AMI Software Utility User Guide

Aptio 5.x AMIBGT User Guide

Document Revision 1.06

August 27, 2021

NDA Required
(NDA)

Legal

Table of Contents

Document Information

Purpose

This document provides information to use the AMIBGT to update the system BIOS.

Audience

Generic BIOS Engineers, OEM Engineers, and Aptio Customers.

Change History

Date	Revision	Description
2013-11-11	1.00	Initial draft
2016-11-28	1.01	Added /CAPSULE and /RECOVERY commands.
2017-04-27	1.02	Added answer for Windows digitally signed driver.
2018-07-10	1.03	Update descriptions of commands and options. Added options information.
2020-02-04	1.04	Update Firmware Requirements. Added Linux Pre-Requisites and Signing Driver and Enrolling Public Key to the System.
2020-10-26	1.05	Update Firmware Requirements.
2021-08-20	1.06	Update error code definition.

Introduction

Overview

AMIBGT (**AMI B**IOS **G**uard Firmware Update **T**ool) is a package of utilities used to update the system BIOS under various operating systems. AMIBGT only works for APTIO with BIOS GUARD support.

AMIBGT Features

This list of features is supported from command line, command prompt, EFI Shell, or Linux shell.

- Flash ROM image
- Command line operating

Requirements

Supported Operating System

AMIBGT is supported by the following operating systems:

- ~~Microsoft® Windows® 7~~
- Microsoft® Windows® 8
- Microsoft® Windows® 8.1
- Microsoft® Windows® 10
- EFI Shell
- Linux

Firmware Requirements

- Compatible with Aptio V.
- For supporting BIOS Guard Flash, the following eModules are required:
  - Intel Bios Guard Technology (5.008_IntelBiosGuard_003)
  - BGT 5.05.0033 or later versions must use the specific BIOS Guard module to support Bios Guard OFBD flash interface. For the actual module version, please refer to the module release note on CRB project.
  - RomImage (5.008_RomImage_001)
  - Flash – Source(5.004_Flash_06)

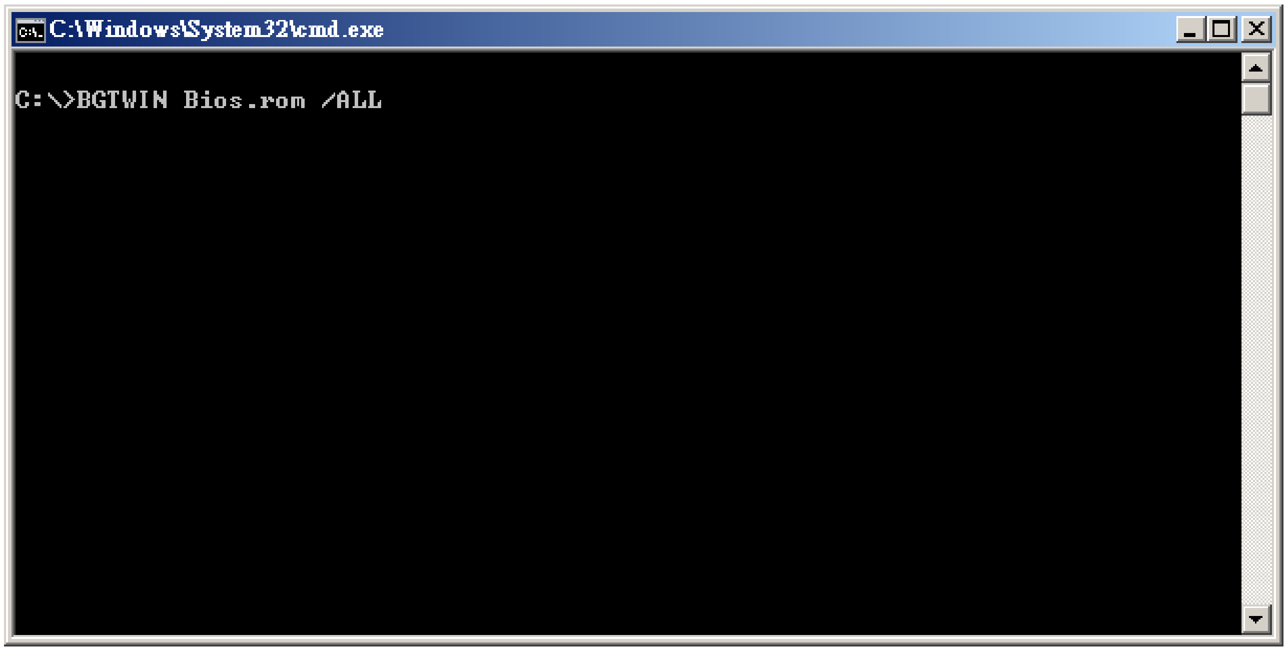
AMIBGT Operation

Overview

This chapter explains the operation of AMIBGT.

The AMIBGT operation mode includes all of the AMIBGT features such as programming all regions with a BIOS ROM file and programming Main BIOS with a BIOS ROM file.

An example of BGTWIN programming in all regions with a BIOS ROM file command screen is shown below:



```
C:\>BGTWIN Bios.rom /ALL
```

Features and Functions

Overview

The AMIBGT offers the following features:

- Program all regions with a BIOS ROM file
- Program Main BIOS with a BIOS ROM file

These features are explained in more detail in this chapter.

Program all regions with a BIOS ROM file

The following command programs all regions with a BIOS ROM file:

BGTEFI <Input BIOS ROM File Name> /ALL

Where BIOS ROM File Name, the mandatory field is used to specify path/filename of the BIOS ROM file with extension.

Program Main BIOS with a BIOS ROM file

The following command programs Main BIOS with a BIOS ROM file:

BGTEFI <Input BIOS ROM File Name> /P

Where BIOS ROM File Name, the mandatory field is used to specify path/filename of the BIOS ROM file with extension.

Options

BGTEFI <BIOS ROM File Name> [Option 1] [Option 2]

Or

BGTEFI <BIOS ROM File Name> <Command>

BIOS ROM File Name

The mandatory field is used to specify path/filename of the BIOS ROM file with extension.

Commands

The mandatory field is used to select an operation mode.

- /BIOSALL	Flash all BIOS block
- /MEALL	Flash all ME region
- /ALL	Flash all (BIOS+ME)
- /CAPSULE	Override BIOS Guard Flash policy to Capsule.
- /RECOVERY	Override BIOS Guard Flash policy to Recovery. (*1)

Options

The optional field supplies more information for flashing BIOS ROM. Following lists the supported optional parameters and format (*2):

- /DESC	Flash descriptor region
- /EC	Flash EC region
- /GBE	Flash GBE region
- /ME	Flash ME region (Need to disable ME)
- /PAD	Flash Padding region (Gap between ME region and Bios region)
- /N	Flash NVRAM region
- /NB	Flash NVRAM backup region
- /OA	Flash OA3 region
- /P	Flash FV_MAIN region

- /DATA	Flash FV_DATA region
- /AB	Flash FV_BB_AFTER_MEMORY region
- /FSPS	Flash FV_FSP_S region (If support FSP)
- /FSPTM	Flash FV_FSP_T_M region (If support FSP)
- /B	Flash FV_BB region
- /OEM	Flash OEM region (Only support in CPASULE mode. /CAPSULE /OEM)
- /P /B /N /CAPSULE	Do capsule update and update FV_MAIN, all boot blocks and NVRAM (/p /b /n usage is same as AFU tool)
- /P /B /N /RECOVERY	Do recovery and update FV_MAIN, all boot blocks and NVRAM (/p /b /n usage is same as AFU tool)

* 1: BGT only sends the recovery file name to BIOS. The action of flashing is handled by BIOS recovery module. The input file must be in root path and only supports 8+3 format.

* 2: This option list demonstrates the default setting. The list is determined by the design of the BIOS ROM in use. The option items, the item order and the item description can be customized during the BIOS implementation.

Rules:

- Any parameter enclosed by < > is a mandatory field.
- Any parameter enclosed by [ ] is an optional field.
- <Commands> cannot co-exist with any [Options].

Error Code Definition

CODE	Definition
0xB09B	Error: Password Retry count exceeded
0xB0E0	Error: Unknown command or option.
0xB0E1	Error: AMI BIOS Guard feature disabled. Please use AFU to flash or enable the BIOS Guard feature in BIOS Setup.
0xB0E2	Error: Tool does not support this BIOS Guard flash interface.
0xB0E3	Error: Load firmware image fail.
0xB0E4	Error: Secure Flash ROM verify fail.
0xB0E5	Error: ME flash not support.
0xB0E6	Error: Runtime flash fail.
0xB0E7	Error: Runtime flash fail.
0xB0E0	Error: Runtime flash get status fail.
0x000E	Error: Kernel source files cannot be found.
0x000F	Error: Unable to make kernel driver
0x0010	Error: Unable to load driver.
0x0011	Error: Unable to unload driver.

Linux Pre-Requisites

1. Log in Linux as root otherwise use sudo (if permitted).

2. The compiler suite (gcc) must be installed. If these packages are not installed, the driver CANNOT be built.

3. For most of the distributions, BGT will generate driver without any notification, if it doesn't exist you need to install kernel sources. Also if Initmem fails, Please follow point 4.

4. Kernel sources must be installed, *CONFIGURED*, and then compiled. Following are steps to do this:

   a. Find Running Kernel's Configuration File:

      To configure the sources, simply change to the kernel source directory (typically **/lib/modules/$(uname -r)/build**). If it doesn't exist, you need to install kernel source.

      Typically, the reference configuration for the kernel can be found in the /boot directory with filename '**.config**', '**kernel.config**', or '**vmlinux-2.4.18-3.config**'. Type '**uname -a**' and use the configuration filename that best matches the output from '**uname -a**'. Also, check for **/dev/mem** directory existence. If it doesn't exist, you need to install kernel sources.

      Normally it comes with the installation unless if the option is deselected.

      On some distributions Red Hat, for instance, there is a config directory under **/lib/modules/$(uname -r)/build**.

      Copy this configuration file into the root of the Linux kernel source tree (usually it is **/lib/modules/$(uname -r)/build)**. This file must be renamed to "**.config**"(dot config).

   b. Make Your AMI Flash Driver (**amifldrv_mod.o**):

      For most distribution, the command to build the driver is:

```
BGTLNX_32 /MAKEDRV
                Or
BGTLNX_64 /MAKEDRV
```

If your Linux's kernel source tree is under **/lib/modules/$(uname -r)/build**, instead of being in the default path '**/lib/modules/$(uname -r)/build**', then add a KERNEL flag:

```
BGTLNX_32 /MAKEDRV KERNEL=/lib/modules/$(uname -r)/build
                              Or
BGTLNX_64 /MAKEDRV KERNEL=/lib/modules/$(uname -r)/build
```

If KERNEL is omitted, the default path is /lib/modules/$(uname -r)/build.
This should work for MOST distributions.

c. Make Your AMI Flash Driver from driver source files (amifldrv_mod.o):

Using command /GENDRV, it will generate driver source files to a specific directory.

```
BGTLNX_32 /GENDRV [Option 1] [Option 2]
                    Or
BGTLNX_64 /GENDRV [Option 1] [Option 2]
```

Where,
[Option 1]: Specific kernel source 'KERNEL=XXXX' same as the /MAKEDRV
[Option 2]: Specific output directory 'OUTPUT=XXXX'

Generate files as outlined below:

File Name	Description
amiwrap.c	Driver source code.
amiwrap.h	Driver header.
amifldrv.o_shipped	Object file for the driver.
Makefile	Makefile

For most distribution, the command to build the driver is: make.

If your Linux's kernel source tree is under **/lib/modules/$(uname -r)/build**, instead of being in the default path '**/lib/modules/$(uname -r)/build**', then add a KERNEL flag:

make KERNEL=/lib/modules/$(uname -r)/build

If KERNEL is omitted, the default is **/lib/modules/$(uname -r)/build.**
This should work for MOST distributions.

d. Check Your Build:
Check the version of running Linux kernel with '**uname -r**'.
Check the version of **amifldrv_mod.o** with '**modinfo amifdrv_mod.o**'.
If they mismatch, you will need to select the correct configuration
File (.config), rebuild your kernel and then rebuild your driver as described in steps a, b, c, and d.

5. The Linux driver's case:

	Secure Boot Enabled	Secure Boot Disabled
WSMT is supported	Need Driver	No Need Driver
Can access file path:/dev/mem	Need Driver	No Need Driver
Run Time Memory Hole support	Need Driver	No Need Driver

Signing Driver and Enrolling Public Key to the System

The following prerequisites are needed on the build system to sign the driver:

1. Login to Linux OS as root otherwise use sudo.
2. The compiler suite (gcc) must be installed. If it's not installed, the BGT driver cannot be built.
3. OpenSSL: Needed to generate cryptographic keys. OpenSSL tool can be downloaded from https://www.openssl.org
4. Perl interpreter: Needed to run the signing script. Perl tool can be downloaded from https://www.perl.org

Follow the below steps to sign the driver:

1. Boot to Linux OS.
2. Generate a Public and Private key pair using below openssl command: > openssl req -x509 -new -nodes -utf8 -sha256 -days 36500 -batch -config configuration_file.config -outform DER -out public_key.der -keyout private_key.priv

   Note: The configuration file configuration_file.config must be created with the required information before running the command. A sample configuration file is shown below. The values in <> must be filled with actual values.

   configuration_file.config:

```
[ req ]
default_bits = 4096
distinguished_name = req_distinguished_name
prompt = no
string_mask = utf8only
x509_extensions = myexts

[ req_distinguished_name ]
O = <organization_name>
CN = <organization_name> Signing Key
emailAddress = <email_address>

[ myexts ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
```

3. Build BGT driver using below command. The driver will be generated in the current directory with name amifldrv_mod.o.

```
> BGTLNX_64 /MAKEDRV
```

4. Execute below command to sign driver with the key generated in step 2.

```
> perl /usr/src/kernels/$(uname -r)/scripts/sign-file sha256 private_key.priv
public_key.der amifldrv_mod.o
```

Or

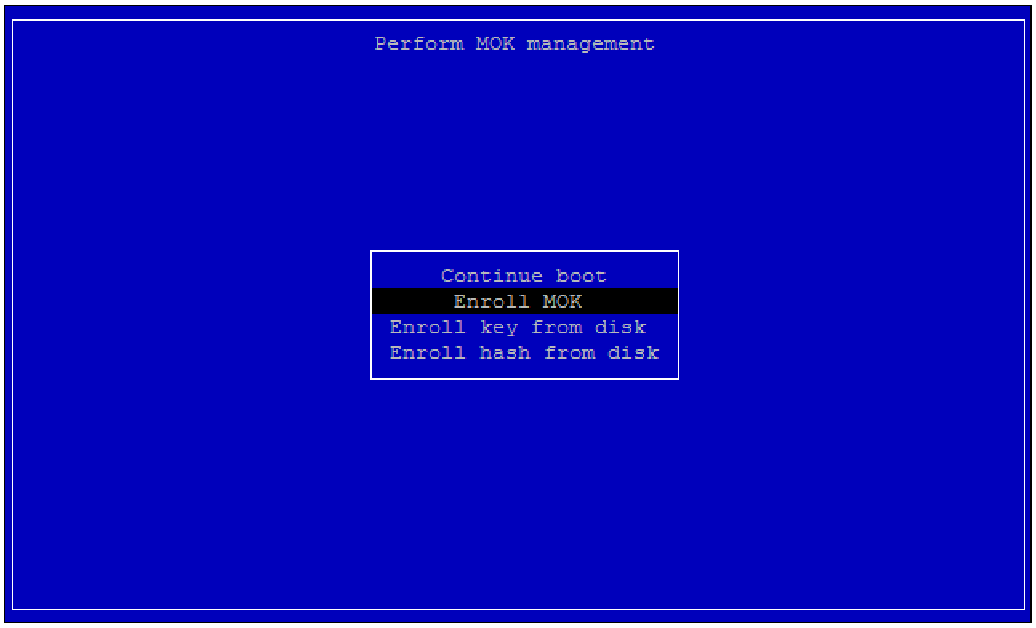
```
> /usr/src/kernels/$(uname -r)/scripts/sign-file sha256 private_key.priv public_key.der
amifldrv_mod.o
```

5. Request addition of a public key to MOK list using mokutil. The command will prompt a password which will be needed during public key enrollment in next step.

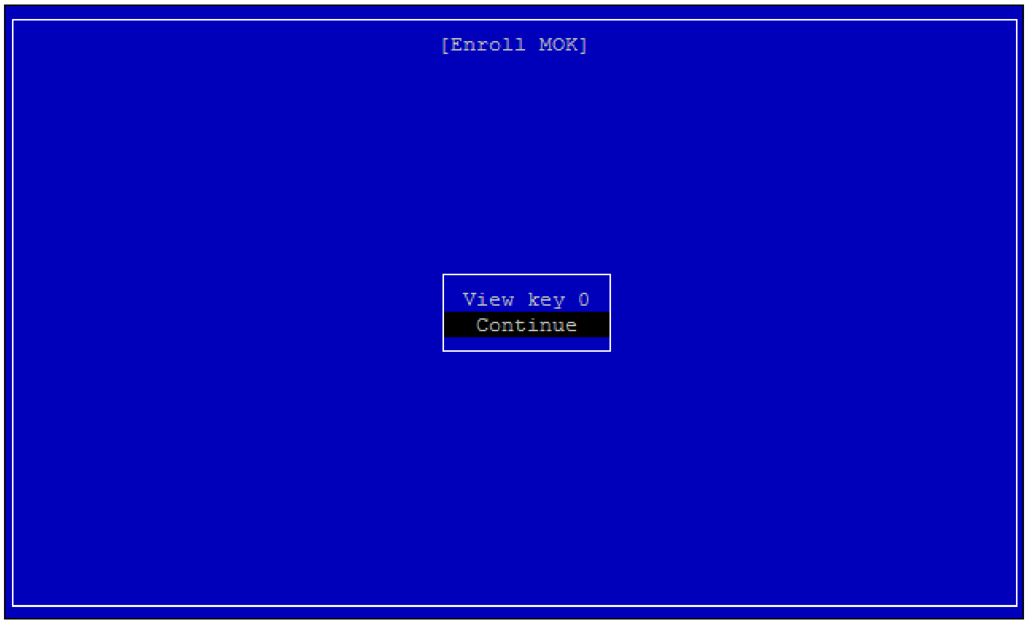
```
> mokutil --import public_key.der
```

6. Reboot the system which will launch MOK manager application to complete public key enrollment.

6-1. Select Enroll MOK.



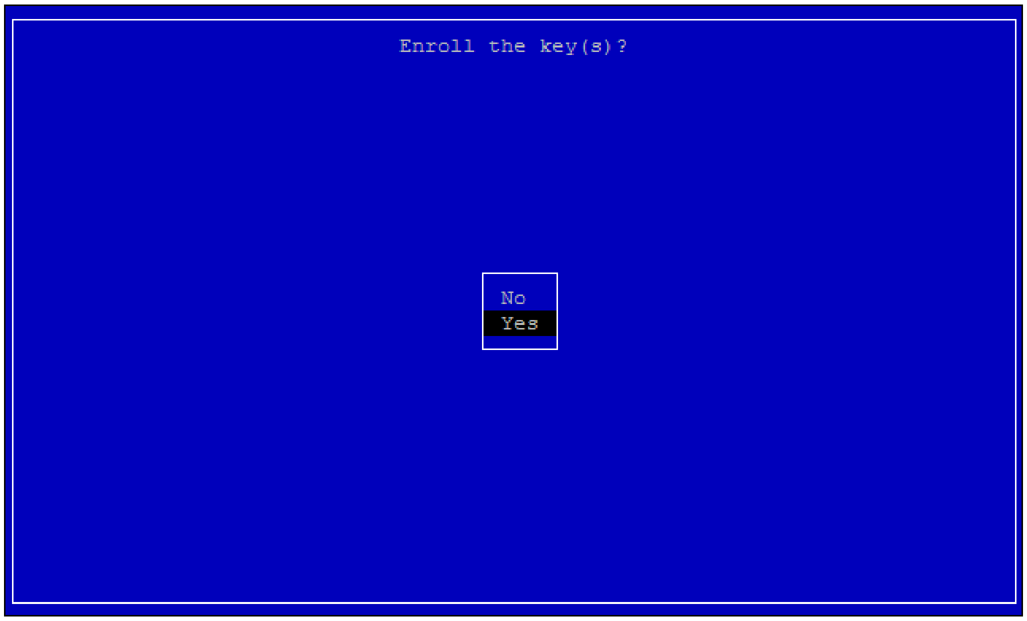
6-2. Select Continue.



```
[Enroll MOK]

View key 0
Continue
```

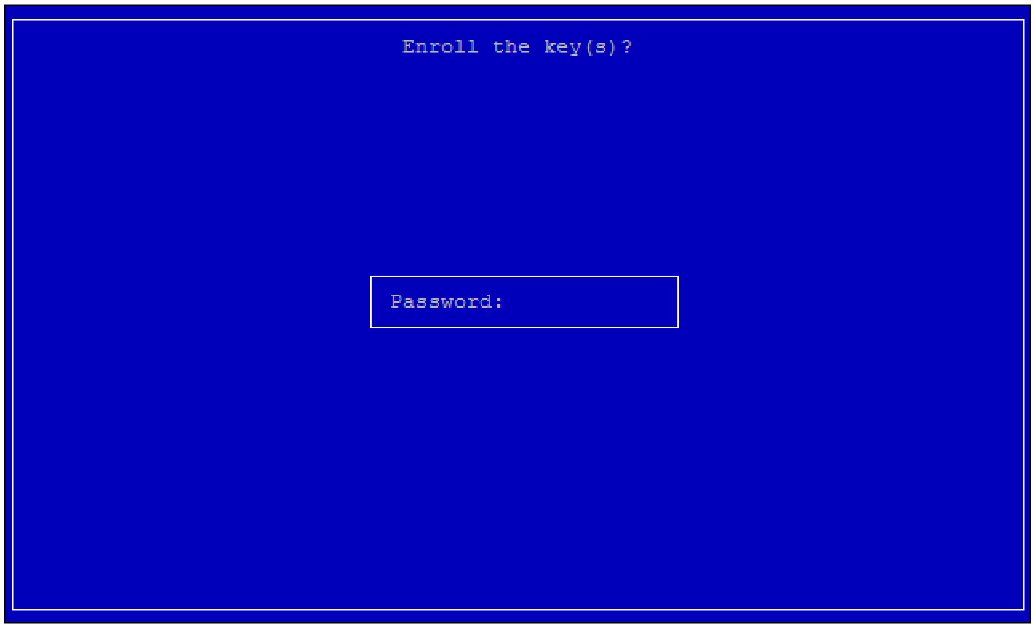
6-3. Select Yes.



```
Enroll the key(s)?

No
Yes
```

6-4. Input step 5 password.



7. Once the public key enrollment is done, Boot to OS and execute below command to ensure the newly added key is available in system key ring.

```
> keyctl list %:.system_keyring
Or
> keyctl list %:.builtin_trusted_keys
```

8. Install signed driver using insmod command.

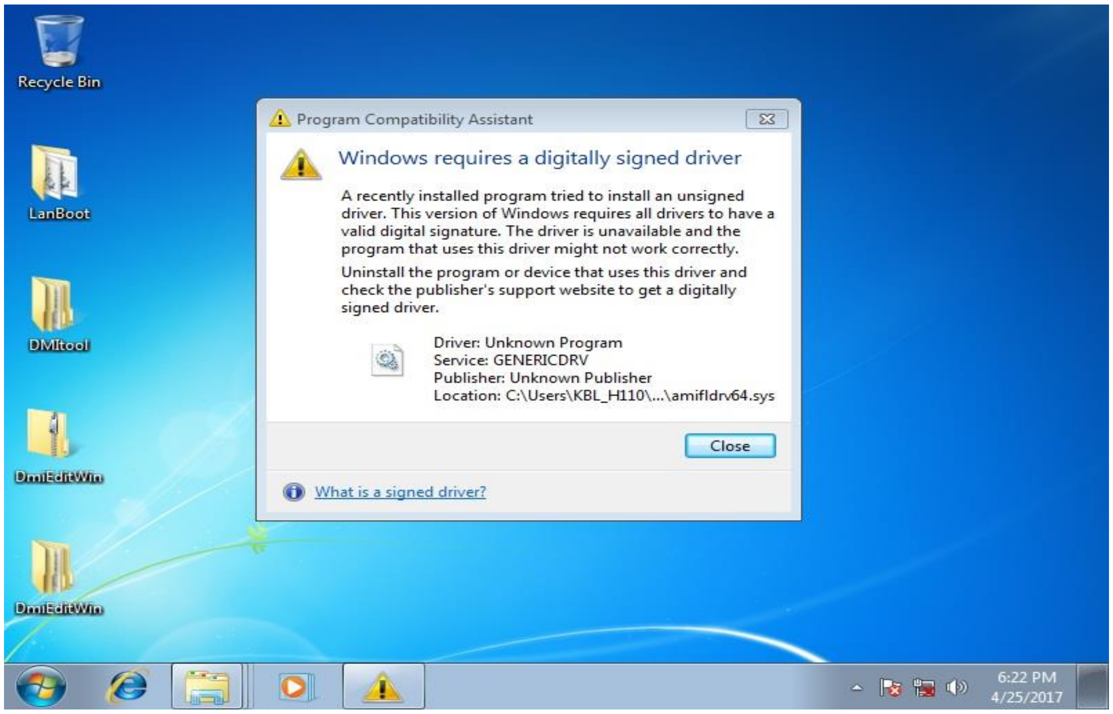
```
> insmod amifldrv_mod.o
```

9. Ensure it is loaded successfully using lsmod command.

Reference: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Kernel_Administration_Guide/sect-signing-kernel-modules-for-secure-boot.html

FAQ

Windows requires a digitally signed driver



Windows requires a digitally signed driver

A recently installed program tried to install an unsigned driver. This version of Windows requires all drivers to have a valid digital signature. The driver is unavailable and the program that uses this driver might not work correctly.

Uninstall the program or device that uses this driver and check the publisher's support website to get a digitally signed driver.

Driver: Unknown Program
Service: GENERICDRV
Publisher: Unknown Publisher
Location: C:\Users\KBL_H110\...\amifldrv64.sys

This issue is resolved by a security fix provided by MS. KB3033929 resolves this issue. The certificate used to sign the driver is higher security and older versions of Win7 don't support it.