 1 MyP@ssw0rd1
sleep 5
ipmitool user priv 1 4 1
sleep 5
ipmitool user enable 1
sleep 5
echo Done configuring IPMI
```

А когда в команде "remotepower" выполняются шаги по выключению порта "RPC", то содержимое скрипта меняется и в него попадает следующая последовательность команд:

```
# cat /data/link/tmp/remotepower.sh

echo Configuring IPMI
ipmitool raw 0x36 0x52 0x00
sleep 15
ipmitool lan set 1 ipsrc static
sleep 5
ipmitool lan set 1 ipaddr 0.0.0.0
sleep 5
ipmitool lan set 1 netmask 0.0.0.0
sleep 5
ipmitool lan set 1 defgw 0.0.0.0 255.255.255.0
sleep 5
ipmitool lan set 1 access off
sleep 5
ipmitool user disable 1
sleep 5
echo Done configuring IPMI
```

Как мы понимаем, AsyncOS выполняет настройку контроллера IMC с помощью интерфейса **IPMI**, напрямую общаясь с контроллером средствами утилиты **ipmitool**. И часть трудно интерпретируемых команд **raw** используется для форсированного ограничения доступа к IMC по таким протоколам как HTTP и SSH.

Эксперименты показали, что если порт RPC уже настроен (включен) с помощью команды "remotepower", то для того, чтобы на IP адресе контроллера IMC активизировался протокол **HTTP**, в AsyncOS достаточно выполнить одну команду:

```
# ipmitool raw 0x36 0x52 0x01
```

После выполнения этой команды можно не переживать за последующие перезагрузки сервера, так как, опыты показали, что AsyncOS 12.5.3 не контролирует настройки доступа к IMC на уровне протокола HTTP при перезагрузке ОС.

Однако следует учесть тот факт, что после перезагрузки ОС в IMC будет установлена та учётная запись, которую мы указали ранее в команде "remotepower".

То есть фактически дефолтная для IMC учётная запись "admin" будет замена учётной записью из "remotepower".

Если помимо HTTP, мы хотим иметь доступ к IMC ещё и по **SSH** (а практика показывает (<https://blog.it-kb.ru/2022/09/15/upgrade-and-downgrade-cimc-controller-firmware-on-cisco-ucs-server-to-solve-the-problem-with-session-expired-flash-content/>), что это крайне полезно), то нам поможет команда следующего вида:

```
# ipmitool raw 0x36 0x52 0x0f
```

Таким образом, чтобы настроить параметры IP для IMC и получить доступ к нему по протоколам HTTP/SSH, нам достаточно будет сначала включить порт "RPC" стандартной командой "remotepower" (пример был ранее (<https://blog.it-kb.ru/2022/09/06/how-to-take-control-of-the-cisco-imc-bmc-and-disable-secure-boot-in-bios-on-cisco-wsa-s695-ucs-c240-m5-server/>)), а затем выполнить в AsyncOS с root-правами последнюю команду.

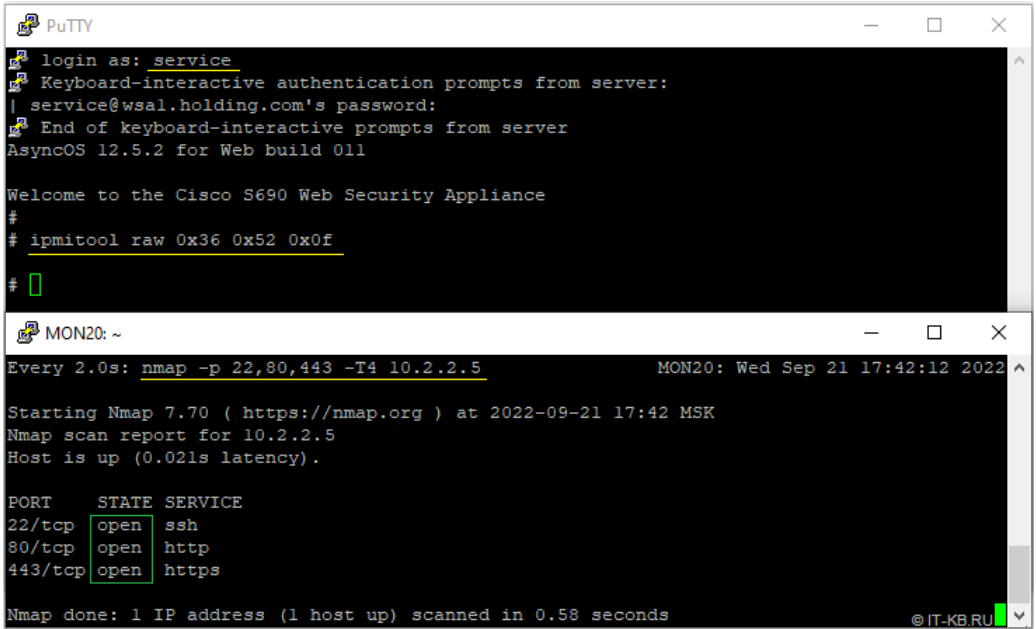
При этом следует учитывать то, что описанные здесь команды были проверены лишь для микрокода IMC версии **2.0(4c)**, и в более поздних версиях микрокода это может уже не сработать и там потребует дополнительное изучение вопроса. Например, было замечено, что при обновлении IMC до более поздних версий 2.0(13*) после выполнения команды "**ipmitool lan set 1 ipsrc static**" (при попытке воспроизвести верхний листинг команд включения "remotepower") мы можем получать ошибку вида:

```
LAN Parameter Data does not match! Write may have failed
```

В этом случае может помочь изменение последовательности выполнения команд по настройке IP, аналогичных тем, что выполняются при включении "remotepower":

```
...
# ipmitool lan set 1 access on
# sleep 5
# ipmitool lan set 1 netmask 255.255.252.0
# sleep 5
# ipmitool lan set 1 defgw ipaddr 10.2.2.1 255.255.252.0
# sleep 5
# ipmitool lan set 1 ipaddr 10.2.2.5
# sleep 5
# ipmitool lan set 1 ipsrc static
...
```

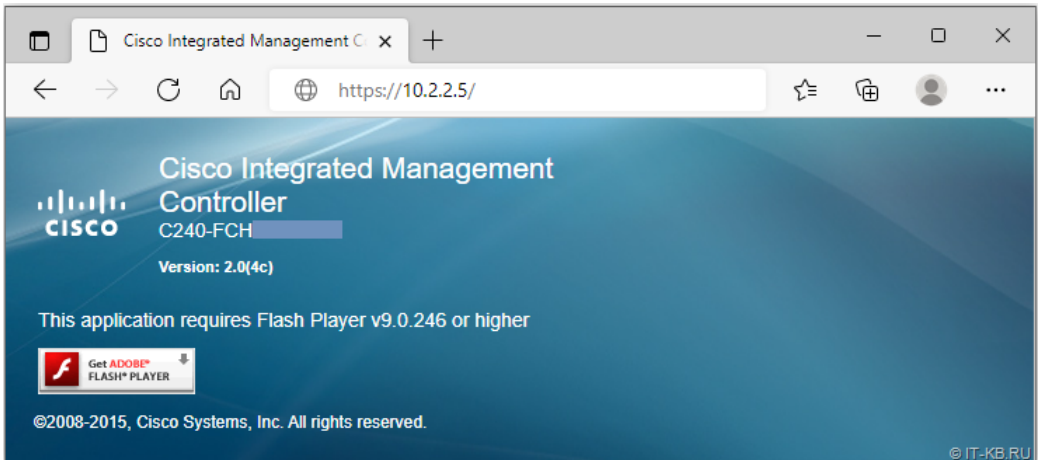
Итак, после настройки параметров IP (через включение "remotepower") и включения протоколов доступа (в root-окружении на AsyncOS из под учётной записи "service") проверим с помощью утилиты **nmap** доступность соответствующих TCP портов на контроллере IMC:



Как видим, порты HTTP/SSH теперь открыты и можно переходить к непосредственному управлению контроллером IMC.

Подключение к интерфейсам управления IMC

Теперь можем попробовать открыть IP адрес контроллера IMC в браузере.



Как видим, веб интерфейс IMC нам доступен, но в нашем случае на контроллере в данный момент старая версия микрокода, где реализация Web UI построена с использованием **Adobe Flash**. Современные браузеры не смогут работать с таким контентом в силу недоступности **Adobe Flash Player**.

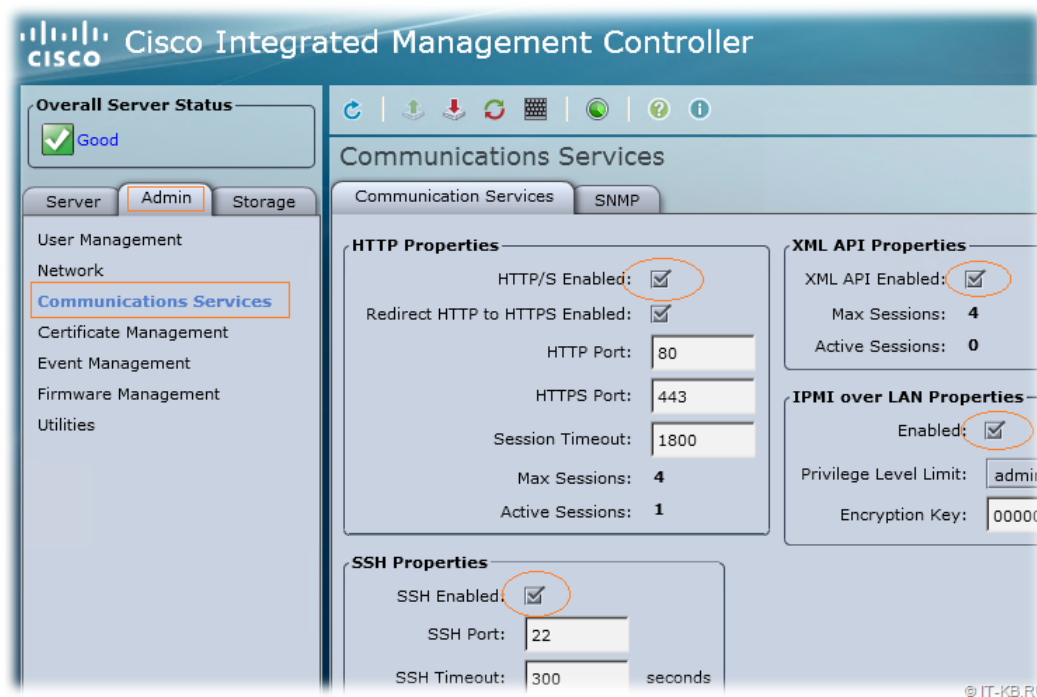
Предпринятые попытки использовать всевозможные **Flash Emulator**, поставляемые в виде расширений к современным версиям браузеров, оказались безуспешными, так как дальше первичной страницы входа нам пройти не удавалось.

Для решения этой проблемы, вместо устаревшей версии Web UI мы можем использовать **SSH** и выполнить обновление микрокода IMC так, как это было описано ранее (<https://blog.it-kb.ru/2022/09/15/upgrade-and-downgrade-cimc-controller-firmware-on-cisco-ucs-server-to-solve-the-problem-with-session-expired-flash-content/>). Но если всё же хочется поупражняться с Web UI, то можно развернуть временную виртуальную машину со старой версией Windows и установить туда старые версии Flash Player и Java (пригодится для работы с vKVM). Мы развернули временную VM с **Windows Server 2012 R2** с обновлениями Windows от 2018 года и поставили туда старую версию **ActiveX** плагина **Adobe Flash Player 10.1.53.64** , которая оказалась под руками. После этого удалось войти в Web UI и штатно работать с интерфейсом CIMC.

Для аутентификации в IMC используем те учётные данные, которые ранее были заданы нами на этапе включения команды "remotepower" (в нашем примере выше это пользователь "petya" с паролем "MyP@ssw0rd1"). Если же эта команда не выполнялась и мы самостоятельно полностью конфигурировали IMC с помощью **ipmitool** из root-окружения AsyncOS, не указывая при этом явном виде учётную запись для доступа, то в IMC должна работать учётная запись по умолчанию с логином "**admin**" и паролем "**password**".

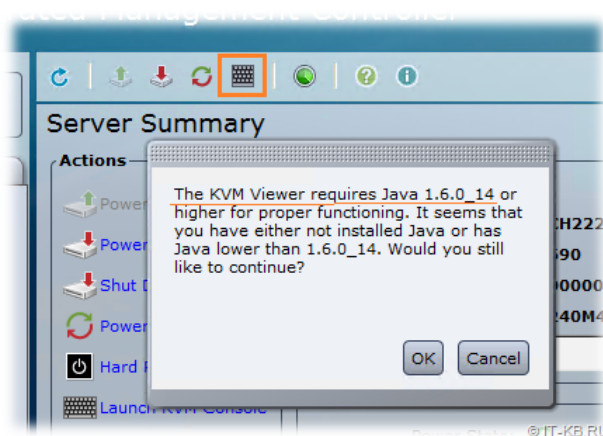


После входа в интерфейс управления IMC первым делом нам важно убедиться в том, что включены все возможные протоколы удалённого доступа, так как это может нам пригодиться в дальнейшем в случае возникновения проблем при обновлении микрокода IMC. Заглянем для этого на вкладку "Admin" в раздел "Communications Services".



В проблемных сетях, возможно, нелишним будет небольшое увеличение таймаутов ожидания для HTTP/SSH.

В дальнейшем нам может оказаться полезным иметь удалённый доступ к консоли сервера, чтобы выключать/включить сервер, менять настройки BIOS, управлять процессом загрузки и так далее. Сделать это можем мы с помощью веб-консоли **vKVM**. Однако следует понимать, что старые версии микрокода IMC имеют и старые версии vKVM, требующие для своей работы определённых древних версий **Java**. Например, в нашем случае при попытке запустить vKVM появилось сообщение о необходимости Java не ниже 1.6.0_14.



Для проверки работы vKVM на нашей виртуальной машине была развёрнута имеющаяся под рукой старая версия **JRE 1.6.0_23**.

Помимо проверки доступности веб интерфейса, крайне желательно убедиться в успешности подключения к IMC по протоколу **SSH**.

Конфигурирование RAID в IMC

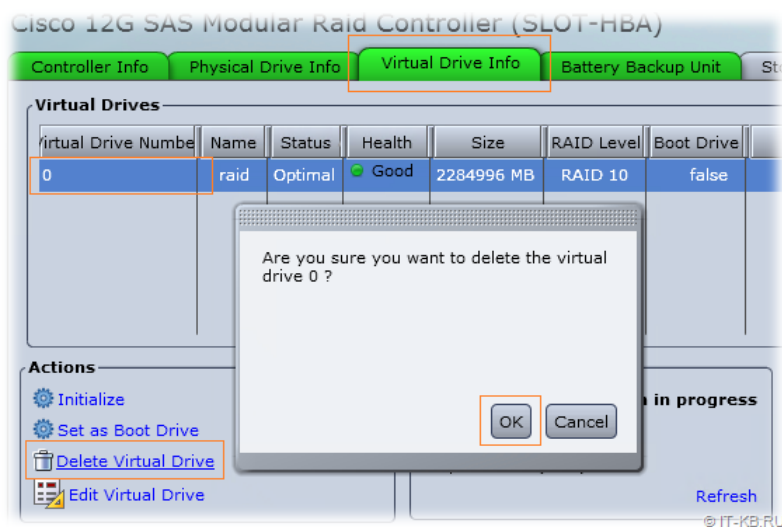
Так как AsyncOS со своими возможностями уже позволила нам захватить полноценное управление контроллером IMC, и теперь её ценность для нас сведена к нулю, мы можем спокойно её изничтожить. Для этого мы воспользуемся вкладкой "Storage" в веб интерфейсе IMC, где сможем разобрать виртуальный диск RAID, на котором развёрнута ОС и расположены все данные WSA.

Перед удалением виртуального диска RAID нам потребуется отключить флаг загрузочного тома ("**Clear Boot Drive**") для этого RAID массива:



После этого переходим на вкладку "**Virtual Drive Info**" и выполняем удаление виртуального диска **vd0** с предустановленной AsyncOS.

Это позволит нам в дальнейшем избежать загрузки ОС WSA и мы уже сможем при дальнейших работах с микрокодом BIOS/CIMC не переживать о том, что AsyncOS нам вдруг снова ломает доступ к IMC.



Ну и теперь здесь же, мы можем по создать новый RAID массив в желаемой конфигурации. В нашем случае в сервере используются диски с физическим блоком 4K, поэтому при построении нового RAID, с которого в последующем должна загружаться ОС, потребуются переключить загрузку в **UEFI** режим. В противном случае мы можем столкнуться с ошибкой "Cannot set 4K native block size drives to boot drive" при попытке включения флага загрузочного тома ("**Set as Boot Drive**").

Выбор версий микрокода BIOS/CIMC

Базовые понятия про возможности обновления микрокода на выделенных серверах Cisco UCS мы рассмотрели ранее (<https://blog.it-kb.ru/2022/09/06/how-to-take-control-of-the-cisco-imc-bmc-and-disable-secure-boot-in-bios-on-cisco-wsa-s695-ucs-c240-m5-server/>).

Образы с разными версиями микрокода (образы HUU) для серверной платформы **Cisco UCS C240 M4** доступны по ссылке: "UCS Server Firmware" (<https://software.cisco.com/download/home/286281356/type/283850974>)

В нашем случае на серверах WSA S690 в качестве исходных версий выступает **BIOS 2.0.4a** и **IMC 2.0.4c** (эти версии можно найти в образе HUU 2.0(4c) 14-May-2015 ([https://software.cisco.com/download/home/286281356/type/283850974/release/2.0\(4c\)](https://software.cisco.com/download/home/286281356/type/283850974/release/2.0(4c))) с файлом ucs-c240m4-huu-2.0.4c.iso). Secure Boot в этой версии "суровый" и не позволяет запускать никакие сторонние загрузчики/инсталляторы ОС.

На форумах Cisco можно встретить информацию (<https://community.cisco.com/t5/unified-computing-system-discussions/cimc-secure-boot-disable/td-p/4294761>) о том, что на родственной платформе того же поколения UCS C220 M4 загрузка Secure Boot имеет ослабленный режим между релизами BIOS 2.0.6b - 2.0.9b. Однако в случае с C240 M4 это неприменимо.

Мы провели серию экспериментов по даунгрейду и апгрейду связок микрокода BIOS/IMC в попытках найти те версии, на которых меняется поведение Secure Boot и появляется возможность использования альтернативных загрузчиков. Подведём некоторые итоги.

Даунгрейд **BIOS** до версии **2.0.3d** (из образа HUU 2.0(3j)1 14-May-2015

([https://software.cisco.com/download/home/286281356/type/283850974/release/2.0\(3j\)1](https://software.cisco.com/download/home/286281356/type/283850974/release/2.0(3j)1)) с файлом ucs-c240m4-huu-2.0.3j-1.iso) показал то, что в настройках BIOS появляется возможность выключения UEFI Boot Mode (работающее переключение в Legacy Boot). А в режиме UEFI загрузка работает то ли без участия Secure Boot, то ли в каком-то ослабленном режиме, то есть появляется возможность загружать произвольные загрузочные накопители и устанавливать на диски сервера альтернативную ОС.

Понижать дальше версии BIOS/IMC особого интереса не было, чтобы не скатиться на совсем старый микрокод. Поэтому следующие эксперименты выполнялись с постепенным пошаговым повышением версий вплоть до самой крайней версии HUU 4.1(2k) 27-Jun-2022

([https://software.cisco.com/download/home/286281356/type/283850974/release/4.1\(2k\)](https://software.cisco.com/download/home/286281356/type/283850974/release/4.1(2k))). В опытную выборку попали только сборки HUU, в которых повышалась версия BIOS (сборки с одинаковыми версиями BIOS были пропущены). В общей сложности было выполнено 35 последовательных апгрейдов связки IMC/BIOS, из которых была выявлена лишь одна связка, позволяющая с некоторыми ограничениями загружать альтернативные загрузчики. Это HUU 3.0(1c) 14-Dec-2016 ([https://software.cisco.com/download/home/286281356/type/283850974/release/3.0\(1c\)](https://software.cisco.com/download/home/286281356/type/283850974/release/3.0(1c))) (файл образа ucs-c240m4-huu-3.0.1c.iso с вложенным микрокодом **BIOS** версии **3.0.1b** и **CIMC** версии **3.0.1c**). Я сказал "с некоторыми ограничениями", потому что возникло ощущение, что в этой связке BIOS/IMC работает только UEFI, а Secure Boot не отключается полностью, а работает в каком-то ослабленном режиме. Например, без проблем загружаются Windows Server 2016/2022, загружаются ESXi 6.5/6.7, но при этом, что само по себе довольно странно, не работает загрузка того же образа HUU.

Попытка максимально поднять версию IMC до 4.1(2k) при использовании интересующей нас версии BIOS 3.0.1b, по аналогии с тем, как мы это делали с платформой M5 (<https://blog.it-kb.ru/2022/09/06/how-to-take-control-of-the-cisco-imc-bmc-and-disable-secure-boot-in-bios-on-cisco-wsa-s695-ucs-c240-m5-server/>), привела к тому, что уже установленная и ранее успешно загружаемая Windows Server 2016 переставала загружаться.

Также следует отметить тот факт, что ни на одной из проверенных версий мне не удалось воспользоваться кнопкой **"F8"** для базовой настройки IMC в ходе загрузки серверной платформы. Даже в тех версиях BIOS, где данный пункт при загрузке был доступен, попытка нажатия "F8" перебрасывала нас в настройки BIOS.

Если у кого-то появится время и стойкое желание самостоятельно протестировать все возможные комбинации прошивок для C240 M4 (их ~80 штук), то следует обратить внимание на ряд моментов, которые могут создать дополнительные сложности:

1) Лучше совсем не связываться с конечными версиями 2 ветки IMC, начиная с **2.0.13o** и до **2.0.13q**, так как все они поставляются в комплекте с глюками Flash, о чём было отмечено ранее (<https://blog.it-kb.ru/2022/09/15/upgrade-and-downgrade-cimc-controller-firmware-on-cisco-ucs-server-to-solve-the-problem-with-session-expired-flash-content/#more-25312>). Выбраться из этой "ямы" можно только с помощью SSH. И поэтому крайне важно иметь работающий SSH перед тем, как приступать к очередной перепрошивке IMC.

2) Для возможности распаковать и дешифровать файлы микрокода из образов HUU, начиная с версии **3.0.3a**, нам понадобится утилита **getfw**, которую можно найти в этом образе в подкаталоге /GETFW/.

Причём распаковку следует выполнять именно той утилитой, которая вложена в образ, так как от версии к версии HUU иногда меняется и сама утилита.

Из особенностей использования этой утилиты можно отметить то, что она работает в Linux и требует наличия подгруженного модуля ядра **squashfs** (modprobe squashfs), а также присутствия старой версии **openssl**.

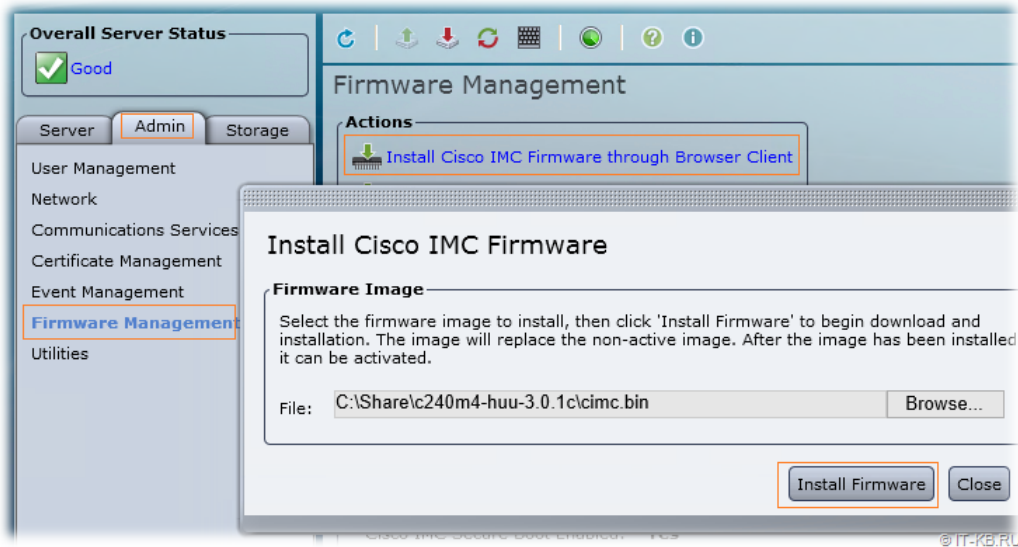
После ряда экспериментов мне удалось заставить работать эту утилиту на BM со старой версией CentOS Linux release 7.2.1511 с предустановленной версией OpenSSL 1.0.1e-fips.

3) Если в результате перепрошивки возникли проблемы и привычные методы доступа к IMC через Web UI или SSH перестали адекватно работать, то можно пробовать альтернативные методы управления IMC (<https://wiki.it-kb.ru/cisco/cisco-integrated-management-controller-cimc-remote-access-methods>) для выполнения отката или дальнейшего повышения версии микрокода.

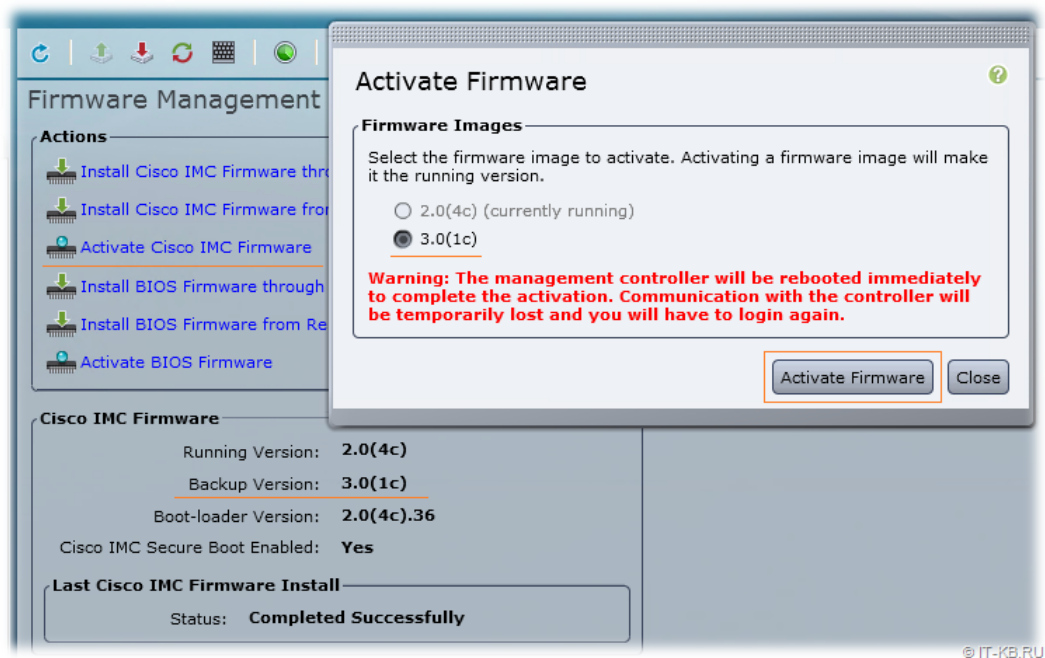
Обновление микрокода CIMC

Итак, мы остановились на том, что под наши задачи следует использовать комплект прошивок IMC/BIOS из образа ucs-c240m4-huu-3.0.1c.iso : **BIOS 3.0.1b** и **CIMC 3.0.1c**.

Для поднятия версии микрокода IMC переходим в веб-интерфейсе IMC на вкладку **"Admin"** > **"Firmware Management"** и выбираем **"Install Cisco IMC Firmware through Browser Client"**. В открывшейся веб форме выбираем файл **cimc.bin** из предварительно распакованного образа HUU. Этот файл можно найти в подкаталоге \cimc файла-контейнера firmware.squashfs из образа ucs-c240m4-huu-3.0.1c.iso. И здесь он ещё не зашифрован и не требует для извлечения утилиты getfw (извлекается простым архиватором типа 7zip).



Дожидаемся окончания верификации, загрузки и установки новой версии прошивки. Загруженная версия будет помечена как **"Backup Version"** и для её активации выбираем пункт **"Activate Cisco IMC Firmware"**:



В ходе активации прошивка **"Backup Version"** перейдёт в стадию **"Running Version"**, а версия, которая была активной ранее, напротив, перейдёт в состояние **"Backup Version"**. Этот удобный механизм даст нам возможность отката (<https://blog.it-kb.ru/2022/09/15/upgrade-and-downgrade-cimc-controller-firmware-on-cisco-ucs-server-to-solve-the-problem-with-session-expired-flash-content/#more-25312>) на ранее установленную версию IMC, если с активированной версией начнутся какие-то серьёзные проблемы.

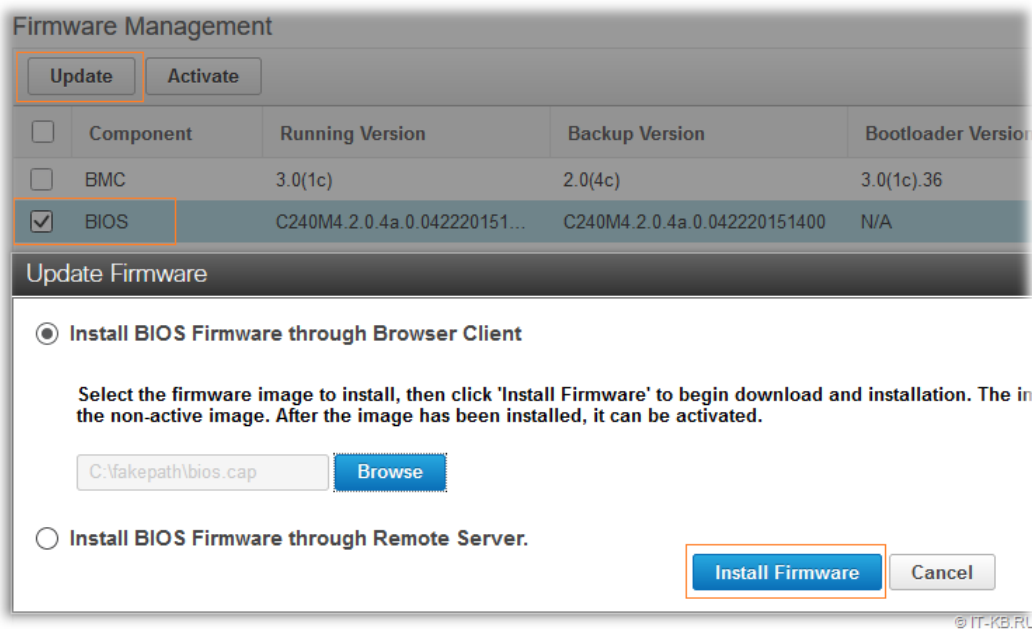
Контроллер IMC в ходе переключения версии микрокода перезагружается и будет недоступен несколько минут. Переподключаемся к обновлённой версии веб-консоли IMC и проверяем текущую активную версию.

<div>Update</div> <div>Activate</div>						
<input type="checkbox"/>	Component	Running Version	Backup Version	Bootloade...	Status	Pr...
<input type="checkbox"/>	BMC	3.0(1c)	2.0(4c)	3.0(1c).36	Completed Successfully	
<input type="checkbox"/>	BIOS	C240M4.2.0.4a.0....	C240M4.2.0.4a....	N/A	Completed Successfully	
<input type="checkbox"/>	BASEFWM4	65023400	65023400	N/A	More...	

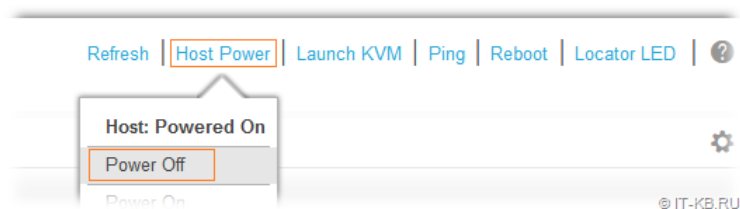
В новой 3 ветке микрокода IMC нам уже не потребуется использовать старые браузеры с поддержкой Adobe Flash, так как веб интерфейс реализован на базе HTML5 и теперь для дальнейшего управления можно будет использовать современные браузеры.

Обновление микрокода BIOS

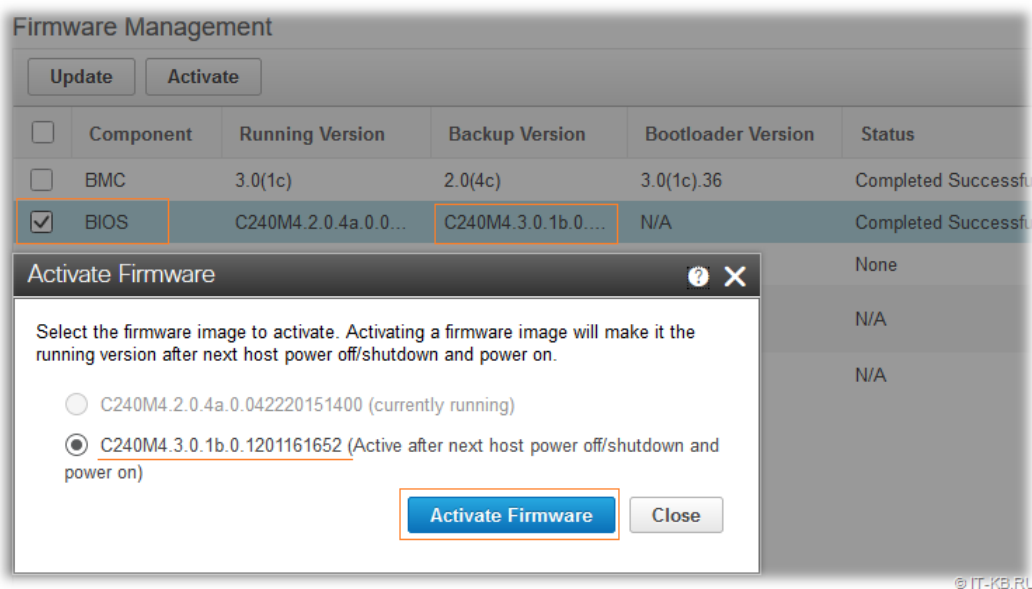
После того, как активирована новая версия IMC, аналогичным образом устанавливаем совместимую с IMC версию BIOS. В открывшейся модальной веб форме выбираем файл **bios.cap** из предварительно распакованного образа HUU. Этот интересующий нас файл можно найти в подкаталоге \bios файла-контейнера firmware.squashfs из образа ucs-c240m4-huu-3.0.1c.iso.



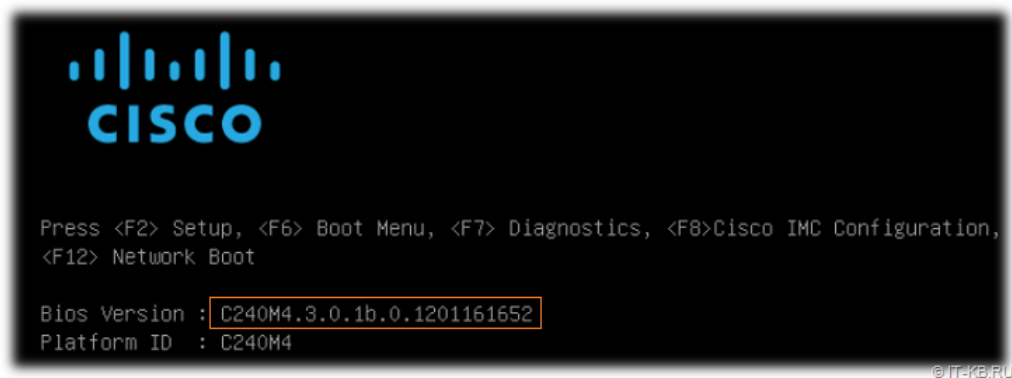
Перед активацией новой версии BIOS выключаем сервер:



Активируем новую версию BIOS:



Пару минут ожидаем, пока в веб интерфейсе не изменится версия "**Running Version**", затем включаем сервер и проверяем результат.



Опциональное обновление микрокода устройств

Как отмечалось выше, проблемы с загрузчиком HUU могут нам не позволить использовать более удобный метод обновления микрокода прочих аппаратных компонент (RAID контроллеры, дисковые накопители и т.п.), как мы описывали ранее (<https://blog.it-kb.ru/2022/09/06/how-to-take-control-of-the-cisco-imc-bmc-and-disable-secure-boot-in-bios-on-cisco-wsa-s695-ucs-c240-m5-server/>). Однако, у нас всё ещё остаётся возможность извлечения более свежих версий микрокода из образов HUU и их прошивка через веб-интерфейс IMC.

Установка желаемой операционной системы

Последним этапом мы выполняем установку желаемой операционной системы на наш бывший сервер Cisco WSA. В ходе установки нам может потребоваться дополнительная подгрузка драйверов, отсутствующих в базовом составе устанавливаемой ОС. В нашем примере при развёртывании ОС **Windows Server 2016** не потребовалось инсталлятору ОС подгружать дополнительные драйверы, так как драйвер совместимый с RAID-контроллером **Cisco 12G SAS Modular Raid Controller (SLOT-HBA)** уже есть базовом наборе драйверов инсталлятора этой ОС.

После успешного окончания установки ОС Windows Server желательно провести замену используемых по умолчанию в Windows драйверов на те драйверы, которые мы можем найти в составе свежего образа `ucs-cxxx-drivers-windows.4.2.2c.iso` ([https://software.cisco.com/download/home/286281356/type/283853158/release/4.2\(2c\)](https://software.cisco.com/download/home/286281356/type/283853158/release/4.2(2c))). В частности, следует запустить инсталлятор драйверов для чипсета Intel из подкаталога `\ChipSet\Intel\CxxxM4\`, а затем обновить драйверы видеоадаптера, сетевых адаптеров и RAID-контроллера.



3 Оценок

Опубликовано в : [Cisco](https://blog.it-kb.ru/category/cisco/) (<https://blog.it-kb.ru/category/cisco/>) , [Hardware](https://blog.it-kb.ru/category/hardware/) (<https://blog.it-kb.ru/category/hardware/>)

Метки : [AsyncoS](https://blog.it-kb.ru/tag/asynco/) (<https://blog.it-kb.ru/tag/asynco/>) , [BIOS](https://blog.it-kb.ru/tag/bios/) (<https://blog.it-kb.ru/tag/bios/>) , [BMC](https://blog.it-kb.ru/tag/bmc/) (<https://blog.it-kb.ru/tag/bmc/>) , [Boot](https://blog.it-kb.ru/tag/boot/) (<https://blog.it-kb.ru/tag/boot/>) , [CIMC](https://blog.it-kb.ru/tag/cimc/) (<https://blog.it-kb.ru/tag/cimc/>) , [Cisco](https://blog.it-kb.ru/tag/cisco/) (<https://blog.it-kb.ru/tag/cisco/>) , [Cisco IMC](https://blog.it-kb.ru/tag/cisco-imc/) (<https://blog.it-kb.ru/tag/cisco-imc/>) , [Cisco UCS](https://blog.it-kb.ru/tag/cisco-ucs/) (<https://blog.it-kb.ru/tag/cisco-ucs/>) , [Cisco WSA](https://blog.it-kb.ru/tag/cisco-wsa/) (<https://blog.it-kb.ru/tag/cisco-wsa/>) , [Downgrade](https://blog.it-kb.ru/tag/downgrade/) (<https://blog.it-kb.ru/tag/downgrade/>) , [Drivers](https://blog.it-kb.ru/tag/drivers/) (<https://blog.it-kb.ru/tag/drivers/>) , [firmware](https://blog.it-kb.ru/tag/firmware/) (<https://blog.it-kb.ru/tag/firmware/>) , [FreeBSD](https://blog.it-kb.ru/tag/freebsd/) (<https://blog.it-kb.ru/tag/freebsd/>) , [Hardware](https://blog.it-kb.ru/tag/hardware/) (<https://blog.it-kb.ru/tag/hardware/>) , [HUU](https://blog.it-kb.ru/tag/huu/) (<https://blog.it-kb.ru/tag/huu/>) , [IMC](https://blog.it-kb.ru/tag/imc/) (<https://blog.it-kb.ru/tag/imc/>)

kb.ru/tag/imc/) , IPMI (<https://blog.it-kb.ru/tag/ipmi/>) , KVM (<https://blog.it-kb.ru/tag/kvm/>) , RAID (<https://blog.it-kb.ru/tag/raid/>) , RPC (<https://blog.it-kb.ru/tag/rpc/>) , Secure Boot (<https://blog.it-kb.ru/tag/secure-boot/>) , UCS (<https://blog.it-kb.ru/tag/ucs/>) , UCS C240 M4 (<https://blog.it-kb.ru/tag/ucs-c240-m4/>) , UEFI (<https://blog.it-kb.ru/tag/uefi/>) , Upgrade (<https://blog.it-kb.ru/tag/upgrade/>) , vKVM (<https://blog.it-kb.ru/tag/vkvm/>) , Windows Server 2016 (<https://blog.it-kb.ru/tag/windows-server-2016/>) , WSA S690 (<https://blog.it-kb.ru/tag/wsa-s690/>)

Добавить комментарий

Введите свой комментарий...

Социальные ссылки

Email: Blog@IT-KB.RU
(<mailto:Blog@IT-KB.RU>)

(<https://blog.it-kb.ru/feed/>)

(https://twitter.com/Blog_IT_KB)

(<https://www.facebook.com/blog.it.kb>)

([https://blog-it-
kb.tumblr.com/](https://blog-it-kb.tumblr.com/))

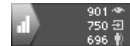
(<https://vk.com/blogitkb>)

Защита SSL



(<https://www.gogetssl.com>)

Статистика



([https://metrika.yandex.ru/stat/?
id=37925125&from=informer](https://metrika.yandex.ru/stat/?id=37925125&from=informer))