HIRS .NET Provisioner Readme

Installation

The HIRS .NET provisioner is self-contained. No dependencies should be needed to run the program.

Windows

Double click the MSI to install it. This will install files to C:\Program Files (x86)\hirs.

Linux

Follow the instructions for your package manager. These will install files to /usr/share/hirs.

Standard Configuration

By default, the HIRS .NET Provisioner is configured to auto detect a discrete or firmware TPM. It will also use a bundled version of paccor to collect hardware information. You will need to make sure it is configured to talk to an ACA at a valid address.

Settings are captured in a JSON file in the installation directory called “appsettings.json”.

Set the server

a. Open appsettings.json.
b. Verify or change the value of **aca_address_port** to point to the server running the HIRS ACA.
    i. Use the format <scheme>://<host>:<port>

Usage

Once installed and configured, the executable is ready to be run.

Some general tips for ensuring a successful provision:

- Verify the ACA policy is appropriately set for your use case
- Upload Trusted Certificates and Platform Certificates to the ACA

Windows

```
tpm_aca_provision.exe
```

After the MSI installation, this command should be accessible from any powershell or command prompt.

Linux

```
/usr/share/hirs/tpm_aca_provision
```

If the TPM auto detection is turned off, additional command line options are necessary. See the Configuration Options sections for additional settings.

Appendix B of the HIRS ACA User's Guide has details about our provisioning process.

https://github.com/nsacyber/HIRS/blob/master/HIRS_AttestationCAPortal/src/main/webapp/docs/HIRS_ACA_UsersGuide_2.1.pdf

Configuration Options

All of these are JSON keys that can be used in the appsettings.json file.

auto_detect_tpm

If set to TRUE, the program will search for a TPM. First it will use an API offered by each OS to locate a TPM. If that doesn't find anything, it will attempt to connect to a TPM on the standard TPM socket of 127.0.0.1:2321.

If set to FALSE, the program must be given direction about where to look for the TPM via command line arguments.

Command Line Argument	Description
`--win`	Connect to a TPM via the Windows API.
`--nix`	Connect to a TPM on Linux.
`--tcp --ip <ip:port>`	Connect to a TPM via socket.
`--tcp --ip <ip:port> --sim`	Connect to a TPM simulator via socket.

aca_address_port

Set to the address for the HIRS ACA. Use the format <scheme>://<host>:<port>

efi_prefix

If artifacts are expected to be retrievable from the device's EFI System Partition (ESP), set this to the path where the ESP is mounted on the system. Artifacts will be captured according to the file structure defined in the TCG FIM or TCG RIM specifications. They will be uploaded along with other evidence used during provisioning, and they don't have to be separately uploaded via the portal.

Recommended setting on Windows

The ESP can be mounted to any open drive path.

1. **`mountvol P: /S`**
2. Set the **`"efi_prefix": "P:"`**

Recommended setting on Linux

The ESP is loaded to a directory by the Linux kernel by default.

Set the **"efi_prefix": "/boot/efi"**

event_log_file

The TCG Event Log must be retrieved when performing RIM validation.

Windows

By default on Windows, the HIRS .NET Provisioner will ask the Windows API to provide the latest TCG Event Log. Alternatively, you may want to set this key in the appsettings file to a specific path.

Linux

On Linux, if this key is not set in the appsettings file, the HIRS .NET Provisioner will attempt to retrieve the TCG Event Log from the standard location it is made available from the Linux kernel.

hardware_manifest_collectors

This can be set to a comma separated list of plugins that implement paccor's IHardwareManifest interface. These plugins will collect hardware information according to different Component Class Registries. Plugins must be installed in a plugins folder under the installation directory. The current default plugin is called paccor_scripts. If this setting is not changed in appsettings.json, hardware information will be collected using paccor's shell scripts on Windows or Linux.

paccor_output_file

Alternatively, hardware information can be read from a JSON file. If this option is set in the appsettings file, the HIRS .NET Provisioner will attempt to read it as a file. The contents will be sent to the ACA. The JSON format is described in paccor's readme file. Example format can be reviewed after running paccor's allcomponents script.

Linux platform descriptor files

The following settings will change where the HIRS .NET Provisioner looks for certain system descriptor files made available by the Linux kernel. These could be useful if the file locations change over time, or if different values are desired for provisioning. Within the ACA these values are used in database lookups to match records.

linux_bios_vendor_file

The location of the system bios_vendor file.

linux_bios_version_file

The location of the system bios_version file.

linux_bios_date_file

The location of the system bios_date file.

linux_sys_vendor_file

The location of the system sys_vendor file.

linux_product_name_file

The location of the system product_name file.

linux_product_version_file

The location of the system product_version file.

linux_product_serial_file

The location of the system product_serial file.

Logging

The HIRS .NET Provisioner uses Serilog for logging. A standard configuration is defined in the appsettings.json file. In this standard configuration, the log file will appear in the directory that the HIRS .NET Provisioner is executed from. The log file contains detailed information about the HIRS provisioning process.

Additional configuration options can be found in Serilog documentation. See https://github.com/serilog/serilog-settings-configuration.

Releases

Distribution packages for the HIRS .NET Provisioner are available for download from the Releases page of our repository on GitHub. https://github.com/nsacyber/hirs/releases. If a different format or CPU target is desired, those options can be configured as part of the build process.

Windows MSI

The HIRS .NET Provisioner is available for download as an .msi file for 64-bit Windows.

Linux RPM or DEB

The HIRS .NET Provisioner is available for download as a .deb or .rpm file for different distributions of 64-bit Linux.

Build from Source

The HIRS .NET Provisioner requires the .NET SDK version 6 or later. Please look for instructions from Microsoft for installing the .NET SDK on your system. You may also look at our GitHub Actions workflows for hints on setting up your build environment.

Most of the build commands work the same way whether they are run from Windows or Linux. Location of where these commands are run matters. The directory the command should be run from is included in each section below.

The repository must be checked out with submodules enabled for successful building of the HIRS .NET Provisioner.

Checkout from GitHub

```
git clone https://github.com/nsacyber/HIRS.git
```

Restore

This command will pull in all dependencies required for building or testing the source code.

```
cd <HIRS repository>/HIRS_Provisioner.NET/
```

```
dotnet restore
```

Build and run Unit Tests

```
dotnet test
```

Create installation package(s)

Depending on your choice below, you might need to install a packaging library. This step is not needed to build an MSI installer.

```
dotnet tool install --global dotnet-deb

dotnet tool install --global dotnet-rpm

dotnet tool install --global dotnet-zip

cd <HIRS repository>/HIRS_Provisioner.NET/hirs
```

MSI

```
dotnet msbuild hirs.csproj /t:Msi /P:TargetFramework=net6.0
/p:RuntimeIdentifier=win-x64 /p:Configuration=Release
```

RPM

```
dotnet rpm -r linux-x64 -c Release
```

DEB

```
dotnet deb -r linux-x64 -c Release
```

ZIP (Pre-built binary for Windows)

```
dotnet zip -r win-x64 -c Release
```

ZIP (Pre-built binary for Linux)

```
dotnet zip -r linux-x64 -c Release
```

Troubleshooting

Nothing prints to the terminal when run

Check that the JSON is valid in the appsettings.json file. Search “JSON validator” on a preferred internet search engine, then copy and paste the JSON into the validator. This will help find any structural errors in the settings file.