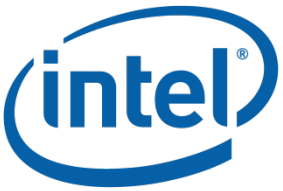

Getting Started with UEFI HTTPS Boot on EDK II

July 2016

Revision 0.7

WHITEPAPER

Wu Jiaxin

Contents

Figures

Tables

Revision History

Revision Number	Description	Revision Date
<0.1>	Initial release.	March 2016
<0.2>	Add UEFI Client Certificate Configuration	May 2016
<0.3>	Simplify the topology and configuration.	May 2016
<0.4>	Refine the content	May 2016
<0.5>	Minor content revisions	May 2016
<0.6>	Updated figure references & document formatting. Text cleanup for information related to IPv4 & IPv6 configuration (multiple sections).	June 2016
<0.7>	Modified document title. Added information on openSUSE implementation of HTTP Boot.	July 2016

1 Introduction

1.1 Overview

HTTP over TLS (HTTPS) boot is a standard implementation for securely booting using the Unified Extensible Firmware Interface (UEFI) over a network device. HTTPS Boot is especially important for clients using potentially insecure networks outside of corporate infrastructure. Security for UEFI HTTPS Boot is provided by the underlying Transport Layer Security (TLS).

This document assumes that the reader is familiar with the EDK II HTTP Boot Getting Started Guide available at tianocore.org. For information on configuring a HTTP Boot server, please refer to the UEFI HTTP Boot with OVMF help page available at opensuse.org.

1.2 Additional Protocols

Note: *All protocols introduced in the EDK II HTTP Boot Getting Started Guide are necessary.*

The following new protocols are related to HTTPS Boot:

- EFI_TLS_SERVICE_BINDING_PROTOCOL
- EFI_TLS_PROTOCOL
- EFI_TLS_CONFIGURATION_PROTOCOL

1.3 Additional Modules

Note: *All modules introduced in the EDK II HTTP Boot Getting Started Guide are necessary.*

The following new TLS modules are also required by HTTPS boot:

- OpenSSL Crypto module
  `CryptoPkg/Library/OpensslLib/OpensslLib.inf`
- OpenSSL TLS module
  `CryptoPkg/Library/OpensslLib/OpensslTlsLib.inf`
- Base Crypto Library
  `CryptoPkg/Library/BaseCryptLib/BaseCryptLib.inf`
- TLS Library
  `CryptoPkg/Library/TlsLib/TlsLib.inf`
- TLS Driver
  `NetworkPkg/TlsDxe/TlsDxe.inf`
- TLS Authentication Config Driver
  `NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf`

2 HTTPS Authentication

Figure 1 shows the regular HTTPS authentication mechanism for both client and server. Leveraging an asymmetric crypto system, the client and server can be authenticated by each other. The steps for mutual authentication are as follows:

1. Server and Client request the corresponding asymmetric key pair from the Certification Authority (CA). Both requested certificates can be verified by the CA.
2. The CA distributes the key pair (servercert/serverkey) and its own certificate (rootcert) to the Server. The distributed certificate (servercert) has been signed by its rootkey.
3. The CA distributes the key pair (clientcert/clientkey) and its own certificate (rootcert) to the Client. The distributed certificate (clientcert) has been signed by its rootkey.
4. Both Server and Client present their own certificate to each other for mutual authentication.
5. When the Server receives the Client certificate (clientcert) the certificate will be verified by rootcert, since it has been signed with the rootkey (and vice versa).

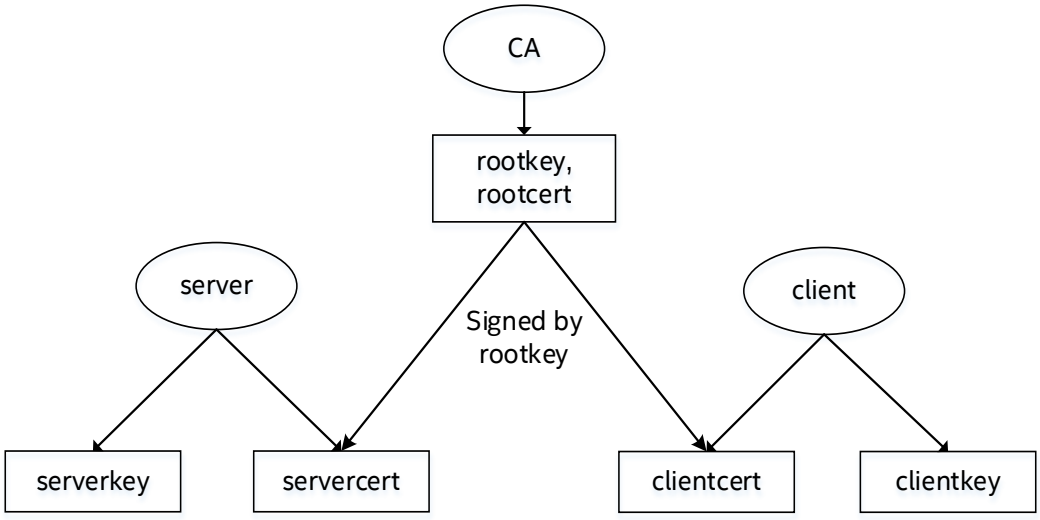

Figure 1: Authentication Mechanism

2.1 TLS Authentication Modes

TLS supports three authentication modes:

1. Two-way authentication: authentication of both parties. In this mode, both server and client will be authenticated.

2. One-way authentication: server authentication with an unauthenticated client. That means only the server is authenticated by the client, and the client won't be authenticated by the server.

3. Total anonymity: the server and client won't authenticate each other.

Table 1 shows the certificate requirement in UEFI client and HTTPS server for each authentication modes support.

Mode \ Part	Authentication of both parties	Server authentication with an unauthenticated client	Total anonymity
HTTPS Client	rootcert, clientcert, clientkey	rootcert	NULL
HTTPS Server	rootcert, servercert, serverkey	servercert, serverkey	servercert, serverkey

Table 1: Certificate Requirement

2.2 Self-Generated Certificate

This example shows how vendors can generate custom certificates for HTTPS Boot:

1. Install openssl (example for Ubuntu: **`sudo apt-get install openssl`**)
2. Create a self-signed CA Certificate:

```
openssl req -new -md5 -keyout rootkey.pem -out rootreq.pem -days
3650

openssl x509 -req -in rootreq.pem -md5 -signkey rootkey.pem -out
rootcert.pem -days 3650

cat rootcert.pem rootkey.pem > root.pem
```

3. Create a server certificate signed by the CA certificate:

```
openssl req -new -md5 -keyout serverkey.pem -out serverreq.pem -
days 3650

openssl x509 -req -in serverreq.pem -md5 -CA root.pem -CAkey
root.pem  -CAcreateserial -out servercert.pem -days 3650

cat servercert.pem serverkey.pem root.pem > server.pem

openssl pkcs12 -export -in server.pem  -out server.pfx
```

Note: *The* ***`.pem`*** *file is encoded as BASE64, but only PKCS12 format key can be used when booting to a Microsoft Windows server. This requires the last step in process above, converting* ***`server.pem`*** *to* ***`server.pfx`****.*

4. Create a client certificate signed by the CA certificate:

```
openssl req -new -md5 -keyout clientkey.pem -out clientreq.pem -
days 3650

openssl x509 -req -in clientreq.pem -md5 -CA root.pem -CAkey
root.pem -CAcreateserial -out clientcert.pem -days 3650

cat clientcert.pem clientkey.pem root.pem > client.pem
```

Using the steps above, the required key pairs are generated as shown in Table 2:

CA	`rootkey.pem, rootcert.pem, root.pem`
Server	`serverkey.pem, servercert.pem, server.pem, server.pfx`
Client	`clientkey.pem, clientcert.pem, client.pem`

Table 2: Key Pair

3 Start Guide

This section assumes the reader has installed EDK II and can build and run the NT32 simulator. This guide gives instructions on how to set up a UEFI HTTPS Boot environment for both IPv4 and IPv6 network environments.

3.1 Configure Server and Build Client

There are three indispensable role for a server in the UEFI HTTPS boot environment: DHCP server, DNS server and HTTPS server. Depending on server requirements, two test-bed solutions are presented for reference: one simple approach for IPv4, and an advanced solution using IPv6. Users can select the proper scenario based on individual requirements. Self-generated certificates from Table 2 are used for HTTPS one-way authentication.

3.1.1 Solution for IPv4

The solution documented in this section uses a single server for the DHCP, DNS and HTTPS functions. This is considered the simplest server configuration for UEFI HTTPS Boot.

3.1.1.1 Network topology for IPv4

This example is based on Microsoft Windows Server 2012 R2. Internet Information Services (IIS) are used to configure HTTPS server. The server and NT32 simulator use the same IPv4 subnet (192.168.10.0) as shown in Figure 2.

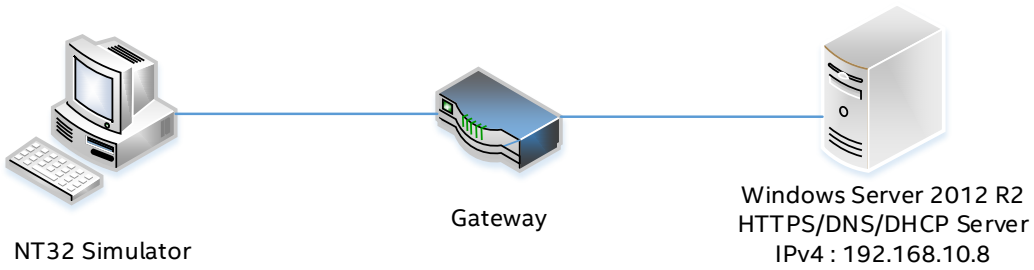

Figure 2: HTTPS Boot, IPv4 Configuration

3.1.1.2 Configure DHCPv4 server

The steps to configure a DHCPv4 server are as follows:

1. Add a DHCP service in Windows Server manager.

2. Right click ‘IPv4 – New Scope’ to create a new scope option for IPv4 including the scope name, address range, and IP address lease duration. See Figure 3 for details.

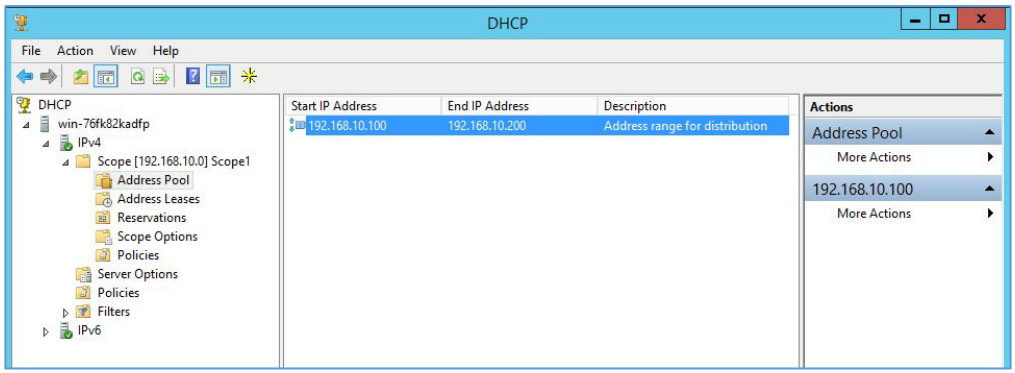

Figure 3: DHCPv4 Server Scope

3. Right click ‘Server Options – Configure Options…’ to configure IPv4 options including option 6, 60 and 67. These options must be configured for proper functionality. After the configuration, the options should appear as shown in Figure 4.
    a. Option 6 indicates the DNS server address.
    b. Option 60 defines the vendor Class ID. The value should be set to ‘HTTPClient’.
    c. Option 67 contains the corresponding boot file URI.

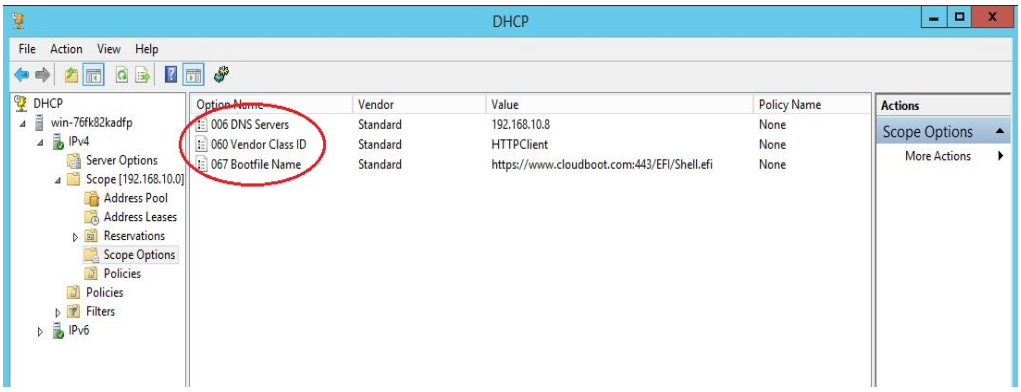

Figure 4: DHCPv4 Server Options

4. Right click the DHCP server name and select the ‘All Tasks – Restart’ option to restart the DHCPv4 service.

3.1.1.3 Configure DNSv4 Server

The steps to configure the DNSv4 server are as follows:

1. Add the DNS service in Windows Server Manager.
2. Add a new forward lookup zone named ‘cloudboot.com’.

3. Add a new Host “www” for IPv4 (192.168.10.8). See Figure 5 for reference.

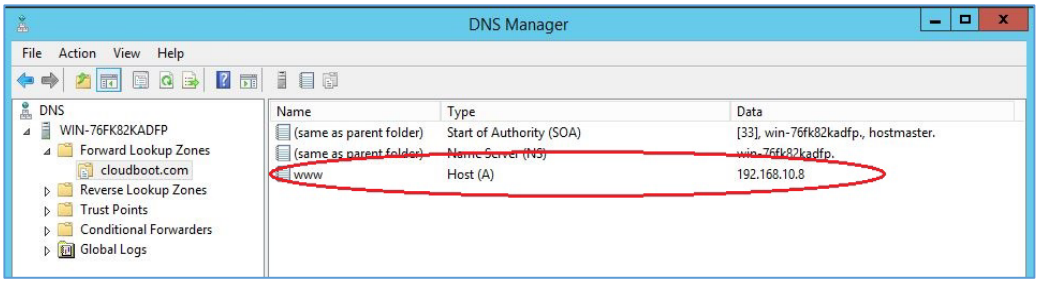

Figure 5: Configure New Host for IPv4

4. Right click the DHCP server name and select the ‘All Tasks – Restart’ option to restart the DHCPv4 service.

3.1.1.4 Configure HTTPS Server for IPv4

The steps to configure the HTTPS server are as follows:

1. Enable the Internet Information Services (IIS) feature in Windows Server manager, based on installation steps available here: http://www.iis.net/learn/install/installing-iis-85/installing-iis-85-on-windows-server-2012-r2.

2. Open the Internet Information Services (IIS) Manager, and add a new MIME type for the resources required by the HTTPS server. In this example, the client will boot to a UEFI Shell image provided by the server. This requires addition of the `.efi` file type. Figure 6 and Figure 7 show the detailed steps.

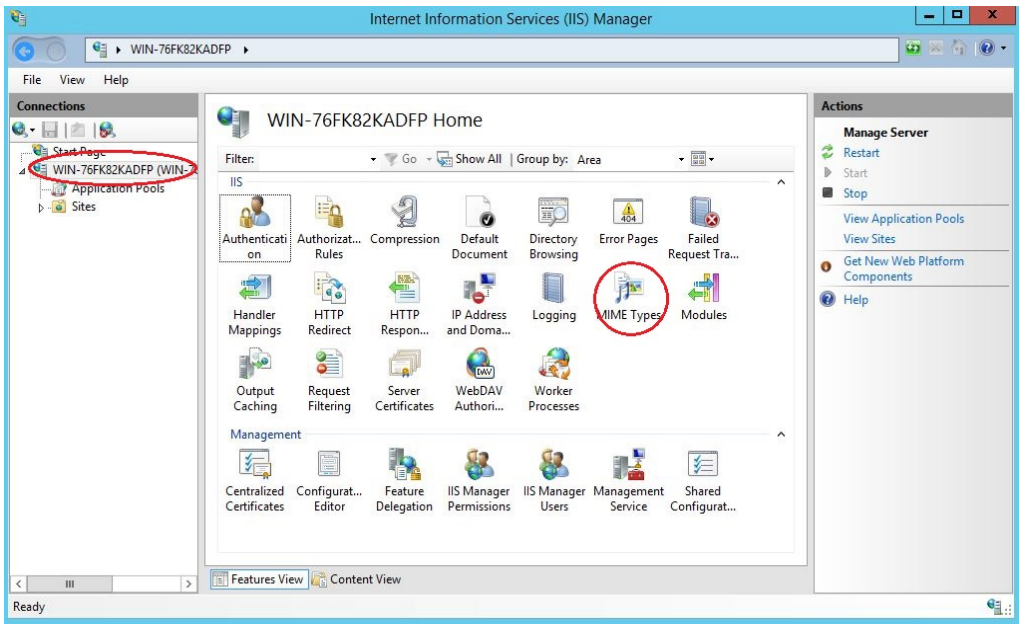

Figure 6: Add MIME Type

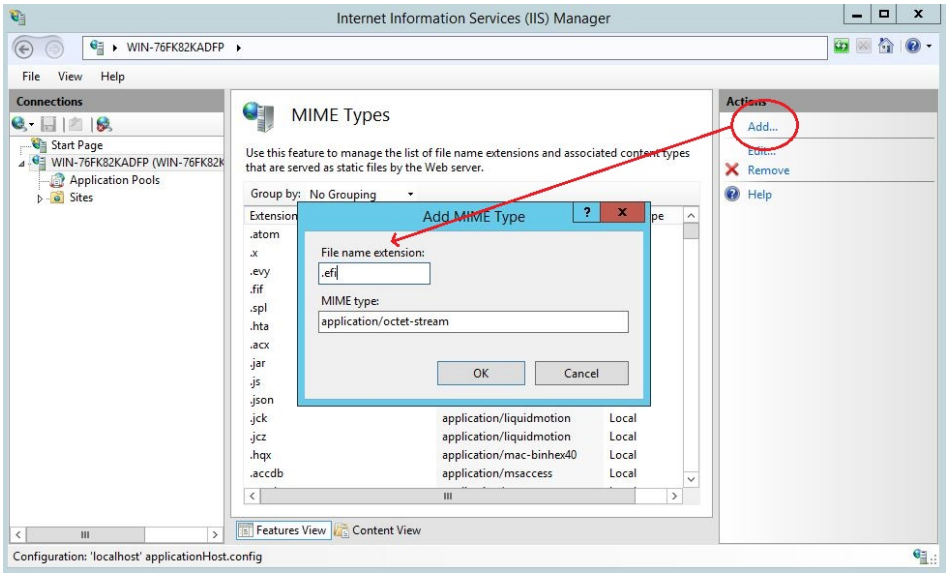

Figure 7: Add a New MIME Type to IIS

3. Enroll the Server key pair (**`server.pfx`**) in 'Server Certificates'. Refer to Figure 8 and Figure 9 for details.

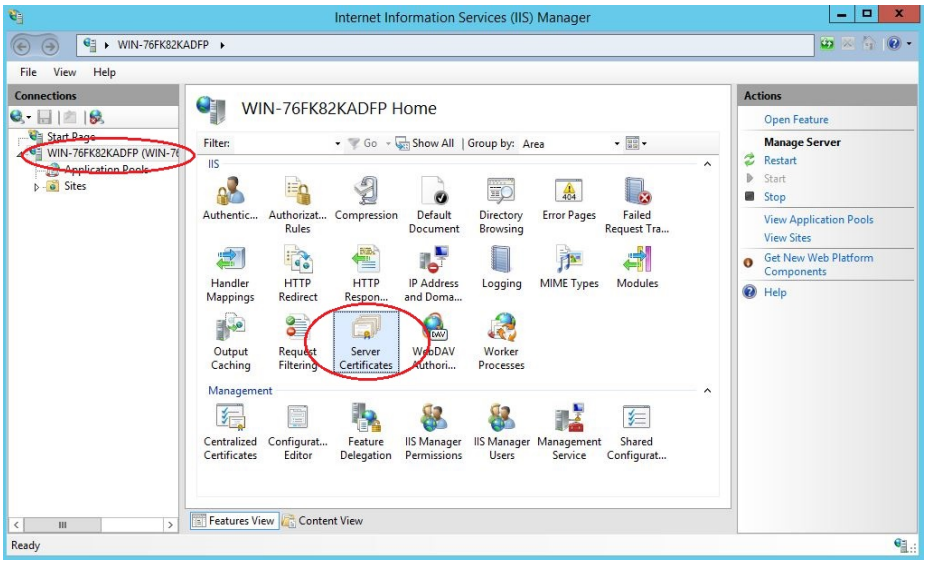

Figure 8: Add Server Certificates

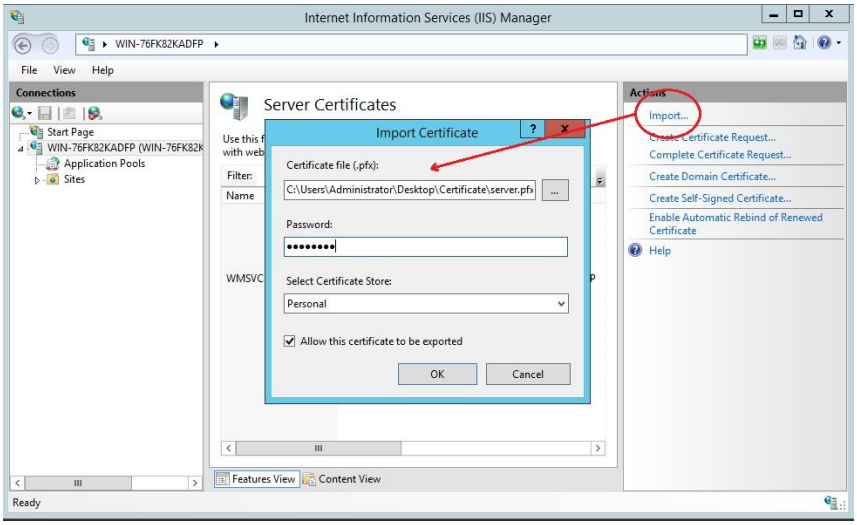

Figure 9: Enroll a Certificate for the HTTPS Server

4. Right-click on 'Sites – Add Website' to create a new website for the HTTPS server. The areas highlighted in Figure 10 are required fields. The 'Physical path' is the default root path for the website. The 'SSL certificate' is the server key pair, which was enrolled in Step 3. The binding type is 'https' and the binding port value is '443'.

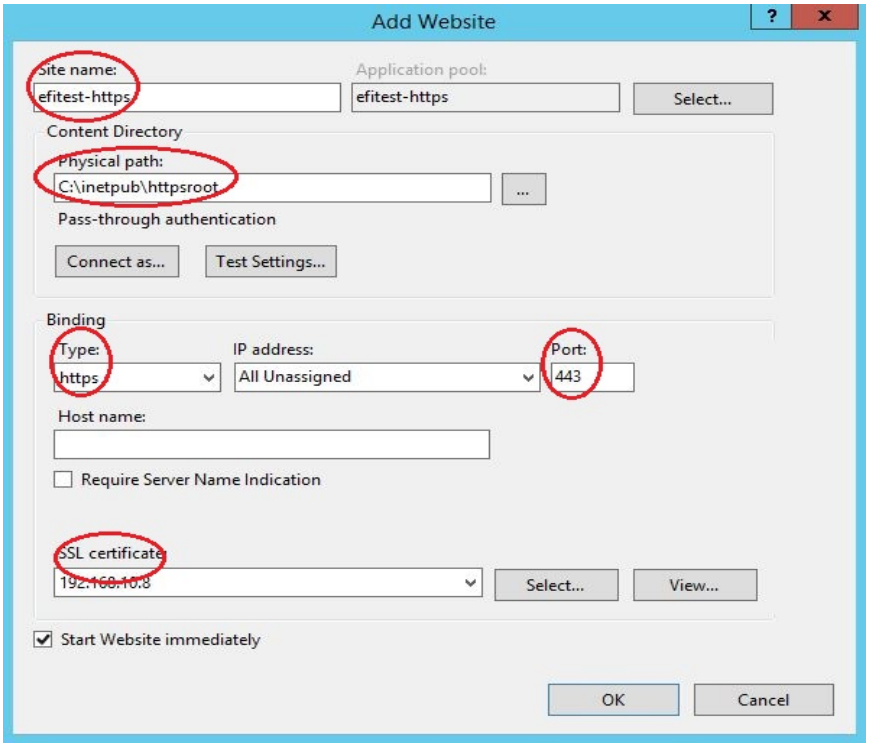

Figure 10: Create a New Website for the HTTPS Server

5. Create an 'EFI' folder in default root path, which was configured in Step 4. Copy the UEFI Shell binary that matches your firmware configuration into this folder. The UEFI Shell binary is in the **`ShellBinPkg`** package on EDK II:
   https://github.com/tianocore/edk2/tree/master/ShellBinPkg
   The file should be renamed **`Shell.efi`** to match the configuration in DHCP option 67. This sets the UEFI Shell boot path as https://www.cloudboot.com:443/EFI/Shell.efi.

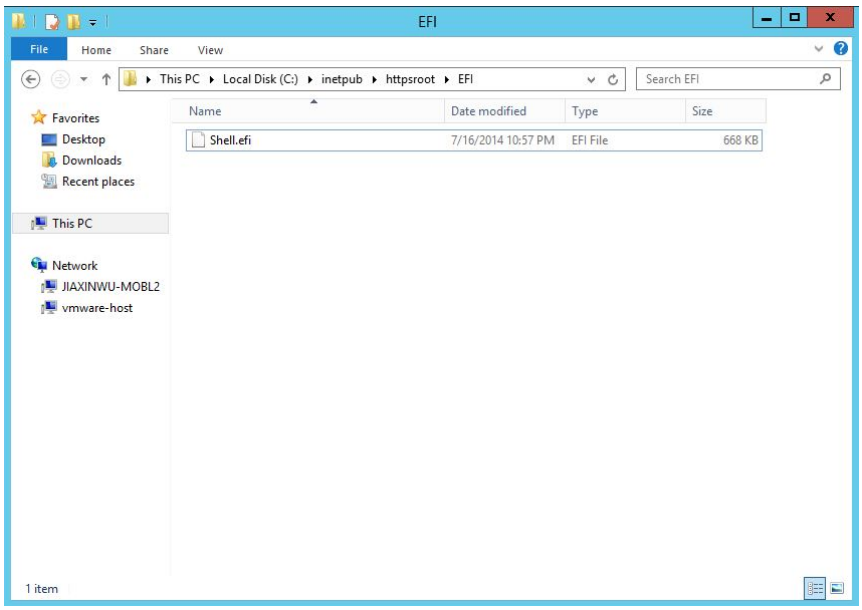

Figure 11: The UEFI Shell file, as viewed in IIS

Note: *The NT32 Simulator uses the IA32 UEFI Shell binary, while most production systems require the x64 UEFI Shell to match the UEFI firmware configuration. This depends on your platform firmware configuration.*

3.1.1.5 Enable NT32 Simulator for IPv4

To enable the UEFI HTTPSv4 Boot feature, the latest EDKII network stack (IPv4) must be built in your system firmware.

Modules in DSC file

The following libraries and drivers are required by HTTPSv4 boot:

Add the following libraries to the **`LibraryClasses`** section:

```
DpcLib|MdeModulePkg/Library/DxeDpcLib/DxeDpcLib.inf
NetLib|MdeModulePkg/Library/DxeNetLib/DxeNetLib.inf
IpIoLib|MdeModulePkg/Library/DxeIpIoLib/DxeIpIoLib.inf
UdpIoLib|MdeModulePkg/Library/DxeUdpIoLib/DxeUdpIoLib.inf
TcpIoLib|MdeModulePkg/Library/DxeTcpIoLib/DxeTcpIoLib.inf
HttpLib|MdeModulePkg/Library/DxeHttpLib/DxeHttpLib.inf
OpensslLib|CryptoPkg/Library/OpensslLib/OpensslLib.inf
OpensslTlsLib|CryptoPkg/Library/OpensslLib/OpensslTlsLib.inf
BaseCryptLib|CryptoPkg/Library/BaseCryptLib/BaseCryptLib.inf
TlsLib|CryptoPkg/Library/TlsLib/TlsLib.inf
```

Add the following drivers to the **Components** section:

```
MdeModulePkg/Universal/Network/DpcDxe/DpcDxe.inf
MdeModulePkg/Universal/Network/SnpDxe/SnpDxe.inf
MdeModulePkg/Universal/Network/MnpDxe/MnpDxe.inf
MdeModulePkg/Universal/Network/ArpDxe/ArpDxe.inf
MdeModulePkg/Universal/Network/Ip4Dxe/Ip4Dxe.inf
MdeModulePkg/Universal/Network/Tcp4Dxe/Tcp4Dxe.inf
MdeModulePkg/Universal/Network/Udp4Dxe/Udp4Dxe.inf
MdeModulePkg/Universal/Network/Dhcp4Dxe/Dhcp4Dxe.inf
NetworkPkg/HttpDxe/HttpDxe.inf
NetworkPkg/HttpBootDxe/HttpBootDxe.inf
NetworkPkg/HttpUtilitiesDxe/HttpUtilitiesDxe.inf
NetworkPkg/DnsDxe/DnsDxe.inf
NetworkPkg/TlsDxe/TlsDxe.inf
NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf
```

Note: *The network controller's UNDI driver also needs to be in the list of platform files*

Modules in FDF file

The following drivers should be added to the **FV** section for HTTPSv4 boot:

```
INF MdeModulePkg/Universal/Network/DpcDxe/DpcDxe.inf
INF MdeModulePkg/Universal/Network/SnpDxe/SnpDxe.inf
INF MdeModulePkg/Universal/Network/MnpDxe/MnpDxe.inf
INF MdeModulePkg/Universal/Network/ArpDxe/ArpDxe.inf
INF MdeModulePkg/Universal/Network/Ip4Dxe/Ip4Dxe.inf
INF MdeModulePkg/Universal/Network/Tcp4Dxe/Tcp4Dxe.inf
INF MdeModulePkg/Universal/Network/Udp4Dxe/Udp4Dxe.inf
INF MdeModulePkg/Universal/Network/Dhcp4Dxe/Dhcp4Dxe.inf
INF NetworkPkg/HttpDxe/HttpDxe.inf
INF NetworkPkg/HttpBootDxe/HttpBootDxe.inf
INF NetworkPkg/HttpUtilitiesDxe/HttpUtilitiesDxe.inf
INF NetworkPkg/DnsDxe/DnsDxe.inf
INF NetworkPkg/TlsDxe/TlsDxe.inf
INF NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf
```

Build the NT32 Simulator

The following command is used to build NT32 simulator:

```
build -a IA32 –t VS2013x86 -p Nt32pkg\Nt32Pkg.dsc
```

3.1.2 Solution for IPv6

For IPv6, the DHCP, DNS and HTTPS server are deployed on different servers. This solution provides a more flexible configuration for the DHCP server, DNS server and HTTPS Server.

3.1.2.1 Network Topology for IPv6

In this example, the DHCP server is deployed on Ubuntu 15.10. The DNS server is deployed on Windows Server 2012 R2, and the HTTPS Server is deployed on another instance of Windows Server 2012 R2. IIS is used to configure HTTPS server. The servers and NT32 simulator are located on the same IPv6 subnet (**2000:bbbb::/64**).

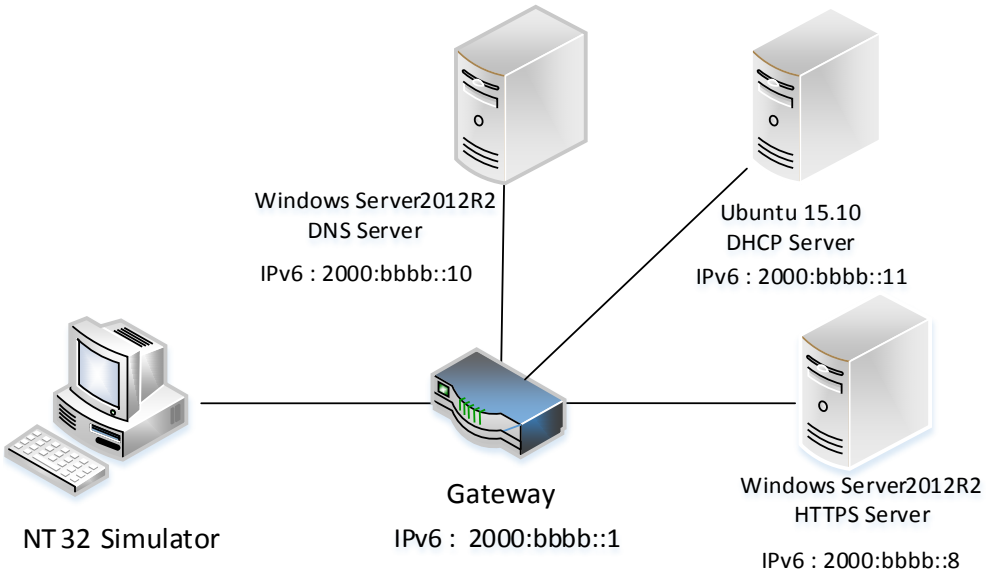

Figure 12: HTTPS boot, IPv6 Configuration

3.1.2.2 Configure the DHCPv6 Server

The steps to configure DHCPv6 on an Ubuntu 15.10 server are shown as follows:

1. Install the DHCP server: **`sudo apt-get install isc-dhcp-server`**
2. Edit **`/etc/dhcp/dhcpd6.conf`** as shown below

Note: *If there is no dhcpd6.conf file in /etc/dhcp/, create it first.*

```
default-lease-time 600;
max-lease-time 7200;
log-facility local7;
#option definitions common to all supported networks…
option dhcp6.vendor-class code 16 = { integer 32, integer
16, string};
option dhcp6.bootfile-url code 59 = string;
subnet6 2000:bbbb::/64 {
        #Range for clients
        range6 2000:bbbb::100 2000:bbbb::ffff;
        option dhcp6.domain-search “cloudboot.com”;
        option dhcp6.name-servers 2000:bbbb::10;
        option dhcp6.vendor-class 0 0 “HTTPClient”;
        option dhcp6.bootfile-url
```

```
        "https://www.cloudboot.com:443/EFI/Shell.efi";
        }
```

3. Configure the server to listen for DHCP requests on the correct network interface. This example assumes **`eth0`** is the primary interface. Edit the **`/etc/default/isc-dhcp-server`** file to configure **`INTERFACE = "eth0";`**
4. Restart the DHCPv6 service: **`sudo service isc-dhcp-server6 restart`**

3.1.2.3 Configure DNSv6 Server

The steps to configure DNSv6 for Microsoft Windows Server 2002 R2 are as follows:

1. Add the DNS service in Windows Server Manager.
2. Add a new forward lookup zone 'cloudboot.com' (see Figure 13).

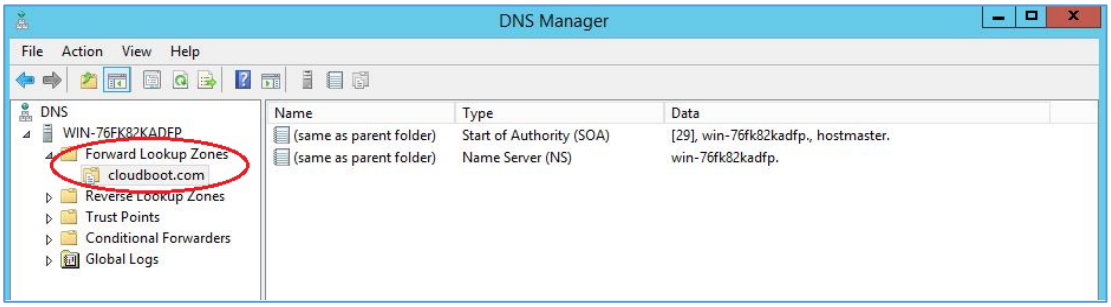

Figure 13: Configure Forward Lookup Zone for IPv6

3. Add a new Host "www" for IPv6 (**`2000:bbbb::8`**) as shown in Figure 14.

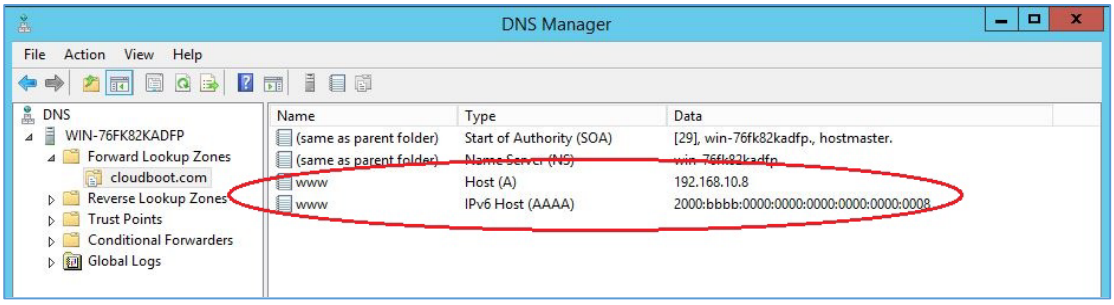

Figure 14: Configure New Host for IPv6

4. Right click the DNS server name and select the 'All Tasks – Restart' option to restart the DNSv6 service.

3.1.2.4 Configure HTTPS Server for IPv6

Please refer to Section 3.1.1.4, as this step is not dependent on IPv4 or IPv6.

3.1.2.5 Enable NT32 Simulator for IPv6

To enable the UEFI Boot feature for HTTPSv6, the latest EDKII network stack (IPv6) must be built in your system firmware.

Modules in DSC file

The following libraries and drivers are required by HTTPSv6 boot:

Add the following libraries to the **LibraryClasses** section:

```
DpcLib|MdeModulePkg/Library/DxeDpcLib/DxeDpcLib.inf
NetLib|MdeModulePkg/Library/DxeNetLib/DxeNetLib.inf
IpIoLib|MdeModulePkg/Library/DxeIpIoLib/DxeIpIoLib.inf
UdpIoLib|MdeModulePkg/Library/DxeUdpIoLib/DxeUdpIoLib.inf
TcpIoLib|MdeModulePkg/Library/DxeTcpIoLib/DxeTcpIoLib.inf
HttpLib|MdeModulePkg/Library/DxeHttpLib/DxeHttpLib.inf
OpensslLib|CryptoPkg/Library/OpensslLib/OpensslLib.inf
OpensslTlsLib|CryptoPkg/Library/OpensslLib/OpensslTlsLib.inf
BaseCryptLib|CryptoPkg/Library/BaseCryptLib/BaseCryptLib.inf
TlsLib|CryptoPkg/Library/TlsLib/TlsLib.inf
```

Add the following drivers to the **Components** section:

```
MdeModulePkg/Universal/Network/DpcDxe/DpcDxe.inf
MdeModulePkg/Universal/Network/SnpDxe/SnpDxe.inf
MdeModulePkg/Universal/Network/MnpDxe/MnpDxe.inf
NetworkPkg/Ip6Dxe/Ip6Dxe.inf
NetworkPkg/TcpDxe/TcpDxe.inf
NetworkPkg/Udp6Dxe/Udp6Dxe.inf
NetworkPkg/Dhcp6Dxe/Dhcp6Dxe.inf
NetworkPkg/HttpDxe/HttpDxe.inf
NetworkPkg/HttpBootDxe/HttpBootDxe.inf
NetworkPkg/HttpUtilitiesDxe/HttpUtilitiesDxe.inf
NetworkPkg/DnsDxe/DnsDxe.inf
NetworkPkg/TlsDxe/TlsDxe.inf
NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf
```

Note: *The network controller's UNDI driver also needs to be in the list of platform files*

Modules in FDF file

The following drivers are required in the **FV** section for HTTPSv6 boot:

```
INF MdeModulePkg/Universal/Network/DpcDxe/DpcDxe.inf
INF MdeModulePkg/Universal/Network/SnpDxe/SnpDxe.inf
INF MdeModulePkg/Universal/Network/MnpDxe/MnpDxe.inf
INF NetworkPkg/Ip6Dxe/Ip6Dxe.inf
INF NetworkPkg/TcpDxe/TcpDxe.inf
INF NetworkPkg/Udp6Dxe/Udp6Dxe.inf
INF NetworkPkg/Dhcp6Dxe/Dhcp6Dxe.inf
INF NetworkPkg/HttpDxe/HttpDxe.inf
INF NetworkPkg/HttpBootDxe/HttpBootDxe.inf
INF NetworkPkg/HttpUtilitiesDxe/HttpUtilitiesDxe.inf
INF NetworkPkg/DnsDxe/DnsDxe.inf
INF NetworkPkg/TlsDxe/TlsDxe.inf
INF NetworkPkg/TlsAuthConfigDxe/TlsAuthConfigDxe.inf
```

Build the NT32 Simulator

The following command is used to build NT32 simulator:

```
build -a IA32 –t VS2013x86 -p Nt32pkg\Nt32Pkg.dsc
```

3.2 Run HTTPS Boot

Currently the UEFI HTTPS Boot feature only supports server authentication with an unauthenticated client. To support this mode, the Server CA certificate (**rootcert.pem**) is required by the Client. A private variable is used to configure the CA certificate on the client system. The **EFI_SIGNATURE_LIST** format is used for this variable:
TlsCaCertificate, {0xfd2340D0, 0x3dab, 0x4349, {0xa6, 0xc7, 0x3b, 0x4f, 0x12, 0xb4, 0x8e, 0xae}}

3.2.1 Configure the Certificate

The server CA certificate must first be configured to enable UEFI HTTPS Boot. The **TlsAuthConfigDxe** driver provides a user interface to support the required certificate configuration. Figure 15 shows the UEFI Client configuration in Boot Manager.

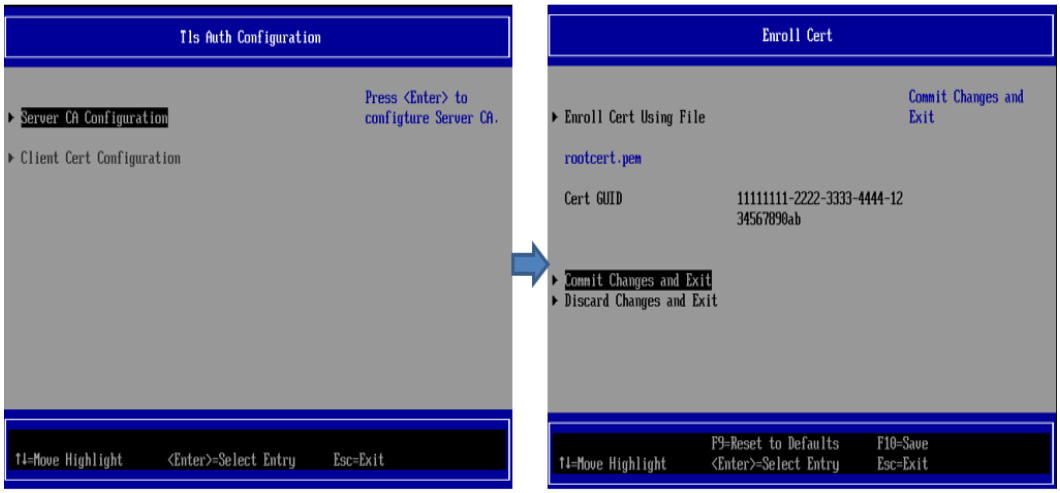

Figure 15: UEFI Client Certificate Configuration

3.2.2 Run HTTPS Boot on the UEFI Client

After the Server CA certificate (**rootcert.pem**) has been configured, the NT32 simulator can perform a HTTPS Boot. Start the NT32 simulator, enter Boot Manager, and select “UEFI HTTPv4” or “UEFI HTTPv6” depending on the server configuration (see Figure 16).

Figure 16: Select Boot Option

During UEFI HTTPS Boot, the **HttpDxe** driver consumes the **TlsDxe** driver. This boot process supports HTTP and HTTPS, depending on the URL. This allows **HttpDxe** to communicate with an HTTPS or HTTP server configuration. The example in this document loads a UEFI Shell image downloaded from the server. Figure 17 shows the result of a successful UEFI HTTPS Boot.

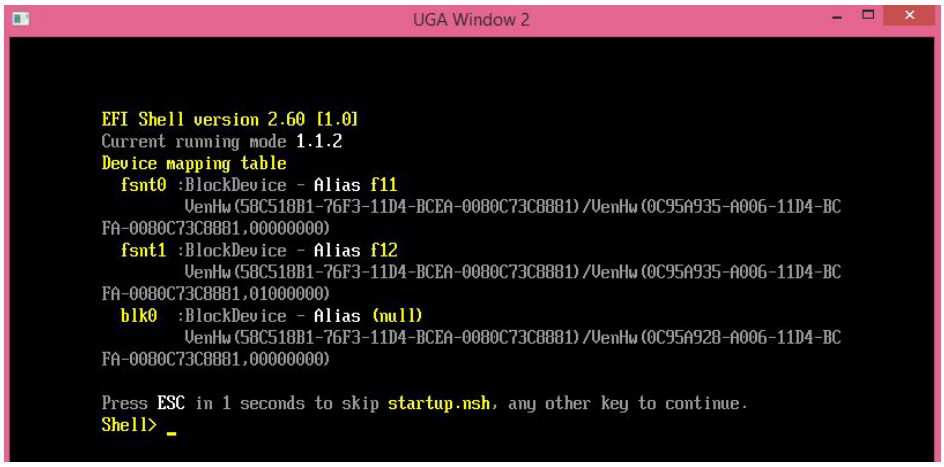

Figure 17: Boot the Downloaded UEFI Shell Image