binarly-io/efiXplorer

IDA plugin for UEFI firmware analysis and reverse engineering automation

ref: ef4d5e17560ae39417b5e2ce8b9a1a775d710d56

License: GNU General Public License v3.0

Stars: 888

Forks: 105

Top Contributors

yeggor (497)

matrosov (36)

assafcarlsbad (6)

pagabuc (6)

TakahiroHaruyama (5)

isciurus (5)

p41l (2)

RolfRolles (2)

cc-crack (2)

river-li (2)

xorpse (2)

Cr4sh (1)

naconaco (1)

NikolajSchlej (1)

Chapter 0.0.0

root

README.md

```
[![License: GPL v3](https://img.shields.io/badge/License-GPL%20v3-blue.svg)]
(http://www.gnu.org/licenses/gpl-3.0)
[![efiXplorer CI](https://github.com/binarly-io/efiXplorer/actions/workflows/ci-
build.yml/badge.svg)](https://github.com/binarly-io/efiXplorer/actions)

<p align="center">
  <img alt="efiXlorer Logo" src="pics/efiXplorer_logo.png" width="45%">
</p>

**efiXplorer** - IDA plugin for UEFI firmware analysis and reverse
engineering automation

**Supported versions of Hex-Rays products:** everytime we focus on last versions of IDA
and Decompiler because we try to use most recent features from new SDK releases. That
means we tested only on recent versions of Hex-Rays products and do not guarantee
stable work on previous generations.

**Why not IDApython:** all code developed in C++ because it's a more stable and
performant way to support a complex plugin and get full power of most recent
SDK's features.

**Supported Platforms:** Windows, Linux and OSX.

## [efiXplorer core features](https://github.com/binarly-
io/efiXplorer/wiki/efiXplorer-features)

## [efiXloader description](https://github.com/binarly-io/efiXplorer/wiki/efiXloader)

## [Build instructions and Installation](https://github.com/binarly-
io/efiXplorer/wiki/Build-instruction-and-installation)

## Publications

- [efiXplorer: Hunting for UEFI Firmware Vulnerabilities at Scale with Automated Static
Analysis](https://i.blackhat.com/eu-20/Wednesday/eu-20-Labunets-efiXplorer-Hunting-For-
UEFI-Firmware-Vulnerabilities-At-Scale-With-Automated-Static-Analysis.pdf)
- [Static analysis-based recovery of service function calls in UEFI firmware]
(https://github.com/binarly-
io/Research_Publications/blob/main/EKO_2020/EKO_2020_efiXplorer.pdf)
- [How efiXplorer helping to solve challenges in reverse engineering of UEFI firmware]
(https://www.youtube.com/watch?v=FFGQJBmRkLw)

## References

- https://github.com/LongSoft/UEFITool
- https://github.com/yeggor/uefi_retool
- https://github.com/gdbinit/EFISwissKnife
- https://github.com/snare/ida-efiutils
- https://github.com/al3xtjames/ghidra-firmware-utils
- https://github.com/DSecurity/efiSeek
- https://github.com/p-state/ida-efitools2
- https://github.com/zznop/bn-uefi-helper
}
```

CMakeLists.txt

```
CMAKE_MINIMUM_REQUIRED(VERSION 3.7)

PROJECT(efiXplorer_everything)

ADD_SUBDIRECTORY(efiXplorer)
ADD_SUBDIRECTORY(efiXloader)
}
```

LICENSE

authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under

the conditions stated below. Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7. This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy. This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged. This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not

used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status

of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently

reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a

party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License,

section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

```
  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.
}
```

build.py

```
#!/usr/bin/env python3

import os
import subprocess
```

```
import click


@click.group()
def cli():
    pass


@click.command()
@click.option(
    "--hexrays_sdk",
    "hexrays_sdk",
    type=str,
    default=str(),
    help="path to hexrays_sdk directory",
)
@click.argument("idasdk")
def build_plugin(idasdk: str, hexrays_sdk: str):
    """Build efiXplorer plugin"""

    os.chdir("efiXplorer")

    if not os.path.isdir("build"):
        os.mkdir("build")

    os.chdir("build")

    command = ["cmake", "..", f"-DIdaSdk_ROOT_DIR={idasdk}"]
    if hexrays_sdk:
        print("[INFO] HexRays analysis will be enabled")
        command.append(f"-DHexRaysSdk_ROOT_DIR={hexrays_sdk}")
    subprocess.call(command)
    subprocess.call(["cmake", "--build", ".", "--config", "Release", "--parallel"])


@click.command()
@click.argument("idasdk")
def build_loader(idasdk: str):
    """Build efiXloader"""

    os.chdir("efiXloader")

    if not os.path.isdir("build"):
        os.mkdir("build")

    os.chdir("build")

    command = ["cmake", "..", f"-DIdaSdk_ROOT_DIR={idasdk}"]
    subprocess.call(command)
    subprocess.call(["cmake", "--build", ".", "--config", "Release", "--parallel"])


@click.command()
@click.option(
    "--hexrays_sdk",
    "hexrays_sdk",
    type=str,
    default=str(),
    help="path to hexrays_sdk directory",
)
@click.argument("idasdk")
def build_everything(idasdk: str, hexrays_sdk: str):
    """Build plugin (efiXplorer) and loader (efiXloader)"""
```

```
    if not os.path.isdir("build"):
        os.mkdir("build")

    os.chdir("build")

    command = ["cmake", "..", f"-DIdaSdk_ROOT_DIR={idasdk}"]
    if hexrays_sdk:
        print("[INFO] HexRays analysis will be enabled")
        command.append(f"-DHexRaysSdk_ROOT_DIR={hexrays_sdk}")
    subprocess.call(command)
    subprocess.call(["cmake", "--build", ".", "--config", "Release", "--parallel"])


cli.add_command(build_plugin)
cli.add_command(build_loader)
cli.add_command(build_everything)


if __name__ == "__main__":
    cli()
}
```

Chapter 1.0.0

efiXloader

efiXloader/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.7)

project(efiXloader)

set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_EXPORT_COMPILE_COMMANDS ON)

if(APPLE)
  # to build Mach-O universal binaries with 2 architectures
  set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -fPIC -arch x86_64 -arch arm64")
  set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -fPIC -arch x86_64 -arch arm64")
else()
  set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -fPIC")
endif()

if(CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  set(CMAKE_CXX_FLAGS
      "${CMAKE_CXX_FLAGS} -Wno-nullability-completeness -Wno-varargs")
endif()

list(APPEND CMAKE_MODULE_PATH ${PROJECT_SOURCE_DIR}/../cmake)

find_package(IdaSdk REQUIRED)

include_directories(${PROJECT_SOURCE_DIR}/../efiXplorer/3rd/nlohmann_json)
```

```
file(
  GLOB
  uefitool_src
  "3rd/uefitool/common/*.h"
  "3rd/uefitool/common/*.c"
  "3rd/uefitool/common/*.cpp"
  "3rd/uefitool/common/LZMA/*.cpp"
  "3rd/uefitool/common/LZMA/*.c"
  "3rd/uefitool/common/LZMA/*.h"
  "3rd/uefitool/common/LZMA/SDK/C/*.c"
  "3rd/uefitool/common/LZMA/SDK/C/*.cpp"
  "3rd/uefitool/common/LZMA/SDK/C/*.h"
  "3rd/uefitool/common/Tiano/*.c"
  "3rd/uefitool/common/Tiano/*.cpp"
  "3rd/uefitool/common/Tiano/*.h"
  "3rd/uefitool/common/bstrlib/*.c"
  "3rd/uefitool/common/bstrlib/*.cpp"
  "3rd/uefitool/common/bstrlib/*.h"
  "3rd/uefitool/common/digest/sha1.c"
  "3rd/uefitool/common/digest/sha256.c"
  "3rd/uefitool/common/digest/sha512.c"
  "3rd/uefitool/common/digest/sm3.c"
  "3rd/uefitool/common/zlib/*.c"
  "3rd/uefitool/common/zlib/*.cpp"
  "3rd/uefitool/common/zlib/*.h"
  "3rd/uefitool/version.h"
  "3rd/uefitool/ffsdumper.cpp"
  "3rd/uefitool/ffsdumper.h"
  "3rd/uefitool/uefidump.cpp"
  "3rd/uefitool/uefidump.h")

# efiLoader sources
file(GLOB efiloader_src "*.cc" "*.h")

add_ida_loader(efiXloader NOEA32 ${PROJECT_SOURCE_DIR}/efi_loader.cc)

set_ida_target_properties(efiXloader PROPERTIES CXX_STANDARD 17)
ida_target_include_directories(efiXloader PRIVATE ${IdaSdk_INCLUDE_DIRS})

add_ida_library(efiXloader_lib ${efiloader_src} ${uefitool_src} uefitool.cc
                uefitool.h)
ida_target_link_libraries(efiXloader efiXloader_lib)
}
```

efiXloader/efi_loader.h

```
/*
 * efiXloader
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
```

```
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include <vector>

#include "ida_core.h"
#include "pe.h"
#include "pe_manager.h"
#include "uefitool.h"

extern loader_t LDSC;

//-----------------------
// definitions
//-----------------------

void idaapi load_binary(const char *fname);
void idaapi close_and_save_db(const char *fname);
void idaapi reanalyse_all(void);
void idaapi wait(void);
void idaapi idb_to_asm(const char *fname);
void idaapi clean_db(void);

void idaapi efi_til_init();

class Ui {
public:
  Ui() { ; }
  int ask_for_single_image();
};

class driver_chooser_t : public chooser_t {
protected:
  static const int widths_drivers[];
  static const char *const drivers_headers[];

public:
  /* remember the addresses in this qvector */
  qvector<qstring> drivers_names;

  /* this object must be allocated using `new` */
  driver_chooser_t(const char *title, bool ok,
                   std::vector<efiloader::File *> drivers);

  /* function that is used to decide whether a new chooser should be opened or
   * we can use the existing one. The contents of the window are completely
   * determined by its title */
  virtual const void *get_obj_id(size_t *len) const {
    *len = strlen(title);
    return title;
  }

  /* function that returns number of lines in the list */
  virtual size_t idaapi get_count() const { return drivers_names.size(); }

  /* function that generates the list line */
  virtual void idaapi get_row(qstrvec_t *cols, int *icon_,
                              chooser_item_attrs_t *attrs, size_t n) const;
```

```
  /* function that is called when the user hits Enter */
  virtual cbret_t idaapi enter(size_t n) {
    if (n < drivers_names.size()) {
      // jumpto(list[n]);
    }
    return cbret_t();
  }

protected:
  void build_list(bool ok, std::vector<efiloader::File *> files) {
    size_t n = 0;
    for (auto file : files) {
      drivers_names.push_back(qstring(file->qname));
      n++;
    }
    ok = true;
  }
};
}
```

efiXloader/ida_core.h

```
/*
 * efiXloader
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#define USE_STANDARD_FILE_FUNCTIONS 1

#include <fpro.h>
#include <pro.h>

#include <auto.hpp>
#include <bytes.hpp>
#include <diskio.hpp>
#include <entry.hpp>
#include <fixup.hpp>
#include <ida.hpp>
#include <idp.hpp>
#include <kernwin.hpp>
#include <loader.hpp>
#include <name.hpp>
#include <offset.hpp>
#include <segment.hpp>
```

```
#include <segregs.hpp>

//-----------------------------------

#define CLASS_CODE "CODE"
#define NAME_CODE ".text"
#define CLASS_DATA "DATA"
#define CLASS_CONST "CONST"
#define NAME_DATA ".data"
#define CLASS_BSS "BSS"
#define NAME_BSS ".bss"
#define NAME_EXTERN "extern"
#define NAME_COMMON "common"
#define NAME_ABS "abs"
#define NAME_UNDEF "UNDEF"
#define CLASS_STACK "STACK"
#define CLASS_RES16 "RESOURCE"
#define LDR_NODE "$ IDALDR node for ids loading $"
#define LDR_INFO_NODE "$ IDALDR node for unload $"

//-------------------------------------------------------------------------
template <class T>
bool _validate_array_count(linput_t *li, T *p_cnt, size_t elsize,
                           int64 current_offset = -1, int64 max_offset = -1) {
  if (current_offset == -1)
    current_offset = qltell(li);
  if (max_offset == -1)
    max_offset = qlsize(li);
  int64 rest = max_offset - current_offset;
  T cnt = *p_cnt;
  if (current_offset >= 0 && rest >= 0) {
#ifndef __X86__
    typedef size_t biggest_t;
#else
    typedef ea_t biggest_t;
#endif
    if (is_mul_ok<biggest_t>(elsize, cnt)) {
      biggest_t needed = elsize * cnt;
#ifdef __X86__
      if (needed == size_t(needed))
#endif
        if (rest >= needed)
          return true; // all ok
    }
    cnt = rest / elsize;
  } else {
    cnt = 0;
  }
  *p_cnt = cnt;
  return false;
}

//-------------------------------------------------------------------------
// Validate a counter taken from the input file. If there are not enough bytes
// in the input file, ask the user if we may continue and fix the counter.
template <class T>
void validate_array_count(linput_t *li, T *p_cnt, size_t elsize,
                          const char *counter_name, int64 curoff = -1,
                          int64 maxoff = -1) {
  T old = *p_cnt;
  if (!_validate_array_count(li, p_cnt, elsize, curoff, maxoff)) {
    static const char *const format =
        "AUTOHIDE SESSION\n"
        "HIDECANCEL\n"
```

```
        "%s %" FMT_64 "u is incorrect, maximum possible value is %" FMT_64
        "u%s";
#ifndef __KERNEL__
    if (ask_yn(ASKBTN_YES, format, counter_name, static_cast<uint64>(old),
               static_cast<uint64>(*p_cnt),
               ". Do you want to continue with the new value?") != ASKBTN_YES) {
      loader_failure(NULL);
    }
#else
    warning(format, counter_name, static_cast<uint64>(old),
            static_cast<uint64>(*p_cnt), "");
#endif
  }
}

//--------------------------------------------------------------------------
// Validate a counter taken from the input file. If there are not enough bytes
// in the input file, die.
template <class T>
void validate_array_count_or_die(linput_t *li, T cnt, size_t elsize,
                                 const char *counter_name, int64 curoff = -1,
                                 int64 maxoff = -1) {
  if (!_validate_array_count(li, &cnt, elsize, curoff, maxoff)) {
    static const char *const format =
        "%s is incorrect, maximum possible value is %u%s";
#ifndef __KERNEL__
    loader_failure(format, counter_name, uint(cnt), "");
#else
    error(format, counter_name, uint(cnt), "");
#endif
  }
}

//----------------------------------
inline uchar readchar(linput_t *li) {
  uchar x;
  lread(li, &x, sizeof(x));
  return x;
}

//----------------------------------
inline uint16 readshort(linput_t *li) {
  uint16 x;
  lread(li, &x, sizeof(x));
  return x;
}

//----------------------------------
inline uint32 readlong(linput_t *li) {
  uint32 x;
  lread(li, &x, sizeof(x));
  return x;
}

inline uint32 mf_readlong(linput_t *li) { return swap32(readlong(li)); }
inline uint16 mf_readshort(linput_t *li) { return swap16(readshort(li)); }
}
```

efiXloader/pe.h

```
/*
 * efiXloader
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include <string>

#include "ida_core.h"
#include "pe_ida.h"
#include "utils.h"

#include <typeinf.hpp>

#define PAGE_SIZE 0x1000

#define MZ_SIGN 0x5A4D       // MZ header
#define MAGIC_P32 0x10B      // Normal PE file
#define MAGIC_P32_PLUS 0x20B // 64-bit image
#define PE_SIGN 0x4550       // PE signature

namespace efiloader {

class PE {
public:
  PE(linput_t *i_li, std::basic_string<char> fname, ea_t *base,
     ushort *sel_base, int ord, uint16_t mt) {
    _image_name = fname.substr(fname.find_last_of("/\\") + 1);
    msg("[efiXloader] image name is %s\n", _image_name.c_str());
    pe_base = base;
    pe_sel_base = sel_base;
    li = i_li;
    utils = new Utils;
    _sec_off = 0;
    _sec_ea = 0;
    _sel = 0;
    _ord = ord;
    inf_set_64bit();
    if (mt == PECPU_ARM64) {
      set_processor_type("arm", SETPROC_LOADER);
    } else {
      set_processor_type("metapc", SETPROC_LOADER);
    }
    cm_t cm = inf_get_cc_cm() & ~CM_MASK;
    inf_set_cc_cm(cm | CM_N64);
    if (default_compiler() == COMP_UNK) {
      set_compiler_id(COMP_MS);
    }
    reset();
```

```
  };
  ~PE() {
    close_linput(li);
    delete utils;
  }
  uint32_t number_of_sections;
  uint32_t number_of_dirs;
  char *name;
  bool is_reloc_dir(uint32_t i) { return i == 5; }
  bool is_debug_dir(uint32_t i) { return i == 6; }
  void set_64_bit_segm_and_rabase(ea_t ea) {
    segment_t *tmp_seg = getseg(ea);
    set_segm_addressing(tmp_seg, 2);
    set_segm_base(tmp_seg, *pe_base);
  }
  void set_64_bit(ea_t ea) {
    segment_t *tmp_seg = getseg(ea);
    set_segm_addressing(tmp_seg, 2);
  }
  bool is_p32();
  bool is_p32_plus();
  bool is_pe();
  bool good();
  bool process();
  uint16_t arch();
  // data processing
  inline size_t make_named_byte(ea_t ea, const char *name,
                                const char *extra = NULL, size_t count = 1);
  inline size_t make_named_word(ea_t ea, const char *name,
                                const char *extra = NULL, size_t count = 1);
  inline size_t make_named_dword(ea_t ea, const char *name,
                                 const char *extra = NULL, size_t count = 1);
  inline size_t make_named_qword(ea_t ea, const char *name,
                                 const char *extra = NULL, size_t count = 1);
  inline ea_t skip(ea_t ea, qoff64_t off) { return ea + off; }
  // ida db processing
  void push_to_idb(ea_t start, ea_t end) {
    // Map header
    file2base(li, 0x0, start, start + headers_size, FILEREG_PATCHABLE);
    // Map sections
    for (int i = 0; i < number_of_sections; i++) {
      file2base(li, _sec_headers[i].s_scnptr, start + _sec_headers[i].s_vaddr,
                start + _sec_headers[i].s_vaddr + _sec_headers[i].s_psize,
                FILEREG_PATCHABLE);
    }
  }

private:
  qvector<ea_t> segments_ea;
  std::basic_string<char> _full_path;
  std::basic_string<char> _image_name;
  efiloader::Utils *utils;
  linput_t *li;
  qoff64_t head_start();
  qoff64_t head_off;
  qoff64_t _pe_header_off;
  uint16_t headers_size;
  peheader_t pe;
  peheader64_t pe64;
  uint16_t _sec_num;
  uint16_t _bits;
  qvector<sel_t> selectors;
  qvector<sel_t> data_selectors;
  qvector<qstring> ds_seg_names;
```

```
  qvector<qstring> cs_seg_names;
  void reset() { qlseek(li, 0); }
  const char *_machine_name();
  //
  // PE image preprocessing
  //
  void preprocess();
  //
  // sections processing
  //
  qvector<pesection_t> _sec_headers;
  ea_t *pe_base;
  ushort *pe_sel_base;
  ushort _sel;
  ea_t _sec_off;
  ea_t _sec_ea;
  // pe ord
  uval_t _ord;
  ea_t image_base;
  uint32_t image_size;
  qvector<size_t> segm_sizes;
  qvector<size_t> segm_raw_sizes;
  qvector<ea_t> segm_entries;

  int preprocess_sections();
  //
  // functions processing
  //
  void make_entry(ea_t ea);
  void make_code(ea_t ea);
  //
  // segments processing
  //
  qstring code_segm_name;
  qstring data_segm_name;
  sel_t data_segment_sel;
  qvector<segment_t *> segments;
  qvector<qstring> segm_names;
  qvector<qstring> secs_names;
  ea_t process_section_entry(ea_t ea);
  segment_t *make_generic_segment(ea_t seg_ea, ea_t seg_ea_end,
                                  const char *section_name, uint32_t flags);
  segment_t *make_head_segment(ea_t start, ea_t end, const char *name);
  void setup_ds_selector();
};
} // namespace efiloader

enum MachineType { AMD64 = 0x8664, I386 = 0x014C, AARCH64 = 0xaa64 };
}
```

efiXloader/pe_ida.h

```
/*
 * efiXloader
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
```

```
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include <stddef.h>
#include <time.h>

#include "ida_core.h"

#pragma pack(push, 1)
//--------------------------------------------------------------------------
//
// 32-bit Portable EXE Header
//
//--------------------------------------------------------------------------
struct petab_t {
  uint32 rva;  // relative virtual address
  uint32 size; // size
}; // PE va/size array element

template <class pointer_t> struct peheader_tpl {
  int32 signature; // 00 Current value is "PE/0/0".

#define PEEXE_ID 0x4550    // 'PE' followed by two zeroes
#define BPEEXE_ID 0x455042 // Borland's extenson for DPMI'host
#define PLEXE_ID                                                               \
  0x4C50 // 'PL', PharLap TNT DOS-Extender Lite file that uses real mode APIs
#define TEEXE_ID 0x5A56      // 'VZ', EFI Terse Executable
  uint16 machine;            // 04 This field specifies the type of CPU
                             // compatibility required by this image to run.
                             // The values are:
#define PECPU_UNKNOWN 0x0000 // unknown

#define PECPU_80386 0x014C // 80386
#define PECPU_80486 0x014D // 80486
#define PECPU_80586 0x014E // 80586

#define PECPU_R3000 0x0162     // MIPS Mark I (R2000, R3000)
#define PECPU_R6000 0x0163     // MIPS Mark II (R6000)
#define PECPU_R4000 0x0166     // MIPS Mark III (R4000)
#define PECPU_R10000 0x0168    // MIPS Mark IV (R10000)
#define PECPU_WCEMIPSV2 0x0169 // MIPS little-endian WCE v2
#define PECPU_MIPS16 0x0266    // MIPS16
#define PECPU_MIPSFPU 0x0366   // MIPS with FPU
#define PECPU_MIPSFPU16 0x0466 // MIPS16 with FPU

#define PECPU_ALPHA 0x0184   // DEC Alpha
#define PECPU_ALPHA64 0x0284 // Dec Alpha 64-bit

#define PECPU_SH3 0x01A2    // SH3
#define PECPU_SH3DSP 0x01A3 // SH3DSP
#define PECPU_SH3E 0x01A4   // SH3E
#define PECPU_SH4 0x01A6    // SH4
#define PECPU_SH5 0x01A8    // SH5
```

```
#define PECPU_ARM 0x01C0   // ARM
#define PECPU_ARMI 0x01C2  // ARM with Thumb
#define PECPU_ARMV7 0x01C4 // ARMv7 (or higher) Thumb mode only

#define PECPU_AM33 0x01D3 // Matsushita (Panasonic) AM33/MN10300

#define PECPU_PPC 0x01F0   // PowerPC
#define PECPU_PPCFP 0x01F1 // PowerPC with floating-point
#define PECPU_PPCBE 0x01F2 // PowerPC Big-Endian

#define PECPU_M32R 0x9041 // M32R little-endian

#define PECPU_IA64 0x0200    // Intel Itanium IA64
#define PECPU_EPOC 0x0A00    // ARM EPOC
#define PECPU_AMD64 0x8664   // AMD64 (x64)
#define PECPU_ARM64 0xaa64   // ARMv8 in 64-bit mode
#define PECPU_M68K 0x0268    // Motorola 68000 series
#define PECPU_EBC 0x0EBC     // EFI Bytecode
#define PECPU_CEF 0x0CEF     // ?
#define PECPU_CEE 0xC0EE     // ?
#define PECPU_TRICORE 0x0520 // TRICORE (Infineon)

  bool is_64bit_cpu(void) const {
    return machine == PECPU_AMD64 || machine == PECPU_IA64 ||
           machine == PECPU_ARM64;
  }
  bool is_pc(void) const {
    return machine == PECPU_80386 || machine == PECPU_80486 ||
           machine == PECPU_80586 || machine == PECPU_AMD64;
  }
  bool is_mips(void) const {
    return machine == PECPU_R3000 || machine == PECPU_R6000 ||
           machine == PECPU_R4000 || machine == PECPU_R10000 ||
           machine == PECPU_WCEMIPSV2 || machine == PECPU_MIPS16 ||
           machine == PECPU_MIPSFPU || machine == PECPU_MIPSFPU16;
  }

  bool is_arm(void) const {
    return machine == PECPU_ARM || machine == PECPU_ARMI ||
           machine == PECPU_ARMV7;
  }

  bool has_code16_bit(void) const { return is_arm() || is_mips(); }

  uint16 nobjs; // 06 This field specifies the number of entries
  // in the Object Table.
  qtime32_t datetime; // 08 Used to store the time and date the file was
  // created or modified by the linker.
  uint32 symtof;  // 0C Symbol Table Offset
  uint32 nsyms;   // 10 Number of Symbols in Symbol Table
  uint16 hdrsize; // 14 This is the number of remaining bytes in the NT
  // header that follow the FLAGS field.
  uint16 flags;          // 16 Flag bits for the image.  0000h Program image.
#define PEF_BRVHI 0x8000 // Big endian: MSB precedes LSB in memory
#define PEF_UP 0x4000    // File should be run only on a UP machine
#define PEF_DLL 0x2000   // Dynamic Link Library (DLL)
#define PEF_SYS 0x1000   // System file
#define PEF_NSWAP                                                          \
  0x0800 // If the image is on network media, fully load it and copy it to the
         // swap file.
#define PEF_SWAP                                                           \
  0x0400                 // If image is on removable media,
                         // copy and run from swap file
#define PEF_NODEB 0x0200 // Debugging info stripped
```

```
#define PEF_32BIT 0x0100 // 32-bit word machine
#define PEF_BRVLO 0x0080 // Little endian: LSB precedes MSB in memory
#define PEF_16BIT 0x0040 // 16-bit word machine
#define PEF_2GB 0x0020   // App can handle > 2gb addresses
#define PEF_TMWKS 0x0010 // Aggressively trim working set
#define PEF_NOSYM 0x0008 // Local symbols stripped
#define PEF_NOLIN 0x0004 // Line numbers stripped
#define PEF_EXEC 0x0002  // Image is executable
#define PEF_NOFIX 0x0001 // Relocation info stripped

  int32 first_section_pos(int32 peoff) const {
    return peoff + offsetof(peheader_tpl, magic) + hdrsize;
  }

  // COFF fields:

  uint16 magic;              // 18 Magic
#define MAGIC_ROM 0x107      // ROM image
#define MAGIC_P32 0x10B      // Normal PE file
#define MAGIC_P32_PLUS 0x20B // 64-bit image
  bool is_pe_plus(void) const { return magic == MAGIC_P32_PLUS; }
  uchar vstamp_major; // 1A Major Linker Version
  uchar vstamp_minor; // 1B Minor Linker Version
  uint32 tsize;       // 1C TEXT size (padded)
  uint32 dsize;       // 20 DATA size (padded)
  uint32 bsize;       // 24 BSS  size (padded)
  uint32 entry;       // 28 Entry point
  uint32 text_start;  // 2C Base of text
  union {
    struct {
      uint32 data_start; // 30 Base of data

      // Win32/NT extensions:

      uint32 imagebase32; // 34 Virtual base of the image.
    };
    uint64 imagebase64;
  };
  uint64 imagebase() const {
    if (is_pe_plus())
      return imagebase64;
    else
      return imagebase32;
  }
  // This will be the virtual address of the first
  // byte of the file (Dos Header).  This must be
  // a multiple of 64K.
  uint32 objalign; // 38 The alignment of the objects. This must be a power
  // of 2 between 512 and 256M inclusive. The default
  // is 64K.
  uint32 filealign; // 3C Alignment factor used to align image pages.
                    // The alignment factor (in bytes) used to align the
                    // base of the image pages and to determine the
                    // granularity of per-object trailing zero pad.
                    // Larger alignment factors will cost more file space;
                    // smaller alignment factors will impact demand load
                    // performance, perhaps significantly. Of the two,
                    // wasting file space is preferable.  This value
                    // should be a power of 2 between 512 and 64K inclusive.
                    // Get the file position aligned:
#define FILEALIGN                                                       \
  512 // IDA5.1: it seems that for standard object alignment (if 4096)
  // the Windows kernel does not use filealign
  // (just checks that it is in the valid range) but uses 512
```

```
  uint32 get_align_mask(void) const {
    return ((objalign == 4096 || filealign == 0) ? FILEALIGN : filealign) - 1;
  }
  uint32 align_up_in_file(uint32 pos) const {
    if (is_efi()) // apparently EFI images are not aligned
      return pos;
    uint32 mask = get_align_mask();
    return (pos + mask) & ~mask;
  }
  uint32 align_down_in_file(uint32 pos) const {
    return is_efi() ? pos : (pos & ~get_align_mask());
  }
  uint16 osmajor;     // 40 OS version number required to run this image.
  uint16 osminor;     // 42 OS version number required to run this image.
  uint16 imagemajor;  // 44 User major version number.
  uint16 imageminor;  // 46 User minor version number.
  uint16 subsysmajor; // 48 Subsystem major version number.
  uint16 subsysminor; // 4A Subsystem minor version number.

  uint32 subsystem_version(void) const {
    return (subsysmajor << 16) | subsysminor;
  }

  uint32 reserved;  // 4C
  uint32 imagesize; // 50 The virtual size (in bytes) of the image.
  // This includes all headers.  The total image size
  // must be a multiple of Object Align.
  uint32 allhdrsize; // 54 Total header size. The combined size of the Dos
  // Header, PE Header and Object Table.
  uint32 checksum; // 58 Checksum for entire file.  Set to 0 by the linker.
  uint16 subsys;   // 5C NT Subsystem required to run this image.
#define PES_UNKNOWN 0x0000 // Unknown
#define PES_NATIVE 0x0001  // Native
#define PES_WINGUI 0x0002  // Windows GUI
#define PES_WINCHAR 0x0003 // Windows Character
#define PES_OS2CHAR 0x0005 // OS/2 Character
#define PES_POSIX 0x0007   // Posix Character
#define PES_NAT9x 0x0008   // image is a native Win9x driver
#define PES_WINCE 0x0009   // Runs on Windows CE.
#define PES_EFI_APP 0x000A // EFI application.
#define PES_EFI_BDV 0x000B // EFI driver that provides boot services.
#define PES_EFI_RDV 0x000C // EFI driver that provides runtime services.
#define PES_EFI_ROM 0x000D // EFI ROM image
#define PES_XBOX 0x000E    // Xbox system
#define PES_BOOTAPP 0x0010 // Windows Boot Application

  bool is_efi(void) const {
    return subsys == PES_EFI_APP || subsys == PES_EFI_BDV ||
           subsys == PES_EFI_RDV || subsys == PES_EFI_ROM;
  }
  bool is_console_app(void) const {
    return subsys == PES_WINCHAR || subsys == PES_OS2CHAR ||
           subsys == PES_POSIX;
  }
  bool is_userland(void) const {
    return subsys == PES_WINGUI || subsys == PES_WINCHAR ||
           subsys == PES_OS2CHAR || subsys == PES_POSIX || subsys == PES_WINCE;
  }
  uint16 dllflags;       // 5E Indicates special loader requirements.
#define PEL_PINIT 0x0001 // Per-Process Library Initialization.
#define PEL_PTERM 0x0002 // Per-Process Library Termination.
#define PEL_TINIT 0x0004 // Per-Thread Library Initialization.
#define PEL_TTERM 0x0008 // Per-Thread Library Termination.
#define PEL_HIGH_ENT                                                    \
```

```
  0x0020 // Image can handle a high entropy 64-bit virtual address space.
#define PEL_DYNAMIC_BASE 0x0040    // The DLL can be relocated at load time.
#define PEL_FORCE_INTEGRITY 0x0080 // Code integrity checks are forced.
#define PEF_NX                                                              \
  0x0100 // The image is compatible with data execution prevention (DEP).
#define PEF_NO_ISOLATION                                                    \
  0x0200 // The image is isolation aware, but should not be isolated.
#define PEF_NO_SEH                                                          \
  0x0400 // The image does not use structured exception handling (SEH). No
         // handlers can be called in this image.
#define PEL_NO_BIND 0x0800      // Do not bind image
#define PEL_APPCONTAINER 0x1000 // Image should execute in an AppContainer
#define PEL_WDM_DRV 0x2000      // Driver is a WDM Driver
#define PEL_GUARDCF 0x4000      // Image supports Control Flow Guard checking
#define PEL_TSRVAWA 0x8000      // Image is Terminal Server aware

  pointer_t stackres; // 60 Stack size needed for image. The memory is
  // reserved, but only the STACK COMMIT SIZE is
  // committed. The next page of the stack is a
  // 'guarded page'. When the application hits the
  // guarded page, the guarded page becomes valid,
  // and the next page becomes the guarded page.
  // This continues until the RESERVE SIZE is reached.
  pointer_t stackcom; // 64 Stack commit size.
  pointer_t heapres;  // 68 Size of local heap to reserve.
  pointer_t heapcom;  // 6C Amount to commit in local heap.
  uint32 loaderflags; // 70 ?
  uint32 nrvas;       // 74 Indicates the size of the VA/SIZE array
  // that follows.
  petab_t expdir; // 0 78 Export Directory
  petab_t impdir; // 1 80 Import Directory
  petab_t resdir; // 2 88 Resource Directory
  petab_t excdir; // 3 90 Exception Directory
  petab_t secdir; // 4 98 Security Directory
  //      The Certificate Table entry points to a table of
  //      attribute certificates. These certificates are not
  //      loaded into memory as part of the image. As such,
  //      the first field of this entry, which is normally
  //      an RVA, is a File Pointer instead
  petab_t reltab; // 5 A0 Relocation Table
  petab_t debdir; // 6 A8 Debug Directory
  petab_t desstr; // 7 B0 Description String
  petab_t cputab; // 8 B8 Machine Value
  petab_t tlsdir; // 9 C0 TLS Directory
  petab_t loddir; // 10 Load Configuration Directory
  petab_t bimtab; // 11 Bound Import Table address and size.
  petab_t iat;    // 12 Import Address Table address and size.
  petab_t didtab; // 13 Address and size of the Delay Import Descriptor.
  petab_t comhdr; // 14 COM+ Runtime Header address and size
  petab_t x00tab; // 15 Reserved

  bool is_te() const { return signature == TEEXE_ID; }
  inline bool has_debdir() const;
};

typedef peheader_tpl<uint32> peheader_t;
typedef peheader_tpl<uint64> peheader64_t;

const size_t total_rvatab_size =
    sizeof(peheader_t) - offsetof(peheader_t, expdir);
const size_t total_rvatab_count = total_rvatab_size / sizeof(petab_t);

//--------------------------------------------------------------------------
struct diheader_t {
```

```
  uint16 signature; // 00 Current value is "DI"
#define DBG_ID 0x4944
  uint16 flags2; // 02 ?? pedump mentions about this
  //    I've never seen something other than 0
  uint16 machine; // 04 This field specifies the type of CPU
  //    compatibility required by this image to run.
  uint16 flags;       // 06 Flag bits for the image.
  qtime32_t datetime; // 08 Used to store the time and date the file was
  //    created or modified by the linker.
  uint32 checksum;  // 12 Checksum
  uint32 imagebase; // 16 Virtual base of the image.
  //    This will be the virtual address of the first
  //    byte of the file (Dos Header).  This must be
  //    a multiple of 64K.
  uint32 imagesize; // 20 The virtual size (in bytes) of the image.
  //    This includes all headers.  The total image size
  //    must be a multiple of Object Align.
  uint32 n_secs;        // 24 Number of sections
  uint32 exp_name_size; // 28 Exported Names Size
  uint32 dbg_dir_size;  // 32 Debug Directory Size
  uint32 reserved[3];   // 36 Reserved fields
};

//------------------------------------------------------------------------
//
//      S E C T I O N S
//
struct pesection_t {
  char s_name[8];           /* section name */
  uint32 s_vsize;           /* virtual size */
  uint32 s_vaddr;           /* virtual address */
  uint32 s_psize;           /* physical size */
  int32 s_scnptr;           /* file ptr to raw data for section */
  int32 s_relptr;           /* file ptr to relocation */
  int32 s_lnnoptr;          /* file ptr to line numbers */
  uint16 s_nreloc;          /* number of relocation entries */
  uint16 s_nlnno;           /* number of line number entries */
  int32 s_flags;            /* flags */
#define PEST_REG 0x00000000 // obsolete: regular: allocated, relocated, loaded
#define PEST_DUMMY                                                      \
  0x00000001 // obsolete: dummy: not allocated, relocated, not loaded
#define PEST_NOLOAD                                                     \
  0x00000002 // obsolete: noload: allocated, relocated, not loaded
#define PEST_GROUP 0x00000004 // obsolete: grouped: formed of input sections
#define PEST_PAD                                                        \
  0x00000008 // obsolete: padding: not allocated, not relocated, loaded
#define PEST_COPY                                                       \
  0x00000010                    // obsolete: copy: for decision function used
                                //    by field update; not
                                //    allocated, not relocated,
                                //    loaded; reloc & lineno
                                //    entries processed normally */
#define PEST_TEXT 0x00000020L   // section contains text only
#define PEST_DATA 0x00000040L   // section contains data only
#define PEST_BSS 0x00000080L    // section contains bss only
#define PEST_EXCEPT 0x00000100L // obsolete: Exception section
#define PEST_INFO                                                       \
  0x00000200L // Comment: not allocated, not relocated, not loaded
#define PEST_OVER                                                       \
  0x00000400L // obsolete: Overlay: not allocated, relocated, not loaded
#define PEST_LIB 0x00000800L // ".lib" section: treated like PEST_INFO

#define PEST_LOADER 0x00001000L // Loader section: COMDAT
#define PEST_DEBUG 0x00002000L  // Debug section:
```

```
#define PEST_TYPCHK 0x00004000L // Type check section:
#define PEST_OVRFLO                                                              \
  0x00008000L                  // obsolete: RLD and line number overflow sec hdr
#define PEST_F0000 0x000F0000L // Unknown
#define PEST_ALIGN 0x00F00000L // Alignment 2^(x-1):
  uint32 get_sect_alignment(void) const {
    int align = ((s_flags >> 20) & 15);
    return align == 0 ? 0 : (1 << (align - 1));
  }

  asize_t get_vsize(const peheader_t &pe) const {
    return align_up(s_vsize ? s_vsize : s_psize, pe.objalign ? pe.objalign : 1);
  }

  asize_t get_psize(const peheader_t &pe) const {
    return qmin(s_psize, get_vsize(pe));
  }

#define PEST_1000000 0x01000000L // Unknown
#define PEST_DISCARD 0x02000000L // Discardable
#define PEST_NOCACHE 0x04000000L // Not cachable
#define PEST_NOPAGE 0x08000000L  // Not pageable
#define PEST_SHARED 0x10000000L  // Shareable
#define PEST_EXEC 0x20000000L    // Executable
#define PEST_READ 0x40000000L    // Readable
#define PEST_WRITE 0x80000000L   // Writable
};

//--------------------------------------------------------------------------
//
//      E X P O R T S
//
struct peexpdir_t {
  uint32 flags;       // Currently set to zero.
  qtime32_t datetime; // Time/Date the export data was created.
  uint16 major;       // A user settable major/minor version number.
  uint16 minor;
  uint32 dllname; // Relative Virtual Address of the Dll asciiz Name.
  // This is the address relative to the Image Base.
  uint32 ordbase; // First valid exported ordinal. This field specifies
  // the starting ordinal number for the export
  // address table for this image. Normally set to 1.
  uint32 naddrs; // Indicates number of entries in the Export Address
  // Table.
  uint32 nnames; // This indicates the number of entries in the Name
  // Ptr Table (and parallel Ordinal Table).
  uint32 adrtab; // Relative Virtual Address of the Export Address
  // Table. This address is relative to the Image Base.
  uint32 namtab; // Relative Virtual Address of the Export Name Table
  // Pointers. This address is relative to the
  // beginning of the Image Base. This table is an
  // array of RVA's with # NAMES entries.
  uint32 ordtab; // Relative Virtual Address of Export Ordinals
                 // Table Entry. This address is relative to the
                 // beginning of the Image Base.
};

//--------------------------------------------------------------------------
//
//      I M P O R T S
//
struct peimpdir_t {
  uint32 table1;      // aka OriginalFirstThunk
  qtime32_t datetime; // Time/Date the import data was pre-snapped or
```

```
  // zero if not pre-snapped.
  uint32 fchain;  // Forwarder chain
  uint32 dllname; // Relative Virtual Address of the Dll asciiz Name.
  // This is the address relative to the Image Base.
  uint32 looktab; // aka FirstThunk
                  // This field contains the address of the start of
                  // the import lookup table for this image. The address
                  // is relative to the beginning of the Image Base.
#define hibit(type) ((type(-1) >> 1) ^ type(-1))
#define IMP_BY_ORD32 hibit(uint32) // Import by ordinal, otherwise by name
#define IMP_BY_ORD64 hibit(uint64) // Import by ordinal, otherwise by name

  peimpdir_t(void) { memset(this, 0, sizeof(peimpdir_t)); }
};

// delayed load import table
struct dimpdir_t {
  uint32 attrs;            // Attributes.
#define DIMP_NOBASE 0x0001 // pe.imagebase was not added to addresses
  uint32 dllname;          // Relative virtual address of the name of the DLL
  // to be loaded. The name resides in the read-only
  // data section of the image.
  uint32 handle; // Relative virtual address of the module handle
  // (in the data section of the image) of the DLL to
  // be delay-loaded. Used for storage by the routine
  // supplied to manage delay-loading.
  uint32 diat; // Relative virtual address of the delay-load import
  // address table. See below for further details.
  uint32 dint; // Relative virtual address of the delay-load name
  // table, which contains the names of the imports
  // that may need to be loaded. Matches the layout of
  // the Import Name Table, Section 6.4.3. Hint/Name Table.
  uint32 dbiat; // Bound Delay Import Table. Relative virtual address
  // of the bound delay-load address table, if it exists.
  uint32 duiat; // Unload Delay Import Table. Relative virtual address
  // of the unload delay-load address table, if it exists.
  // This is an exact copy of the Delay Import Address
  // Table. In the event that the caller unloads the DLL,
  // this table should be copied back over the Delay IAT
  // such that subsequent calls to the DLL continue to
  // use the thunking mechanism correctly.
  qtime32_t datetime; // Time stamp of DLL to which this image has been bound.
};

// Bound Import Table format:

struct BOUND_IMPORT_DESCRIPTOR {
  qtime32_t TimeDateStamp;
  uint16 OffsetModuleName;
  uint16 NumberOfModuleForwarderRefs;
  // Array of zero or more IMAGE_BOUND_FORWARDER_REF follows
};

struct BOUND_FORWARDER_REF {
  qtime32_t TimeDateStamp;
  uint16 OffsetModuleName;
  uint16 Reserved;
};

//--------------------------------------------------------------------------
//
//      T H R E A D   L O C A L   D A T A
//
```

```
struct image_tls_directory64 {
  uint64 StartAddressOfRawData;
  uint64 EndAddressOfRawData;
  uint64 AddressOfIndex;     // PDWORD
  uint64 AddressOfCallBacks; // PIMAGE_TLS_CALLBACK *;
  uint32 SizeOfZeroFill;
  uint32 Characteristics;
};

struct image_tls_directory32 {
  uint32 StartAddressOfRawData;
  uint32 EndAddressOfRawData;
  uint32 AddressOfIndex;     // PDWORD
  uint32 AddressOfCallBacks; // PIMAGE_TLS_CALLBACK *
  uint32 SizeOfZeroFill;
  uint32 Characteristics;
};

//----------------------------------------------------------------------
//
//      Exception Tables (.pdata)
//

// ARM, PowerPC, SH3 and SH4 WindowsCE platforms
struct function_entry_ce {
  uint32 FuncStart;         // Virtual address of the corresponding function.
  uint32 PrologLen : 8;     // Number of instructions in the function's prolog.
  uint32 FuncLen : 22;      // Number of instructions in the function.
  uint32 ThirtyTwoBit : 1;  // Set if the function is comprised of 32-bit
                            // instructions, cleared for a 16-bit function.
  uint32 ExceptionFlag : 1; // Set if an exception handler exists for the
                            // function.
};

// ARMv7
struct function_entry_armv7 {
  uint32 BeginAddress; // The RVA of the corresponding function
  uint32 UnwindInfo;   // The RVA of the unwind information, including function
                       // length.
                       // If the low 2 bits are non-zero, then this word
                       // represents a compacted inline form of the unwind
                       // information, including function length.
};

// for MIPS and 32-bit Alpha
struct function_entry_alpha {
  uint32 BeginAddress;     // Virtual address of the corresponding function.
  uint32 EndAddress;       // Virtual address of the end of the function.
  uint32 ExceptionHandler; // Pointer to the exception handler to be executed.
  uint32 HandlerData; // Pointer to additional information to be passed to the
                      // handler.
  uint32 PrologEndAddress; // Virtual address of the end of the function's
                           // prolog.
};

// x64
typedef enum _UNWIND_OP_CODES {
  UWOP_PUSH_NONVOL = 0,     // info == register number
  UWOP_ALLOC_LARGE = 1,     // alloc size/8 in next 1(info=0) or 2(info=1) slots
  UWOP_ALLOC_SMALL = 2,     // info == size of allocation / 8 - 1
  UWOP_SET_FPREG = 3,       // FP = RSP + UNWIND_INFO.FPRegOffset*16
  UWOP_SAVE_NONVOL = 4,     // info == register number, offset/8 in next slot
  UWOP_SAVE_NONVOL_FAR = 5, // info == register number, offset/8 in next 2 slots
  UWOP_SAVE_XMM = 6, // Version 1: info == XMM reg number, offset/8 in next slot
```

```
  UWOP_EPILOG = 6,   // Version 2; code offset is epilog size;
  UWOP_SAVE_XMM_FAR =
      7, // version 1:info == XMM reg number, offset/8 in next 2 slots
  UWOP_SPARE_CODE =
      7, // unused ("previously 64-bit UWOP_SAVE_XMM_FAR"); skip 2 slots
  UWOP_SAVE_XMM128 = 8,     // info == XMM reg number, offset/16 in next slot
  UWOP_SAVE_XMM128_FAR = 9, // info == XMM reg number, offset/16 in next 2 slots
  UWOP_PUSH_MACHFRAME = 10, // info == 0: no error-code, 1: with error code
} UNWIND_CODE_OPS;

// Define unwind information flags.
//

#define UNW_FLAG_NHANDLER 0x0
#define UNW_FLAG_EHANDLER 0x1
#define UNW_FLAG_UHANDLER 0x2
#define UNW_FLAG_CHAININFO 0x4

//------------------------------------------------------------------------
//
//      F I X U P S
//
struct pefixup_t {
  uint32 page; // The image base plus the page rva is added to each offset
  // to create the virtual address of where the fixup needs to
  // be applied.
  uint32 size; // Number of bytes in the fixup block. This includes the
               // PAGE RVA and SIZE fields.
};

#define PER_OFF 0x0FFF
#define PER_TYPE 0xF000
#define PER_ABS 0x0000  // This is a NOP. The fixup is skipped.
#define PER_HIGH 0x1000 // Add the high 16-bits of the delta to the
// 16-bit field at Offset. The 16-bit field
// represents the high value of a 32-bit word.
#define PER_LOW 0x2000 // Add the low 16-bits of the delta to the
// 16-bit field at Offset. The 16-bit field
// represents the low half value of a
// 32-bit word.  This fixup will only be
// emitted for a RISC machine when the image
// Object Align isn't the default of 64K.
#define PER_HIGHLOW 0x3000 // Apply the 32-bit delta to the 32-bit field
// at Offset.
#define PER_HIGHADJUST 0x4000 // This fixup requires a full 32-bit value.
// The high 16-bits is located at Offset, and
// the low 16-bits is located in the next
// Offset array element (this array element
// is included in the SIZE field). The two
// need to be combined into a signed variable.
// Add the 32-bit delta. Then add 0x8000 and
// store the high 16-bits of the signed
// variable to the 16-bit field at Offset.
#define PER_REL5000 0x5000 // Machine-specific
#define PER_SECTION 0x6000 // Reserved for future use
#define PER_REL32 0x7000   // Relative intrasection
#define PER_REL7000 0x7000 // Machine-specific
#define PER_REL8000 0x8000 // Machine-specific
#define PER_REL9000 0x9000 // Machine-specific
#define PER_DIR64 0xA000   // This fixup applies the delta to the 64-bit
// field at Offset
#define PER_HIGH3ADJ 0xB000 // The fixup adds the high 16 bits of the delta
// to the 16-bit field at Offset. The 16-bit
// field represents the high value of a 48-bit
```

```
// word. The low 32 bits of the 48-bit value are
// stored in the 32-bit word that follows this
// base relocation. This means that this base
// relocation occupies three slots.

// Platform-specific based relocation types.

#define PER_IA64_IMM64 0x9000

#define PER_MIPS_JMPADDR                                                    \
  0x5000 // base relocation applies to a MIPS jump instruction.
#define PER_MIPS_JMPADDR16                                                  \
  0x9000 // base relocation applies to a MIPS16 jump instruction.

#define PER_ARM_MOV32A 0x5000 // base relocation applies the difference to the
// 32-bit value encoded in the immediate fields of
// a contiguous MOVW+MOVT pair in ARM mode at offset.
#define PER_ARM_MOV32T 0x7000 // base relocation applies the difference to the
// 32-bit value encoded in the immediate fields of
// a contiguous MOVW+MOVT pair in Thumb mode at offset.

//-------------------------------------------------------------------------
//
//      DBG file debug entry format
//
struct debug_entry_t {
  uint32 flags; // usually zero
  qtime32_t datetime;
  uint16 major;
  uint16 minor;
  int32 type;
#define DBG_COFF 1
#define DBG_CV 2
#define DBG_FPO 3
#define DBG_MISC 4
#define DBG_EXCEPTION 5
#define DBG_FIXUP 6
#define DBG_OMAP_TO_SRC 7
#define DBG_OMAP_FROM_SRC 8
#define DBG_BORLAND 9
#define DBG_RES10 10
#define DBG_CLSID 11
#define DBG_VCFEATURE 12
#define DBG_POGO 13
#define DBG_ILTCG 14
#define DBG_MPX 15
  uint32 size;
  uint32 rva;  // virtual address
  uint32 seek; // ptr to data in the file
};

// now we can define has_debdir() because we have debug_entry_t defined
template <> inline bool peheader_t::has_debdir() const {
  return debdir.size >= sizeof(debug_entry_t) && debdir.rva != 0;
}

//-------------------------------------------------------------------------
//
//      DBG file COFF debug information header
//
struct coff_debug_t {
  uint32 NumberOfSymbols;
  uint32 LvaToFirstSymbol;
  uint32 NumberOfLinenumbers;
```

```
  uint32 LvaToFirstLinenumber;
  uint32 RvaToFirstByteOfCode;
  uint32 RvaToLastByteOfCode;
  uint32 RvaToFirstByteOfData;
  uint32 RvaToLastByteOfData;
};

//--------------------------------------------------------------------------
//
//      DBG file FPO debug information
//
struct fpo_t {
  uint32 address;
  uint32 size;
  uint32 locals;
  uint16 params;
  uchar prolog;
  uchar regs;
#define FPO_REGS 0x07 // register number
#define FPO_SEH 0x08  //
#define FPO_BP 0x10   // has BP frame?
#define FPO_TYPE 0xC0
#define FPO_T_FPO 0x00
#define FPO_T_TRAP 0x40
#define FPO_T_TSS 0x80
#define FPO_T_NONFPO 0xC0
};

//      DBG file OMAP debug information

struct omap_t {
  uint32 a1;
  uint32 a2;
};

// misc entry format
struct misc_debug_t {
  uint32 type; // type of misc data, see defines
#define MISC_EXENAME 1
  uint32 length; // total length of record, rounded to four
  // byte multiple.
  uchar unicode;     // TRUE if data is unicode string
  uchar reserved[3]; // padding
  uchar data[1];     // Actual data
};

//--------------------------------------------------------------------------
// Resource information
struct rsc_dir_t {
  uint32 Characteristics;
  uint32 TimeDateStamp;
  uint16 MajorVersion;
  uint16 MinorVersion;
  uint16 NumberOfNamedEntries;
  uint16 NumberOfIdEntries;
};

struct rsc_dir_entry_t {
  union {
    struct {
      uint32 NameOffset : 31;
      uint32 NameIsString : 1;
    };
    uint32 Name;
```

```
    uint16 Id;
  };
  union {
    uint32 OffsetToData;
    struct {
      uint32 OffsetToDirectory : 31;
      uint32 DataIsDirectory : 1;
    };
  };
};

struct rsc_data_entry_t {
  uint32 OffsetToData;
  uint32 Size;
  uint32 CodePage;
  uint32 Reserved;
};

// Resource types
#define PE_RT_CURSOR 1
#define PE_RT_BITMAP 2
#define PE_RT_ICON 3
#define PE_RT_MENU 4
#define PE_RT_DIALOG 5
#define PE_RT_STRING 6
#define PE_RT_FONTDIR 7
#define PE_RT_FONT 8
#define PE_RT_ACCELERATOR 9
#define PE_RT_RCDATA 10
#define PE_RT_MESSAGETABLE 11
#define PE_RT_GROUP_CURSOR 12
#define PE_RT_GROUP_ICON 14
#define PE_RT_VERSION 16
#define PE_RT_DLGINCLUDE 17
#define PE_RT_PLUGPLAY 19
#define PE_RT_VXD 20
#define PE_RT_ANICURSOR 21
#define PE_RT_ANIICON 22
#define PE_RT_HTML 23
#define PE_RT_MANIFEST 24

// Language codes
#define PE_LANG_NEUTRAL 0x00
#define PE_LANG_INVARIANT 0x7f

#define PE_LANG_AFRIKAANS 0x36
#define PE_LANG_ALBANIAN 0x1c
#define PE_LANG_ARABIC 0x01
#define PE_LANG_ARMENIAN 0x2b
#define PE_LANG_ASSAMESE 0x4d
#define PE_LANG_AZERI 0x2c
#define PE_LANG_BASQUE 0x2d
#define PE_LANG_BELARUSIAN 0x23
#define PE_LANG_BENGALI 0x45
#define PE_LANG_BULGARIAN 0x02
#define PE_LANG_CATALAN 0x03
#define PE_LANG_CHINESE 0x04
#define PE_LANG_CROATIAN 0x1a
#define PE_LANG_CZECH 0x05
#define PE_LANG_DANISH 0x06
#define PE_LANG_DIVEHI 0x65
#define PE_LANG_DUTCH 0x13
#define PE_LANG_ENGLISH 0x09
#define PE_LANG_ESTONIAN 0x25
```

```
#define PE_LANG_FAEROESE 0x38
#define PE_LANG_FARSI 0x29
#define PE_LANG_FINNISH 0x0b
#define PE_LANG_FRENCH 0x0c
#define PE_LANG_GALICIAN 0x56
#define PE_LANG_GEORGIAN 0x37
#define PE_LANG_GERMAN 0x07
#define PE_LANG_GREEK 0x08
#define PE_LANG_GUJARATI 0x47
#define PE_LANG_HEBREW 0x0d
#define PE_LANG_HINDI 0x39
#define PE_LANG_HUNGARIAN 0x0e
#define PE_LANG_ICELANDIC 0x0f
#define PE_LANG_INDONESIAN 0x21
#define PE_LANG_ITALIAN 0x10
#define PE_LANG_JAPANESE 0x11
#define PE_LANG_KANNADA 0x4b
#define PE_LANG_KASHMIRI 0x60
#define PE_LANG_KAZAK 0x3f
#define PE_LANG_KONKANI 0x57
#define PE_LANG_KOREAN 0x12
#define PE_LANG_KYRGYZ 0x40
#define PE_LANG_LATVIAN 0x26
#define PE_LANG_LITHUANIAN 0x27
#define PE_LANG_MACEDONIAN 0x2f // the Former Yugoslav Republic of Macedonia
#define PE_LANG_MALAY 0x3e
#define PE_LANG_MALAYALAM 0x4c
#define PE_LANG_MANIPURI 0x58
#define PE_LANG_MARATHI 0x4e
#define PE_LANG_MONGOLIAN 0x50
#define PE_LANG_NEPALI 0x61
#define PE_LANG_NORWEGIAN 0x14
#define PE_LANG_ORIYA 0x48
#define PE_LANG_POLISH 0x15
#define PE_LANG_PORTUGUESE 0x16
#define PE_LANG_PUNJABI 0x46
#define PE_LANG_ROMANIAN 0x18
#define PE_LANG_RUSSIAN 0x19
#define PE_LANG_SANSKRIT 0x4f
#define PE_LANG_SINDHI 0x59
#define PE_LANG_SLOVAK 0x1b
#define PE_LANG_SLOVENIAN 0x24
#define PE_LANG_SPANISH 0x0a
#define PE_LANG_SWAHILI 0x41
#define PE_LANG_SWEDISH 0x1d
#define PE_LANG_SYRIAC 0x5a
#define PE_LANG_TAMIL 0x49
#define PE_LANG_TATAR 0x44
#define PE_LANG_TELUGU 0x4a
#define PE_LANG_THAI 0x1e
#define PE_LANG_TURKISH 0x1f
#define PE_LANG_UKRAINIAN 0x22
#define PE_LANG_URDU 0x20
#define PE_LANG_UZBEK 0x43
#define PE_LANG_VIETNAMESE 0x2a

//-------------------------------------------------------------------------

#define PE_NODE "$ PE header" // netnode name for PE header
// value()        -> peheader_t
// altval(segnum) -> s->start_ea
#define PE_ALT_DBG_FPOS                                                 \
  nodeidx_t(-1) // altval() -> translated fpos of debuginfo
#define PE_ALT_IMAGEBASE                                                \
```

```
  nodeidx_t(-2) // altval() -> loading address (usually pe.imagebase)
#define PE_ALT_PEHDR_OFF nodeidx_t(-3) // altval() -> offset of PE header
#define PE_ALT_NEFLAGS nodeidx_t(-4)   // altval() -> neflags
#define PE_ALT_TDS_LOADED                                                \
  nodeidx_t(-5) // altval() -> tds already loaded(1) or invalid(-1)
#define PE_ALT_PSXDLL                                                    \
  nodeidx_t(-6) // altval() -> if POSIX(x86) imports from PSXDLL netnode
#define PE_ALT_OVRVA nodeidx_t(-7)    // altval() -> overlay rva (if present)
#define PE_ALT_OVRSZ nodeidx_t(-8)    // altval() -> overlay size (if present)
#define PE_SUPSTR_PDBNM nodeidx_t(-9) // supstr() -> pdb file name
// supval(segnum) -> pesection_t
// blob(0, PE_NODE_RELOC)  -> relocation info
// blob(0, RSDS_TAG)  -> rsds_t structure
// blob(0, NB10_TAG)  -> cv_info_pdb20_t structure
#define PE_ALT_NTAPI nodeidx_t(-10) // altval() -> uses Native API
#define PE_NODE_RELOC 'r'
#define RSDS_TAG 's'
#define NB10_TAG 'n'
#define UTDS_TAG 't'

struct load_config_t {
  uint32 Size;
  uint32 TimeDateStamp;
  uint16 MajorVersion;
  uint16 MinorVersion;
  uint32 GlobalFlagsClear;
  uint32 GlobalFlagsSet;
  uint32 CriticalSectionDefaultTimeout;
  uint32 DeCommitFreeBlockThreshold;
  uint32 DeCommitTotalFreeThreshold;
  uint32 LockPrefixTable; // VA
  uint32 MaximumAllocationSize;
  uint32 VirtualMemoryThreshold;
  uint32 ProcessHeapFlags;
  uint32 ProcessAffinityMask;
  uint16 CSDVersion;
  uint16 Reserved1;
  uint32 EditList;       // VA
  uint32 SecurityCookie; // VA
  // Version 2
  uint32 SEHandlerTable; // VA
  uint32 SEHandlerCount;
  // Version 3
  uint32 GuardCFCheckFunctionPointer;    // VA
  uint32 GuardCFDispatchFunctionPointer; // VA
  uint32 GuardCFFunctionTable;           // VA
  uint32 GuardCFFunctionCount;
  uint32 GuardFlags;
};

struct load_config64_t {
  uint32 Size;
  uint32 TimeDateStamp;
  uint16 MajorVersion;
  uint16 MinorVersion;
  uint32 GlobalFlagsClear;
  uint32 GlobalFlagsSet;
  uint32 CriticalSectionDefaultTimeout;
  uint64 DeCommitFreeBlockThreshold;
  uint64 DeCommitTotalFreeThreshold;
  uint64 LockPrefixTable; // VA
  uint64 MaximumAllocationSize;
  uint64 VirtualMemoryThreshold;
  uint64 ProcessAffinityMask;
```

```
  uint32 ProcessHeapFlags;
  uint16 CSDVersion;
  uint16 Reserved1;
  uint64 EditList;       // VA
  uint64 SecurityCookie; // VA
  // Version 2
  uint64 SEHandlerTable; // VA
  uint64 SEHandlerCount;
  // Version 3
  uint64 GuardCFCheckFunctionPointer;    // VA
  uint64 GuardCFDispatchFunctionPointer; // VA
  uint64 GuardCFFunctionTable;           // VA
  uint64 GuardCFFunctionCount;
  uint32 GuardFlags;
};

#ifndef IMAGE_GUARD_CF_INSTRUMENTED
#define IMAGE_GUARD_CF_INSTRUMENTED                                         \
  0x000000100 // Module performs control flow integrity checks using
              // system-supplied support
#define IMAGE_GUARD_CFW_INSTRUMENTED                                        \
  0x000000200 // Module performs control flow and write integrity checks
#define IMAGE_GUARD_CF_FUNCTION_TABLE_PRESENT                               \
  0x000000400 // Module contains valid control flow target metadata
#define IMAGE_GUARD_SECURITY_COOKIE_UNUSED                                  \
  0x000000800 // Module does not make use of the /GS security cookie
#define IMAGE_GUARD_PROTECT_DELAYLOAD_IAT                                   \
  0x00001000 // Module supports read only delay load IAT
#define IMAGE_GUARD_DELAYLOAD_IAT_IN_ITS_OWN_SECTION                        \
  0x00002000 // Delayload import table in its own .didat section (with nothing
             // else in it) that can be freely reprotected
#define IMAGE_GUARD_CF_FUNCTION_TABLE_SIZE_MASK                             \
  0xF0000000 // Stride of Guard CF function table encoded in these bits
             // (additional count of bytes per element)
#define IMAGE_GUARD_CF_FUNCTION_TABLE_SIZE_SHIFT                            \
  28 // Shift to right-justify Guard CF function table stride
#endif

//-------------------------------------------------------------------------
// MS Windows CLSID, GUID
struct clsid_t {
  uint32 id1;
  uint16 id2;
  uint16 id3;
  uchar id4[8];
  bool operator==(const struct clsid_t &r) const {
    return memcmp(this, &r, sizeof(r)) == 0;
  }
};

//-------------------------------------------------------------------------
// RSDS debug information
struct rsds_t {
  uint32 magic;
#define RSDS_MAGIC MC4('R', 'S', 'D', 'S')
#define UTDS_MAGIC MC4('u', 'T', 'D', 'S')
  clsid_t guid;
  uint32 age;
  //  char name[];  // followed by a zero-terminated UTF8 file name
};

//-------------------------------------------------------------------------
// NB10 debug information
struct cv_info_pdb20_t {
```

```
  uint32 magic; // 'NB10'
#define NB10_MAGIC MC4('N', 'B', '1', '0')
  uint32 offset;
  uint32 signature;
  uint32 age;
  // char pdb_file_name[];
};

//-------------------------------------------------------------------------
// MTOC debug information.
// denotes EFI binaries that were built on OSX as Mach-O, then converted to PE
// by the 'mtoc' utility. see
// https://opensource.apple.com/source/cctools/cctools-921/efitools/mtoc.c.auto.html
struct mtoc_info_t {
  uint32 magic; // 'MTOC'
#define MTOC_MAGIC MC4('M', 'T', 'O', 'C')
  uchar uuid[16]; // UUID of original Mach-O file
                  // char debug_filename[];
};

// TE (Terse Executable)
struct teheader_t {
  uint16 signature; // 00
  uint16 machine;   // 02 same as in PE

  bool is_64bit_cpu(void) const {
    return machine == PECPU_AMD64 || machine == PECPU_IA64 ||
           machine == PECPU_ARM64;
  }

  uint8 nobjs;         // 04 number of sections
  uint8 subsys;        // 05 target subsystem
  uint16 strippedsize; // 06 number of bytes removed from the base of the
                       // original image

  int32 first_section_pos(int32 peoff) const {
    return peoff + sizeof(teheader_t);
  }

  // value which should be added to the sections' file offsets and RVAs
  int32 te_adjust() const { return sizeof(teheader_t) - strippedsize; }

  uint32 entry;       // 08 Entry point
  uint32 text_start;  // 0C Base of code
  uint64 imagebase64; // 10 Virtual base of the image.
  uint64 imagebase() const { return imagebase64; }
  petab_t reltab; // 18 Relocation Table
  petab_t debdir; // 20 Debug Directory
};

const char *get_pe_machine_name(uint16 machine);
void print_pe_flags(uint16 flags);

#pragma pack(pop)
}
```

efiXloader/pe_manager.h

```
/*
 * efiXloader
```

```
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include <string>

#include "ida_core.h"
#include "pe.h"

namespace efiloader {
class PeManager {
public:
  explicit PeManager(uint16_t mt) {
    inf_set_64bit();
    set_imagebase(0x0);
    if (mt == PECPU_ARM64) {
      set_processor_type("arm", SETPROC_LOADER);
    } else {
      set_processor_type("metapc", SETPROC_LOADER);
    }
    pe_base = 0;
    pe_sel_base = 0;
    machine_type = mt;
  }
  void process(linput_t *li, std::basic_string<char> fname, int ord);
  uint16_t machine_type;

private:
  void to_base(linput_t *);
  efiloader::PE *pe;
  qvector<efiloader::PE *> pe_files;
  ushort pe_sel_base;
  ea_t pe_base;
  // head processing
  void pe_head_to_base(linput_t *li);
};
} // namespace efiloader
}
```

efiXloader/uefitool.h

```
/*
 * efiXloader
 * Copyright (C) 2020-2024 Binarly
 *
```

```
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include <set>
#ifdef _WIN32
#include <direct.h>
#else
#include <sys/stat.h>
#endif
#include <codecvt>
#include <filesystem>
#include <string>
#include <utility>
#include <vector>

#include "3rd/uefitool/common/LZMA/LzmaCompress.h"
#include "3rd/uefitool/common/LZMA/LzmaDecompress.h"
#include "3rd/uefitool/common/Tiano/EfiTianoCompress.h"
#include "3rd/uefitool/common/Tiano/EfiTianoDecompress.h"
#include "3rd/uefitool/common/basetypes.h"
#include "3rd/uefitool/common/ffs.h"
#include "3rd/uefitool/common/ffsparser.h"
#include "3rd/uefitool/common/ffsreport.h"
#include "3rd/uefitool/common/filesystem.h"
#include "3rd/uefitool/common/guiddatabase.h"
#include "3rd/uefitool/common/treeitem.h"
#include "3rd/uefitool/common/treemodel.h"
#include "3rd/uefitool/common/ustring.h"
#include "3rd/uefitool/version.h"

#include "3rd/uefitool/UEFIExtract/ffsdumper.h"
#include "3rd/uefitool/UEFIExtract/uefidump.h"
#include "fstream"
#include "json.hpp"

#include "ida_core.h"

using nlohmann::json;

enum FILE_SECTION_TYPE {
  PE_DEPENDENCY_SECTION = 0,
  PE_TE_IMAGE_SECTION = 1,
  UI_SECTION = 2,
  VERSION_SECTION = 3
};

namespace efiloader {
class File {
public:
  File() {}
```

```
  void set_data(char *data_in, uint32_t size_in) {
    qname.qclear();
    bytes.resize(size_in);
    memcpy(&bytes[0], data_in, size_in);
  }
  void write() {
    qstring idb_path(get_path(PATH_TYPE_IDB));
    qstring images_path = idb_path + qstring(".efiloader");
#ifdef WIN32
    _mkdir(images_path.c_str());
#else
    mkdir(images_path.c_str(), S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
#endif
    if (!qname.empty()) {
      qstring image_path = images_path + qstring("/") + qstring(qname.c_str());
      std::ofstream file;
      file.open(image_path.c_str(), std::ios::out | std::ios::binary);
      file.write(ubytes.constData(), ubytes.size());
      file.close();
      dump_name.swap(image_path);
    }
  }
  void print();
  UByteArray ubytes;
  UByteArray uname;
  bytevec_t bytes;
  char *data = NULL;
  uint32_t size = 0;
  std::string name_utf8;
  std::string name_utf16;
  qstring qname;
  qstring dump_name;
  bool is_pe = false;
  bool is_te = false;
  bool has_ui = false;
};

class Uefitool {
public:
  explicit Uefitool(bytevec_t &data) {
    buffer = (const char *)&data[0];
    buffer_size = data.size();
    UByteArray ubuffer(buffer, buffer_size);
    FfsParser ffs(&model);
    if (ffs.parse(ubuffer)) {
      loader_failure("failed to parse data via UEFITool");
    }
    messages = ffs.getMessages();
  }
  ~Uefitool() {}
  void show_messages();
  bool messages_occurs() { return !messages.empty(); }
  void dump();
  void dump(const UModelIndex &index);
  void dump(const UModelIndex &index, uint8_t el_type, File *pe_file);
  void handle_raw_section(const UModelIndex &index);
  bool is_pe_index(const UModelIndex &index) {
    return model.rowCount(index) == 4;
  }
  bool is_file_index(const UModelIndex &index) {
    return model.type(index) == Types::File;
  }
  void get_unique_name(qstring &image_name);
  void get_image_guid(qstring &image_guid, UModelIndex index);
```

```
  std::vector<std::string> parse_depex_section_body(const UModelIndex &index,
                                                    UString &parsed);
  std::vector<std::string> parse_apriori_raw_section(const UModelIndex &index);
  void get_deps(UModelIndex index, std::string key);
  void get_apriori(UModelIndex index, std::string key);
  void dump_jsons();

  // DEPEX information for each image
  json all_deps;

  json images_guids;
  TreeModel model;
  const char *buffer;
  uint32_t buffer_size;
  std::vector<std::pair<UString, UModelIndex>> messages;
  std::set<qstring> unique_names;
  std::vector<efiloader::File *> files;
  USTATUS err;
  void set_machine_type(UByteArray pe_body);
  uint16_t machine_type = 0xffff;
  bool machine_type_detected = false;

private:
  std::string get_kind(const UModelIndex &index) {
    return fileTypeToUString(model.subtype(index.parent())).toLocal8Bit();
  }
};
} // namespace efiloader
}
```

efiXloader/utils.h

```
/*
 * efiXloader
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include <algorithm>
#include <cstdint>
#include <string>
#include <vector>

#include "ida_core.h"
```

```
namespace efiloader {

class Utils {
public:
  Utils() {}
  void show_hex(void *buffer, size_t length, const char *prefix);
  bool find_vol(bytevec_t &frm, std::string &sig, qoff64_t &vol_off);
  qoff64_t find_vol_new(linput_t *li, char *sig);
  qoff64_t find_vol_test(bytevec_t &data);
  void skip(memory_deserializer_t *ser, size_t size, size_t count);
};
} // namespace efiloader
}
```

Chapter 2.0.0

efiXplorer

efiXplorer/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.7)

project(efiXplorer CXX)

set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_EXPORT_COMPILE_COMMANDS ON)

if(APPLE)
  # to build Mach-O universal binaries with 2 architectures
  set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -fPIC -arch x86_64 -arch arm64")
else()
  set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -fPIC")
endif()

if(CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  set(CMAKE_CXX_FLAGS
      "${CMAKE_CXX_FLAGS} -Wno-nullability-completeness -Wno-varargs")
endif()

list(APPEND CMAKE_MODULE_PATH ${PROJECT_SOURCE_DIR}/../cmake)

find_package(IdaSdk REQUIRED)

include_directories(${PROJECT_SOURCE_DIR})
include_directories(${PROJECT_SOURCE_DIR}/3rd/nlohmann_json)

# efiXplorer sources
set(efiXplorer_src
    "efi_defs.h"
    "efi_defs.cc"
    "efi_analysis.h"
    "efi_analysis_arm.cc"
    "efi_analysis_x86.cc"
```

```
    "efi_deps.cc"
    "efi_deps.h"
    "efi_global.cc"
    "efi_global.h"
    "efi_smm_utils.cc"
    "efi_smm_utils.h"
    "efi_ui.cc"
    "efi_ui.h"
    "efi_utils.cc"
    "efi_utils.h"
    "efixplorer.cc")

if(HexRaysSdk_ROOT_DIR)
  add_definitions(-DHEX_RAYS=1)
  set(HexRaysSdk_INCLUDE_DIRS ${HexRaysSdk_ROOT_DIR}/include)
  include_directories(${HexRaysSdk_INCLUDE_DIRS})
  list(APPEND efiXplorer_src "efi_hexrays.cc" "efi_hexrays.h")
endif()

add_ida_plugin(efiXplorer ${PROJECT_SOURCE_DIR}/efixplorer.cc)

set_ida_target_properties(efiXplorer PROPERTIES CXX_STANDARD 20)
ida_target_include_directories(efiXplorer PRIVATE ${IdaSdk_INCLUDE_DIRS})

add_ida_library(efiXplorer_lib ${efiXplorer_src})
ida_target_link_libraries(efiXplorer efiXplorer_lib)
}
```

efiXplorer/efi_analysis.h

```
/*
 * efiXplorer
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include "efi_defs.h"
#include "efi_smm_utils.h"
#include "efi_utils.h"
#ifdef HEX_RAYS
#include "efi_hexrays.h"
#endif
#include <map>
#include <string>
```

```
namespace efi_analysis {

class efi_analyser_t {
public:
  efi_analyser_t();
  ~efi_analyser_t();

  ea_list_t m_bs_list;   // gBS list (boot services addresses)
  ea_list_t m_rt_list;   // gRT list (runtime services addresses)
  ea_list_t m_smst_list; // gSmst list (SMM system table addresses)
  ea_list_t m_st_list;   // gST list (system table addresses)

  ea_list_t m_funcs;

  func_list_t m_smi_handlers;
  json_list_t m_all_guids;
  json_list_t m_all_ppis;
  json_list_t m_all_protocols;
  json_list_t m_all_services;
  json_list_t m_nvram_variables;

  // all .text and .data segments
  // for compatibility with the efixloader
  segment_list_t m_code_segs;
  segment_list_t m_data_segs;

  arch_file_type_t m_arch = arch_file_type_t::unsupported;
  ffs_file_type_t m_ftype = ffs_file_type_t::unsupported;

  void dump_json();
  void get_segments();
  void set_pvalues();
  void annotate_protocol_guids();
  void annotate_data_guids();
  bool smm_cpu_protocol_resolver();
  void find_smi_handlers();
  bool analyse_nvram_variables();

protected:
  ea_t m_start_addr = 0;
  ea_t m_end_addr = 0;

  // a map to look up a GUID name by value
  std::map<json, std::string> m_guiddb_map;
  json m_guiddb;

  std::filesystem::path m_guids_json_path;

  // protocol related boot services
  string_list_t m_prot_bs_names = {"InstallProtocolInterface",
                                   "ReinstallProtocolInterface",
                                   "UninstallProtocolInterface",
                                   "HandleProtocol",
                                   "RegisterProtocolNotify",
                                   "OpenProtocol",
                                   "CloseProtocol",
                                   "OpenProtocolInformation",
                                   "ProtocolsPerHandle",
                                   "LocateHandleBuffer",
                                   "LocateProtocol",
                                   "InstallMultipleProtocolInterfaces",
                                   "UninstallMultipleProtocolInterfaces"};

  // protocol related SMM services
```

```
  string_list_t m_prot_smms_names = {"SmmInstallProtocolInterface",
                                     "SmmUninstallProtocolInterface",
                                     "SmmHandleProtocol",
                                     "SmmRegisterProtocolNotify",
                                     "SmmLocateHandle",
                                     "SmmLocateProtocol"};

  // protocol related PEI services
  string_list_t m_ppi_peis_names = {"InstallPpi", "ReInstallPpi", "LocatePpi",
                                    "NotifyPpi"};

  // data and stack guids
  json_list_t m_data_guids;
  json_list_t m_stack_guids;

  ea_list_t m_annotated_protocols;
  json m_boot_services;
  json m_pei_services_all;
  json m_ppi_calls_all;
  json m_runtime_services_all;
  json m_smm_services_all;
  json m_smm_services;

  ea_list_t m_image_handle_list; // gImageHandle list (image handle addresses)
  ea_list_t m_runtime_services_list; // runtime services list

  // for SMM callout scanners
  ea_list_t m_callout_addrs;
  ea_list_t m_read_save_state_calls;
  func_list_t m_child_smi_handlers;
  func_list_t m_exc_funcs;

  // for double GetVariable scanners
  ea_list_t m_double_get_variable_pei;
  ea_list_t m_double_get_variable_smm;
  ea_list_t m_double_get_variable;

  bool add_protocol(std::string service_name, ea_t guid_addr, ea_t xref_addr,
                    ea_t call_addr);

private:
  // format dependent settings
  // protocols for DXE/SMM PPIs for PEI
  json_list_t *m_ptable;
  std::string m_pname;
  std::string m_pkey;

  uint64_list_t m_ppi_flags = {
      0x1,        0x10,       0x11,       0x20,       0x21,       0x30,
      0x31,       0x40,       0x41,       0x50,       0x51,       0x60,
      0x61,       0x70,       0x71,       0x80000000, 0x80000001, 0x80000010,
      0x80000011, 0x80000020, 0x80000021, 0x80000030, 0x80000031, 0x80000040,
      0x80000041, 0x80000050, 0x80000051, 0x80000060, 0x80000061, 0x80000070,
      0x80000071,
  };

  // EFI_SMM_SW_DISPATCH2_PROTOCOL_GUID
  efi_guid_t m_sw_guid2 = {0x18a3c6dc,
                           0x5eea,
                           0x48c8,
                           {0xa1, 0xc1, 0xb5, 0x33, 0x89, 0xf9, 0x89, 0x99}};
  // EFI_SMM_SW_DISPATCH_PROTOCOL_GUID
  efi_guid_t m_sw_guid = {0xe541b773,
                          0xdd11,
```

```
                        0x420c,
                        {0xb0, 0x26, 0xdf, 0x99, 0x36, 0x53, 0xf8, 0xbf}};
// EFI_SMM_SX_DISPATCH2_PROTOCOL_GUID
efi_guid_t m_sx_guid2 = {0x456d2859,
                         0xa84b,
                         0x4e47,
                         {0xa2, 0xee, 0x32, 0x76, 0xd8, 0x86, 0x99, 0x7d}};
// EFI_SMM_SX_DISPATCH_PROTOCOL_GUID
efi_guid_t m_sx_guid = {0x14FC52BE,
                        0x01DC,
                        0x426C,
                        {0x91, 0xAE, 0xA2, 0x3C, 0x3E, 0x22, 0x0A, 0xE8}};
// EFI_SMM_IO_TRAP_DISPATCH2_PROTOCOL_GUID
efi_guid_t m_io_trap_guid2 = {
    0x58DC368D,
    0x7BFA,
    0x4E77,
    {0xAB, 0xBC, 0x0E, 0x29, 0x41, 0x8D, 0xF9, 0x30}};
// EFI_SMM_IO_TRAP_DISPATCH_PROTOCOL_GUID
efi_guid_t m_io_trap_guid = {
    0xDB7F536B,
    0xEDE4,
    0x4714,
    {0xA5, 0xC8, 0xE3, 0x46, 0xEB, 0xAA, 0x20, 0x1D}};
// EFI_SMM_GPI_DISPATCH2_PROTOCOL_GUID
efi_guid_t m_gpi_guid2 = {0x25566B03,
                          0xB577,
                          0x4CBF,
                          {0x95, 0x8C, 0xED, 0x66, 0x3E, 0xA2, 0x43, 0x80}};
// EFI_SMM_GPI_DISPATCH_PROTOCOL_GUID
efi_guid_t m_gpi_guid = {0xE0744B81,
                         0x9513,
                         0x49CD,
                         {0x8C, 0xEA, 0xE9, 0x24, 0x5E, 0x70, 0x39, 0xDA}};
// EFI_SMM_USB_DISPATCH2_PROTOCOL_GUID
efi_guid_t m_usb_guid2 = {0xEE9B8D90,
                          0xC5A6,
                          0x40A2,
                          {0xBD, 0xE2, 0x52, 0x55, 0x8D, 0x33, 0xCC, 0xA1}};
// EFI_SMM_USB_DISPATCH_PROTOCOL_GUID
efi_guid_t m_usb_guid = {0xA05B6FFD,
                         0x87AF,
                         0x4E42,
                         {0x95, 0xC9, 0x62, 0x28, 0xB6, 0x3C, 0xF3, 0xF3}};
// EFI_SMM_STANDBY_BUTTON_DISPATCH2_PROTOCOL_GUID
efi_guid_t m_standby_button_guid2 = {
    0x7300C4A1,
    0x43F2,
    0x4017,
    {0xA5, 0x1B, 0xC8, 0x1A, 0x7F, 0x40, 0x58, 0x5B}};
// EFI_SMM_STANDBY_BUTTON_DISPATCH_PROTOCOL_GUID
efi_guid_t m_standby_button_guid = {
    0x78965B98,
    0xB0BF,
    0x449E,
    {0x8B, 0x22, 0xD2, 0x91, 0x4E, 0x49, 0x8A, 0x98}};
// EFI_SMM_PERIODIC_TIMER_DISPATCH2_PROTOCOL_GUID
efi_guid_t m_periodic_timer_guid2 = {
    0x4CEC368E,
    0x8E8E,
    0x4D71,
    {0x8B, 0xE1, 0x95, 0x8C, 0x45, 0xFC, 0x8A, 0x53}};
// EFI_SMM_PERIODIC_TIMER_DISPATCH_PROTOCOL_GUID
efi_guid_t m_periodic_timer_guid = {
```

```
    0x9CCA03FC,
    0x4C9E,
    0x4A19,
    {0x9B, 0x06, 0xED, 0x7B, 0x47, 0x9B, 0xDE, 0x55}};
// EFI_SMM_POWER_BUTTON_DISPATCH2_PROTOCOL_GUID
efi_guid_t m_power_button_guid2 = {
    0x1B1183FA,
    0x1823,
    0x46A7,
    {0x88, 0x72, 0x9C, 0x57, 0x87, 0x55, 0x40, 0x9D}};
// EFI_SMM_POWER_BUTTON_DISPATCH_PROTOCOL_GUID
efi_guid_t m_power_button_guid = {
    0xB709EFA0,
    0x47A6,
    0x4B41,
    {0xB9, 0x31, 0x12, 0xEC, 0xE7, 0xA8, 0xEE, 0x56}};
// EFI_SMM_ICHN_DISPATCH_PROTOCOL_GUID
efi_guid_t m_ichn_guid = {0xC50B323E,
                          0x9075,
                          0x4F2A,
                          {0xAC, 0x8E, 0xD2, 0x59, 0x6A, 0x10, 0x85, 0xCC}};
// EFI_SMM_ICHN_DISPATCH2_PROTOCOL_GUID
efi_guid_t m_ichn_guid2 = {0xADF3A128,
                           0x416D,
                           0x4060,
                           {0x8D, 0xDF, 0x30, 0xA1, 0xD7, 0xAA, 0xB6, 0x99}};
// PCH_TCO_SMI_DISPATCH_PROTOCOL_GUID
efi_guid_t m_tco_guid = {0x9E71D609,
                         0x6D24,
                         0x47FD,
                         {0xB5, 0x72, 0x61, 0x40, 0xF8, 0xD9, 0xC2, 0xA4}};
// PCH_PCIE_SMI_DISPATCH_PROTOCOL_GUID
efi_guid_t m_pcie_guid = {0x3E7D2B56,
                          0x3F47,
                          0x42AA,
                          {0x8F, 0x6B, 0x22, 0xF5, 0x19, 0x81, 0x8D, 0xAB}};
// PCH_ACPI_SMI_DISPATCH_PROTOCOL_GUID
efi_guid_t m_acpi_guid = {0xD52BB262,
                          0xF022,
                          0x49EC,
                          {0x86, 0xD2, 0x7A, 0x29, 0x3A, 0x7A, 0x5, 0x4B}};
// PCH_GPIO_UNLOCK_SMI_DISPATCH_PROTOCOL_GUID
efi_guid_t m_gpio_unlock_guid = {
    0x83339EF7,
    0x9392,
    0x4716,
    {0x8D, 0x3A, 0xD1, 0xFC, 0x67, 0xCD, 0x55, 0xDB}};
// PCH_SMI_DISPATCH_PROTOCOL_GUID
efi_guid_t m_pch_guid = {0xE6A81BBF,
                         0x873D,
                         0x47FD,
                         {0xB6, 0xBE, 0x61, 0xB3, 0xE5, 0x72, 0x9, 0x93}};
// PCH_ESPI_SMI_DISPATCH_PROTOCOL_GUID
efi_guid_t m_espi_guid = {0xB3C14FF3,
                          0xBAE8,
                          0x456C,
                          {0x86, 0x31, 0x27, 0xFE, 0x0C, 0xEB, 0x34, 0x0C}};
// EFI_ACPI_EN_DISPATCH_PROTOCOL_GUID
efi_guid_t m_acpi_en_guid = {
    0xBD88EC68,
    0xEBE4,
    0x4F7B,
    {0x93, 0x5A, 0x4F, 0x66, 0x66, 0x42, 0xE7, 0x5F}};
// EFI_ACPI_DIS_DISPATCH_PROTOCOL_GUID
```

```
    efi_guid_t m_acpi_dis_guid = {
        0x9C939BA6,
        0x1FCC,
        0x46F6,
        {0xB4, 0xE1, 0x10, 0x2D, 0xBE, 0x18, 0x65, 0x67}};
    // FCH_SMM_GPI_DISPATCH2_PROTOCOL_GUID
    efi_guid_t m_fch_gpi_guid2 = {
        0x7051ab6d,
        0x9ec2,
        0x42eb,
        {0xa2, 0x13, 0xde, 0x48, 0x81, 0xf1, 0xf7, 0x87}};
    // FCH_SMM_IO_TRAP_DISPATCH2_PROTOCOL_GUID
    efi_guid_t m_fch_io_trap_guid2 = {
        0x91288fc4,
        0xe64b,
        0x4ef9,
        {0xa4, 0x63, 0x66, 0x88, 0x0, 0x71, 0x7f, 0xca}};
    // FCH_SMM_PERIODICAL_DISPATCH2_PROTOCOL_GUID
    efi_guid_t m_fch_periodical_guid2 = {
        0x736102f1,
        0x9584,
        0x44e7,
        {0x82, 0x8a, 0x43, 0x4b, 0x1e, 0x67, 0x5c, 0xc4}};
    // FCH_SMM_PWR_BTN_DISPATCH2_PROTOCOL_GUID
    efi_guid_t m_fch_pwr_btn_guid2 = {
        0xa365240e,
        0x56b0,
        0x426d,
        {0x83, 0xa, 0x30, 0x66, 0xc6, 0x81, 0xbe, 0x9a}};
    // FCH_SMM_SW_DISPATCH2_PROTOCOL_GUID
    efi_guid_t m_fch_sw_guid2 = {
        0x881b4ab6,
        0x17b0,
        0x4bdf,
        {0x88, 0xe2, 0xd4, 0x29, 0xda, 0x42, 0x5f, 0xfd}};
    // FCH_SMM_SX_DISPATCH2_PROTOCOL_GUID
    efi_guid_t m_fch_sx_guid2 = {
        0x87e2a6cf,
        0x91fb,
        0x4581,
        {0x90, 0xa9, 0x6f, 0x50, 0x5d, 0xdc, 0x1c, 0xb2}};
    // FCH_SMM_USB_DISPATCH_PROTOCOL_GUID
    efi_guid_t m_fch_usb_guid = {0x59053b0d,
                                 0xeeb8,
                                 0x4379,
                                 {0xb1, 0xc8, 0x14, 0x5f, 0x1b, 0xb, 0xe4, 0xb9}};
    // FCH_SMM_USB_DISPATCH2_PROTOCOL_GUID
    efi_guid_t m_fch_usb_guid2 = {
        0xfbbb2ea9,
        0xce0e,
        0x4689,
        {0xb3, 0xf0, 0xc6, 0xb8, 0xf0, 0x76, 0xbd, 0x20}};
    // FCH_SMM_MISC_DISPATCH_PROTOCOL_GUID
    efi_guid_t m_fch_misc_guid = {
        0x13bd659b,
        0xb4c6,
        0x47da,
        {0x9b, 0x22, 0x11, 0x50, 0xd4, 0xf3, 0xb, 0xda}};
    // FCH_SMM_APU_RAS_DISPATCH_PROTOCOL_GUID
    efi_guid_t m_fch_apu_ras_guid = {
        0xf871ee59,
        0x29d2,
        0x4b15,
        {0x9e, 0x67, 0xaf, 0x32, 0xcd, 0xc1, 0x41, 0x73}};
```

```
  void print_interfaces();
  bool analyse_variable_service(ea_t ea, std::string service_str);
};

class efi_analyser_x86_t : public efi_analyser_t {
public:
  efi_analyser_x86_t() : efi_analyser_t() {
    // import necessary types
    const til_t *idati = get_idati();
    import_type(idati, -1, "EFI_GUID");
    import_type(idati, -1, "EFI_HANDLE");
    import_type(idati, -1, "EFI_SYSTEM_TABLE");
    import_type(idati, -1, "EFI_BOOT_SERVICES");
    import_type(idati, -1, "EFI_RUNTIME_SERVICES");
    import_type(idati, -1, "_EFI_SMM_SYSTEM_TABLE2");
    import_type(idati, -1, "EFI_PEI_SERVICES");
    import_type(idati, -1, "EFI_PEI_READ_ONLY_VARIABLE2_PPI");
    import_type(idati, -1, "EFI_SMM_VARIABLE_PROTOCOL");
#ifdef HEX_RAYS
    for (auto idx = 0; idx < get_entry_qty(); idx++) {
      uval_t ord = get_entry_ordinal(idx);
      ea_t ep = get_entry(ord);
      efi_hexrays::track_entry_params(get_func(ep), 0);
    }
#endif
  }

  bool find_boot_services_tables();
  bool find_double_get_variable_pei();
  bool find_double_get_variable_smm();
  bool find_double_get_variable();
  bool find_image_handle64();
  bool find_runtime_services_tables();
  bool find_smm_callout();
  bool find_smst_postproc64();
  bool find_smst64();
  bool find_system_table64();
  void find_local_guids64();
  void find_other_boot_services_tables64();
  void get_boot_services_all();
  void get_bs_prot_names32();
  void get_bs_prot_names64();
  void get_pei_services_all32();
  void get_ppi_names32();
  void get_prot_boot_services32();
  void get_prot_boot_services64();
  void get_runtime_services_all();
  void get_smm_prot_names64();
  void get_smm_services_all64();
  void get_variable_ppi_calls_all32();
  void show_all_choosers();

private:
  void find_callout_rec(func_t *func);
  bool install_multiple_prot_interfaces_analyser();
};

class efi_analyser_arm_t : public efi_analyser_t {
public:
  efi_analyser_arm_t() : efi_analyser_t() {
    // in order to make it work, it is necessary to copy
    // uefi.til, uefi64.til files in {idadir}/til/arm/
    add_til("uefi64.til", ADDTIL_DEFAULT);
```

```
    const til_t *idati = get_idati();
    import_type(idati, -1, "EFI_GUID");
    import_type(idati, -1, "EFI_HANDLE");
    import_type(idati, -1, "EFI_SYSTEM_TABLE");
    import_type(idati, -1, "EFI_BOOT_SERVICES");
    import_type(idati, -1, "EFI_RUNTIME_SERVICES");
  }

  void find_boot_services_tables();
  void find_pei_services_function();
  void fix_offsets();
  void initial_analysis();
  void initial_gvars_detection();
  void detect_protocols_all();
  void detect_services_all();
  void show_all_choosers();

private:
  bool get_protocol(ea_t address, uint32_t p_reg, std::string service_name);
};

bool efi_analyse_main_x86_64();
bool efi_analyse_main_x86_32();
bool efi_analyse_main_aarch64();
}; // namespace efi_analysis
}
```

efiXplorer/efi_defs.h

```
/*
 * efiXplorer
 * Copyright (C) 2020-2024  Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include <cinttypes>
#include <cstddef>
#include <cstdint>
#include <fstream>
#include <set>
#include <string>
#include <vector>

#include <allins.hpp>
```

```
#include <auto.hpp>
#include <bytes.hpp>
#include <diskio.hpp>
#include <entry.hpp>
#include <frame.hpp>
#include <graph.hpp>
#include <ida.hpp>
#include <idp.hpp>
#include <kernwin.hpp>
#include <lines.hpp>
#include <loader.hpp>
#include <name.hpp>
#include <pro.h>
#if IDA_SDK_VERSION < 900
#include <struct.hpp>
#endif
#include <typeinf.hpp>
#ifdef HEX_RAYS
#include <hexrays.hpp>
#endif

#include <json.hpp>

using nlohmann::json;

#define COPYRIGHT                                                                \
  "(C) 2020-2024  Binarly - https://github.com/binarly-io/efiXplorer"

#define BTOA(x) ((x) ? "true" : "false")

#define VZ 0x5A56
#define MZ 0x5A4D

#define BS_OFFSET_64 0x60
#define BS_OFFSET_32 0x3c
#define RT_OFFSET_64 0x58
#define RT_OFFSET_32 0x38

#define NONE_REG 0xffff
#define NONE_OFFSET 0xffff
#define NONE_PUSH 0xffff

enum class arch_file_type_t { unsupported, x86_32, x86_64, aarch64, uefi };
enum class ffs_file_type_t { unsupported = 0, pei = 6, dxe_smm = 7 };
enum class module_type_t { dxe_smm = 0, pei = 1 };

enum machine_type_t { AMD64 = 0x8664, I386 = 0x014C, AARCH64 = 0xaa64 };

enum regs_x86_32_t {
  R_EAX,
  R_ECX,
  R_EDX,
  R_EBX,
  R_ESP,
  R_EBP,
  R_ESI,
  R_EDI,
  R_AL = 0x10,
  R_DL = 0x12
};

enum regs_x86_64_t {
  R_RAX,
  R_RCX,
```

```
  R_RDX,
  R_RBX,
  R_RSP,
  R_RBP,
  R_RSI,
  R_RDI,
  R_R8,
  R_R9,
  R_R10,
  R_R11,
  R_R12,
  R_R13,
  R_R14,
};

enum regs_aarch64_t {
  R_C0 = 0,
  R_C13 = 13,
  R_X0 = 129,
  R_X1,
  R_X2,
  R_X3,
  R_X4,
  R_X5,
  R_X6,
  R_X7,
  R_X8,
  R_X9,
  R_X10,
  R_X11,
  R_X12,
  R_X13,
  R_X14,
  R_X15,
  R_X16,
  R_X17,
  R_X18,
  R_X19,
  R_X20,
  R_X21,
  R_X22,
  R_X23,
  R_X24,
  R_X25,
  R_X26,
  R_X27,
  R_X28,
  R_X29,
  R_X30,
  R_XZR,
  R_XSP,
  R_XPC,
};

struct service_info_64_t {
  char name[64];
  uint32_t offset;
  uint32_t reg;
  uint16_t arg_index;
};

struct service_info_32_t {
  char name[64];
  uint32_t offset;
```

```
  uint16_t push_number;
};

struct service_t {
  char name[64];
  uint32_t offset64;
  uint32_t offset32;
};

using ea_list_t = std::vector<ea_t>;
using func_list_t = std::vector<func_t *>;
using json_list_t = std::vector<json>;
using segment_list_t = std::vector<segment_t *>;
using string_list_t = std::vector<std::string>;
using string_set_t = std::set<std::string>;
using uchar_list_t = std::vector<uchar>;
using uint64_list_t = std::vector<uint64_t>;
using uint8_list_t = std::vector<uint8_t>;

struct efi_guid_t {
  uint32_t data1;
  uint16_t data2;
  uint16_t data3;
  uint8_t data4[8];

  uchar_list_t uchar_data() {
    uchar_list_t res;
    res.push_back(data1 & 0xff);
    res.push_back(data1 >> 8 & 0xff);
    res.push_back(data1 >> 16 & 0xff);
    res.push_back(data1 >> 24 & 0xff);
    res.push_back(data2 & 0xff);
    res.push_back(data2 >> 8 & 0xff);
    res.push_back(data3 & 0xff);
    res.push_back(data3 >> 8 & 0xff);
    for (auto i = 0; i < 8; i++) {
      res.push_back(data4[i]);
    }
    return res;
  }

  std::string to_string() const {
    char guid_str[37] = {0};
    snprintf(guid_str, sizeof(guid_str),
             "%08X-%04X-%04X-%02X%02X-%02X%02X%02X%02X%02X%02X", data1, data2,
             data3, data4[0], data4[1], data4[2], data4[3], data4[4], data4[5],
             data4[6], data4[7]);
    return guid_str;
  }
};

extern service_info_64_t g_boot_services_table_aarch64[];
extern size_t g_boot_services_table_aarch64_count;

extern service_info_64_t g_boot_services_table64[];
extern size_t g_boot_services_table64_count;

extern service_info_32_t g_boot_services_table32[];
extern size_t g_boot_services_table32_count;

extern service_t g_boot_services_table_all[];
extern size_t g_boot_services_table_all_count;

extern service_t g_runtime_services_table_all[];
```

```
extern size_t g_runtime_services_table_all_count;

extern service_info_64_t g_smm_services_prot64[];
extern size_t g_smm_services_prot64_count;

extern service_t g_smm_services_table_all[];
extern size_t g_smm_services_table_all_count;

extern service_info_32_t g_pei_services_table32[];
extern size_t g_pei_services_table32_count;

extern service_t g_pei_services_table_all[];
extern size_t g_pei_services_table_all_count;

extern service_t g_variable_ppi_table_all[];
extern size_t g_variable_ppi_table_all_count;

extern const char *g_plugin_name;
}
```

efiXplorer/efi_deps.h

```
/*
 * efiXplorer
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include "efi_utils.h"
#include <map>
#include <string>

class efi_deps_t {
public:
  efi_deps_t();
  ~efi_deps_t();

  json m_additional_installers;
  json m_modules_guids;
  json m_modules_info;
  json m_modules_sequence;
  json m_protocols_by_guids;
  json m_protocols_chooser;
  json m_uefitool_deps;
  string_set_t m_modules_from_idb;
```

```
  string_set_t m_untracked_protocols;

  // input: protocols from report
  void get_protocols_by_guids(json_list_t protocols);
  void get_protocols_chooser(json_list_t protocols);
  // get dependencies for specific protocol
  json get_deps_for(std::string protocol);
  // get installers by protocol GUIDs by searching
  // in the firmware and analysing xrefs
  void get_additional_installers();
  bool build_modules_sequence();
  bool get_modules_info();

private:
  string_set_t m_protocols_without_installers;

  bool installer_found(std::string protocol);
  bool load_deps_from_uefitool();
  bool load_modules_with_guids();
  json get_module_info(std::string module);
  std::string get_installer(std::string protocol);
  string_set_t get_apriori_modules();
  void get_installers_modules();
  void get_modules();
  void get_protocols_without_installers();
};
}
```

efiXplorer/efi_global.h

```
/*
 * efiXplorer
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include "efi_deps.h"

struct args_t {
  module_type_t module_type;
  int disable_ui;
  int disable_vuln_hunt;
};

extern args_t g_args;
```

```
extern efi_deps_t g_deps;
}
```

efiXplorer/efi_hexrays.h

```
/*
 * efiXplorer
 * Copyright (C) 2020-2024 Binarly, Rolf Rolles
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include "efi_utils.h"
#include <map>
#include <string>
#include <utility>
#include <vector>

namespace efi_hexrays {
bool apply_all_types_for_interfaces_smm(json_list_t guids);
bool apply_all_types_for_interfaces(json_list_t guids);
bool detect_pei_services(func_t *f);
bool is_pod_array(tinfo_t tif, unsigned int ptr_depth);
bool offset_of(tinfo_t tif, const char *name, unsigned int *offset);
bool set_hexrays_var_info_and_handle_interfaces(ea_t func_addr, lvar_t &ll,
                                                tinfo_t tif, std::string name);
bool set_hexrays_var_info(ea_t func_addr, lvar_t &ll, tinfo_t tif,
                          std::string name);
bool set_lvar_name(qstring name, lvar_t &lvar, ea_t func_addr);
bool track_entry_params(func_t *f, uint8_t depth);
const char *expr_to_string(cexpr_t *e, qstring *out);
json detect_vars(func_t *f);
json_list_t detect_pei_services_arm(func_t *f);
json_list_t detect_services(func_t *f);
uint8_t variables_info_extract_all(func_t *f, ea_t code_addr);
xreflist_t xrefs_to_stack_var(ea_t func_addr, lvar_t &ll, qstring name);

// description of a function pointer within a structure. Ultimately, this
// plugin is looking for calls to specific UEFI functions. This structure
// describes basic information about those functions:
struct target_funcptr_t {
  const char *name;      // name of function pointer in structure
  int offset;            // offset of function pointer (filled in later)
  unsigned int args;     // number of expected arguments
  unsigned int guid_arg; // which argument has the EFI_GUID *
  unsigned int out_arg;  // which argument retrieves the output
```

```
};

// this class holds all function pointer descriptors for one structure, as well
// as providing a utility to look up function pointers by offset
class service_descriptor_t {
  // instance data
protected:
  // the type of the containing structure (e.g. EFI_BOOT_SERVICES)
  tinfo_t m_type;

  // the name of the type (e.g. "EFI_BOOT_SERVICES")
  qstring m_name;

  // the ordinal of the type (e.g. 4)
  uint32 m_ordinal;

  // a vector of the structures above, copied, and with the offsets filled in
  std::vector<target_funcptr_t> m_targets;

  bool b_initialised;

  // ensure we can look up the type that this instance describes
  bool init_type(const char *name) {
    // import type
    import_type(get_idati(), -1, name);

    // get type by name
    if (!m_type.get_named_type(get_idati(), name))
      return false;

    // save ordinal and name
    m_ordinal = m_type.get_ordinal();
    m_name = name;
    return true;
  }

  // look up the offsets for all function pointer targets; save the results
  // in the vector; return false if offset lookup fails
  bool init_targets(target_funcptr_t *targets, size_t num) {
    // iterate through all targets
    for (int i = 0; i < num; ++i) {
      // copy the target structure into our local vector
      target_funcptr_t &tgt = m_targets.emplace_back();
      tgt = targets[i];

      // retrieve the offsets of each named function pointer
      unsigned int offset;
      if (!offset_of(m_type, targets[i].name, &offset)) {
        return false;
      }
    }
    return true;
  }

public:
  // constructor does nothing
  service_descriptor_t() : m_ordinal(0), b_initialised(false) {}

  // accessor for ordinal
  uint32 get_ordinal() { return m_ordinal; }

  // accessor for name
  const char *get_name() { return m_name.c_str(); }
```

```
  // needs to be called before the object can be used
  bool initialise(const char *name, target_funcptr_t *targets, size_t num) {
    if (b_initialised)
      return true;
    b_initialised = init_type(name) && init_targets(targets, num);
    return b_initialised;
  }

  // after initialisation, look up a target by offset
  bool lookup_offset(unsigned int offset, target_funcptr_t **tgt) {
    // iterating through a vector generally is inefficient compared
    // to a map, but there are at most 3 function pointers so far, so it
    // outweighs the overhead of the associative containers.
    for (auto &it : m_targets) {
      // Match by offset
      if (it.offset == offset) {
        *tgt = &it;
        return true;
      }
    }
    // if we don't find it, it's not necessarily "bad" from the
    // point of view of the plugin's logic. After all, we're looking at every
    // access to the selected structures, and so, quite rightly, we'll want to
    // ignore the function pointers that we're not tracking.
    return false;
  }
};

// this class manages multiple instances of the class above. Each
// such structure is associated with the ordinal of its containing structure
// type. Then, when the Hex-Rays visitor needs to look up a function pointer
// access into a structure, it just passes the structure ordinal and offset.
// This class looks up the service_descriptor_t in a map by ordinal, and then
// looks up the offset if that succeeded.
class service_descriptor_map_t {
protected:
  // our map for looking up service_descriptor_t structures. I
  // should probably change the value type to a pointer.
  std::map<uint32, service_descriptor_t> m_services;

public:
  // add a new service_descriptor_t to the map. I should change the
  // argument type to match whatever I change the value type of the map to.
  bool register_sd(service_descriptor_t sd) {
    // get the ordinal from the service_descriptor_t
    uint32 ord = sd.get_ordinal();

    // are we already tracking this structure?
    if (m_services.find(ord) != m_services.end()) {
      return false;
    }
    // if not, register it. Get rid of std::move
    m_services[ord] = std::move(sd);
    return true;
  }

  // this function could be protected, but whatever. Given an ordinal, get
  // the tracked service_descriptor_t, if applicable
  bool lookup_ordinal(uint32 ord, service_descriptor_t **sd) {
    auto it = m_services.find(ord);
    if (it == m_services.end()) {
      return false;
    }
    *sd = &it->second;
```

```
    return true;
  }

  // this is the high-level function that clients call. Given a structure
  // ordinal and offset of a function pointer, see if it's something we're
  // tracking. If so, get pointers to the tracked objects and return true.
  bool lookup_offset(uint32 ord, unsigned int offset, service_descriptor_t **sd,
                     target_funcptr_t **tgt) {
    if (!lookup_ordinal(ord, sd))
      return false;
    if (!(*sd)->lookup_offset(offset, tgt))
      return false;
    return true;
  }
};

// base class for two visitors that require similar functionality. Here we
// collect all of the common data and functionality that will be used by both
// of those visitors. This allows the derivatives to be very succinct.
class guid_related_visitor_base_t : public ctree_visitor_t {
public:
  // we need access to a service_descriptor_map_t from above
  explicit guid_related_visitor_base_t(service_descriptor_map_t &m)
      : ctree_visitor_t(CV_FAST), m_debug(true), m_services(m) {}

  // we need the function ea when setting Hex-Rays variable types
  void set_func_ea(ea_t ea) { m_func_ea = ea; }
  void set_code_ea(ea_t ea) { m_code_ea = ea; }
  void set_protocols(json_list_t protocols) { m_protocols = protocols; }

protected:
  ea_t m_func_ea;
  ea_t m_code_ea;
  json_list_t m_protocols;
  bool m_debug = true;

  // used for looking up calls to function pointers in structures
  service_descriptor_map_t &m_services;

  //
  // state variables, cleared on every iteration. I debated with myself
  // whether this was a nasty design decision. I think it's fine. These
  // variables are only valid to access after the client has called
  // validate_call_and_guid, and it returned true. If you called that and it
  // returned false, these will be in an inconsistent state. Don't touch them
  // if that's the case.
  //

  // address of the indirect function call
  ea_t m_ea;

  // the pointer type that's being accessed (that of the structure)
  tinfo_t m_tif;

  // the structure type, with the pointer indirection removed
  tinfo_t m_tif_noptr;

  // the service_descriptor_t for the containing structure
  service_descriptor_t *m_service;

  // the ordinal of the structure type
  uint32 m_ordinal;

  // the offset of the function pointer in the structure
```

```
unsigned int m_offset;

// details about the target of the indirect call (e.g. name)
target_funcptr_t *m_target;

// the list of arguments for the indirect call
carglist_t *m_args;

// the argument that specifies the GUID for the indirect call
cexpr_t *m_guid_arg;

// the argument that gets the output for the indirect call
cexpr_t *m_out_arg;

// the GUID argument will be &x; this is x
cexpr_t *m_guid_arg_ref_to;

// the address of the GUID being passed to the indirect call
ea_t m_guid_ea;

void clear() {
  m_ea = BADADDR;
  m_tif.clear();
  m_tif_noptr.clear();
  m_service = nullptr;
  m_ordinal = 0;
  m_offset = -1;
  m_target = nullptr;
  m_args = nullptr;
  m_guid_arg = nullptr;
  m_out_arg = nullptr;
  m_guid_arg_ref_to = nullptr;
  m_guid_ea = BADADDR;
}

// this is the first function called every time the visitor visits an
// expression. This function determines if the expression is a call to a
// function pointer contained in a structure
bool get_call_ord_and_offset(cexpr_t *e) {
  // set instance variable for call address
  m_ea = e->ea;

  if (m_ea != m_code_ea) {
    return false;
  }

  // if it's not a call, we're done
  if (e->op != cot_call)
    return false;

  // set instance variable with call arguments
  m_args = e->a;

  // if it's a direct call, we're done
  cexpr_t *call_dest = e->x;
  if (call_dest->op == cot_obj)
    return false;

  // eat any casts on the type of what's being called
  while (call_dest->op == cot_cast)
    call_dest = call_dest->x;

  // if the destination is not a member of a structure, we're done
  if (call_dest->op != cot_memptr)
```

```
    return false;

  // set instance variable with type of structure containing pointer
  m_tif = call_dest->x->type;

  // ensure that the structure is being accessed via pointer, and not as a
  // reference (i.e., through a structure held on the stack as a local
  // variable)
  if (!m_tif.is_ptr()) {
    return false;
  }

  // remove pointer from containing structure type, set instance variable
  m_tif_noptr = remove_pointer(m_tif);

  // get the ordinal of the structure
  m_ordinal = m_tif_noptr.get_ordinal();

  // if we can't get a type for the structure, that's bad
  if (m_ordinal == 0)
    return false;

  // get the offset of the function pointer in the structure
  m_offset = call_dest->m;

  // now we know we're dealing with an indirect call to a function
  // pointer contained in a structure, where the structure is being
  // accessed by a pointer
  return true;
}

// this is the second function called as part of indirect call validation.
// Now we want to know: is it a call to something that we're tracking?
bool validate_call_destination() {
  // look up the structure ordinal and function offset; get the associated
  // service_descriptor_t and target_funcptr_t (instance variables)
  if (!m_services.lookup_offset(m_ordinal, m_offset, &m_service, &m_target))
    return false;

  // it was something that we were tracking. Now, sanity-check the
  // number of arguments on the function call. (Hex-Rays might have gotten
  // this wrong. The user can fix it via "set call type")
  size_t args_size = m_args->size();
  size_t args = m_target->args;
  if (args_size != args) {
    return false;
  }

  // the target_funcptr_t tells us which argument takes an EFI_GUID *,
  // and which one retrieves the output. Get those arguments, and save them
  // as instance variables
  m_guid_arg = &m_args->at(m_target->guid_arg);
  m_out_arg = &m_args->at(m_target->out_arg);

  // now we know that the expression is an indirect call to
  // something that we're tracking, and that Hex-Rays decompiled the call
  // the way we expected it to
  return true;
}

// this is a helper function used to get the thing being referred to. What
// does that m_ean?
//
// - for GUID arguments, we'll usually have &globvar. Return globvar
```

```
// - for output arguments, we'll usually have &globvar or &locvar. Due to
//   Hex-Rays internal heuristics, we might end up with "locarray", which
//   does not actually have a "&" when passed as a call argument. There's
//   a bit of extra logic to check for that case
cexpr_t *get_referent(cexpr_t *e, const char *desc, bool b_accept_var) {
  // Eat casts
  cexpr_t *x = e;
  while (x->op == cot_cast)
    x = x->x;

  qstring estr;
  // if we're accepting local variables, and this is a variable (note: not
  // a *reference* to a variable)
  if (b_accept_var && x->op == cot_var) {
    // get the variable details
    var_ref_t var_ref = x->v;
    lvar_t dest_var = var_ref.mba->vars[var_ref.idx];

    // ensure that it's an array of POD types, or pointers to them
    bool bis_pod_array = is_pod_array(dest_var.tif, 1);

    // if it is a POD array, good, we'll take it
    return bis_pod_array ? x : nullptr;
  }

  // for everything else, we really want it to be a reference: either to a
  // global or local variable. If it's not a reference, we can't get the
  // referent, so fail
  if (x->op != cot_ref) {
    return nullptr;
  }

  // if we get here, we know it's a reference. Return the referent.
  return x->x;
}

// the third function in the validation logic. We already know the
// expression is an indirect call to something that we're tracking, and
// that Hex-Rays' decompilation matches on the number of arguments. Now,
// we validate that the GUID argument does in fact point to a global
// variable
bool validate_guid_arg() {
  // does the GUID argument point to a local variable?
  m_guid_arg_ref_to = get_referent(m_guid_arg, "GUID", false);
  if (!m_guid_arg_ref_to)
    return false;

  // if we get here, we know it was a reference to *something*. Ensure that
  // something is a global variable
  if (m_guid_arg_ref_to->op != cot_obj) {
    return false;
  }

  // save the address of the global variable to which the GUID argument is
  // pointing
  m_guid_ea = m_guid_arg_ref_to->obj_ea;

  // now we know we're dealing with an indirect call to something
  // we're tracking; that Hex-Rays decompiled the call with the proper
  // number of arguments; and that the GUID argument did in fact point to
  // a global variable, whose address we now have in an instance variable.
  return true;
}
```

```
  // finally, this function combines all three checks above into one single
  // function. If you call this and it returns true, feel free to access the
  // instance variables, as they are guaranteed to be valid. If it returns
  // false, they aren't, so don't touch them
  bool validate_call_and_guid(cexpr_t *e) {
    // Reset all instance variables. Not strictly necessary; call it
    // "defensive programming".
    clear();

    // validate according to the logic above
    return (get_call_ord_and_offset(e) && validate_call_destination() &&
            validate_guid_arg());
  }
};

// now that we've implemented all that validation logic, this class is pretty
// simple. This one is responsible for ensuring that the GUID is something that
// we know about, and setting the types of the output variables accordingly
class guid_retyper_t : public guid_related_visitor_base_t {
public:
  explicit guid_retyper_t(service_descriptor_map_t &m)
      : guid_related_visitor_base_t(m), m_num_applied(0) {}

  // this is the callback function that Hex-Rays invokes for every expression
  // in the CTREE
  int visit_expr(cexpr_t *e) {
    // perform the checks from guid_related_visitor_base_t. If they fail, we're
    // not equipped to deal with this expression, so bail out
    if (!validate_call_and_guid(e))
      return 0;

    m_guid_arg_ref_to = get_referent(m_guid_arg, "GUID", false);
    if (m_guid_arg_ref_to == nullptr)
      return 0;
    ea_t guidAddr = m_guid_arg_ref_to->obj_ea;

    // get interface type name
    std::string guid_name;
    for (auto g : m_protocols) {
      if (guidAddr == g["address"]) {
        guid_name = g["prot_name"];
        break;
      }
    }
    if (guid_name.empty()) {
      return 0;
    }

    std::string interface_type_name =
        guid_name.substr(0, guid_name.find("_GUID"));
    if (!interface_type_name.find("FCH_")) {
      // convert FCH_SMM_* dispatcher type to EFI_SMM_* dispatcher type
      interface_type_name.replace(0, 4, "EFI_");
    }

    // need to get the type for the interface variable here
    tinfo_t tif;
    import_type(get_idati(), -1, interface_type_name.c_str());
    if (!tif.get_named_type(get_idati(), interface_type_name.c_str())) {
      // get the referent for the interface argument
      cexpr_t *out_arg_referent = get_referent(m_out_arg, "ptr", true);
      if (out_arg_referent == nullptr)
        return 0;
      apply_name(out_arg_referent, interface_type_name);
```

```
      return 0;
    }

    qstring tstr;
    if (!tif.get_type_name(&tstr)) {
      return 0;
    }

    tinfo_t tif_guid_ptr;
    if (!tif_guid_ptr.create_ptr(tif)) {
      return 0;
    }

    // get the referent for the interface argument
    cexpr_t *out_arg_referent = get_referent(m_out_arg, "ptr", true);
    if (out_arg_referent == nullptr)
      return 0;

    // apply the type to the output referent
    apply_type(out_arg_referent, tif_guid_ptr, tstr);
    return 1;
  }

protected:
  unsigned int m_num_applied;

  // given an expression (either a local or global variable) and a type to
  // apply, apply the type. This is just a bit of IDA/Hex-Rays type system
  // skullduggery
  void apply_type(cexpr_t *out_arg, tinfo_t ptr_tif, qstring tstr) {
    ea_t dest_ea = out_arg->obj_ea;

    // for global variables
    if (out_arg->op == cot_obj) {
      // just apply the type information to the address
      apply_tinfo(dest_ea, ptr_tif, TINFO_DEFINITE);
      ++m_num_applied;

      // rename global variable
      auto name = "g" + efi_utils::type_to_name(tstr.c_str());
      set_name(dest_ea, name.c_str(), SN_FORCE);

      // get xrefs to global variable
      auto xrefs = efi_utils::get_xrefs(dest_ea);
      qstring type_name;
      ptr_type_data_t pi;
      ptr_tif.get_ptr_details(&pi);
      pi.obj_type.get_type_name(&type_name);

      // handling all interface functions (to rename function arguments)
      efi_utils::op_stroff_for_global_interface(xrefs, type_name);
    } else if (out_arg->op == cot_var) { // for local variables
      var_ref_t var_ref = out_arg->v;
      lvar_t &dest_var = var_ref.mba->vars[var_ref.idx];

      // set the Hex-Rays variable type
      auto name = efi_utils::type_to_name(tstr.c_str());
      set_lvar_name(name.c_str(), dest_var, m_func_ea);
      if (set_hexrays_var_info_and_handle_interfaces(m_func_ea, dest_var,
                                                     ptr_tif, name)) {
        ++m_num_applied;
      }
    }
  }
```

```
  void apply_name(cexpr_t *out_arg, std::string type_name) {
    ea_t dest_ea = out_arg->obj_ea;

    // for global variables
    if (out_arg->op == cot_obj) {
      // rename global variable
      auto name = "g" + efi_utils::type_to_name(type_name);
      set_name(dest_ea, name.c_str(), SN_FORCE);
    } else if (out_arg->op == cot_var) { // for local variables
      var_ref_t var_ref = out_arg->v;
      lvar_t &dest_var = var_ref.mba->vars[var_ref.idx];
      // set the Hex-Rays variable type
      auto name = efi_utils::type_to_name(type_name);
      set_lvar_name(name.c_str(), dest_var, m_func_ea);
    }
  }
};

class variables_info_extractor_t : public ctree_visitor_t {
public:
  explicit variables_info_extractor_t(ea_t code_addr)
      : ctree_visitor_t(CV_FAST) {
    m_code_addr = code_addr;
  }

  uint8_t m_attributes = 0xff;

  // this is the callback function that Hex-Rays invokes for every expression
  // in the CTREE
  int visit_expr(cexpr_t *e) {
    if (m_code_addr == BADADDR) {
      return 0;
    }

    if (e->ea != m_code_addr) {
      return 0;
    }

    if (e->op != cot_call)
      return 0;

    carglist_t *args = e->a;
    if (args == nullptr) {
      return 0;
    }

    size_t args_size = args->size();
    if (args_size < 3) {
      return 0;
    }

    cexpr_t *attributes_arg = &args->at(2);
    if (attributes_arg->op == cot_num) {
      attributes_arg->numval();
      m_attributes = static_cast<uint8_t>(attributes_arg->numval());
    }

    return 0;
  }

protected:
  ea_t m_code_addr = BADADDR;
  bool m_debug = true;
```

```
};

class prototypes_fixer_t : public ctree_visitor_t {
public:
  prototypes_fixer_t() : ctree_visitor_t(CV_FAST) {}
  ea_list_t m_child_functions;

  // this is the callback function that Hex-Rays invokes for every expression
  // in the CTREE
  int visit_expr(cexpr_t *e) {
    if (e->op != cot_call)
      return 0;

    // get child function address
    if (e->x->op != cot_obj) {
      return 0;
    }
    if (m_debug) {
      efi_utils::log("child function address: 0x%" PRIx64 "\n",
                     u64_addr(e->x->obj_ea));
    }

    carglist_t *args = e->a;
    if (args == nullptr) {
      return 0;
    }

    // get child function prototype
    ea_t func_addr = e->x->obj_ea;
    hexrays_failure_t hf;
    func_t *f = get_func(func_addr);
    if (f == nullptr) {
      return 0;
    }

    cfuncptr_t cf = decompile(f, &hf, DECOMP_NO_WAIT);
    if (cf == nullptr) {
      return 0;
    }

    if (m_debug) {
      efi_utils::log("call address: 0x%" PRIx64 "\n", u64_addr(e->ea));
    }
    for (auto i = 0; i < args->size(); i++) {
      cexpr_t *arg = &args->at(i);
      if (arg->op == cot_cast || arg->op == cot_var) {
        // extract argument type
        tinfo_t arg_type;
        tinfo_t arg_type_no_ptr;
        if (arg->op == cot_var) {
          arg_type = arg->type;
        }
        if (arg->op == cot_cast) {
          arg_type = arg->x->type;
        }

        // print type
        if (arg_type.is_ptr()) {
          arg_type_no_ptr = remove_pointer(arg_type);
        }

        qstring type_name;
        bool is_ptr = false;
        if (!arg_type.get_type_name(&type_name)) {
```

```
          if (!arg_type_no_ptr.get_type_name(&type_name)) {
            continue;
          }
          is_ptr = true;
        }

        if (is_ptr) {
          efi_utils::log("arg #%d, type = %s *\n", i, type_name.c_str());
        } else {
          efi_utils::log("arg #%d, type = %s\n", i, type_name.c_str());
        }

        if (type_name == qstring("EFI_HANDLE") ||
            type_name == qstring("EFI_SYSTEM_TABLE")) {
          if (!efi_utils::addr_in_vec(m_child_functions, func_addr)) {
            m_child_functions.push_back(func_addr);
          }

          // set argument type and name
          if (cf->argidx.size() <= i) {
            return 0;
          }

          auto argid = cf->argidx[i];
          lvar_t &arg_var = cf->mba->vars[argid]; // get lvar for argument
          if (type_name == qstring("EFI_HANDLE")) {
            set_hexrays_var_info(func_addr, arg_var, arg_type, "ImageHandle");
          }
          if (type_name == qstring("EFI_SYSTEM_TABLE")) {
            set_hexrays_var_info(func_addr, arg_var, arg_type, "SystemTable");
          }
        }
      }
    }

    return 0;
  }

protected:
  bool m_debug = true;
};

class variables_detector_t : public ctree_visitor_t {
public:
  variables_detector_t() : ctree_visitor_t(CV_FAST) {}

  ea_list_t m_child_functions;

  ea_list_t m_image_handle_list;
  ea_list_t m_st_list;
  ea_list_t m_bs_list;
  ea_list_t m_rt_list;

  void set_func_ea(ea_t ea) { m_func_ea = ea; }

  // this is the callback function that Hex-Rays invokes for every expression
  // in the CTREE
  int visit_expr(cexpr_t *e) {
    if (e->op == cot_asg) {
      // saving a child function for recursive analysis
      if (!efi_utils::addr_in_vec(m_child_functions, e->ea)) {
        m_child_functions.push_back(e->x->obj_ea);
      }
    }
```

```
bool global_var = false;
bool local_var = false;
if (e->op != cot_asg) {
  return 0;
}

switch (e->x->op) {
case cot_obj:
  // asg operation for global variable
  global_var = true;
  break;
case cot_var:
  // asg operation for local variable
  local_var = true;
  break;
default:
  return 0;
}

// extract variable type
tinfo_t var_type;
tinfo_t var_type_no_ptr;
if (e->y->op == cot_memptr && e->y->x->op == cot_var) {
  var_type = e->y->type;
} else if (e->y->op == cot_var) {
  var_type = e->y->type;
} else if (e->y->op == cot_cast) {
  var_type = e->y->x->type;
} else {
  return 0;
}

if (var_type.is_ptr()) {
  var_type_no_ptr = remove_pointer(var_type);
}

qstring type_name;
bool is_ptr = false;
if (!var_type.get_type_name(&type_name)) {
  if (!var_type_no_ptr.get_type_name(&type_name)) {
    return 0;
  }
  is_ptr = true;
}

if (m_debug) {
  efi_utils::log("code address: 0x%" PRIx64 ", type name: %s\n",
                 u64_addr(e->ea), type_name.c_str());
}

if (global_var) {
  // extract variable data
  ea_t g_addr = e->x->obj_ea;
  std::string type_name_str = type_name.c_str();
  if (type_name == "EFI_HANDLE") {
    efi_utils::set_type_and_name(g_addr, "gImageHandle", type_name_str);
    if (!efi_utils::addr_in_vec(m_image_handle_list, g_addr)) {
      m_image_handle_list.push_back(g_addr);
    }
  }
  if (type_name == "EFI_SYSTEM_TABLE") {
    efi_utils::set_ptr_type_and_name(g_addr, "gST", type_name_str);
    if (!efi_utils::addr_in_vec(m_st_list, g_addr)) {
```

```
          m_st_list.push_back(g_addr);
        }
      }
      if (type_name == "EFI_BOOT_SERVICES") {
        efi_utils::set_ptr_type_and_name(g_addr, "gBS", type_name_str);
        if (!efi_utils::addr_in_vec(m_bs_list, g_addr)) {
          m_bs_list.push_back(g_addr);
        }
      }
      if (type_name == "EFI_RUNTIME_SERVICES") {
        efi_utils::set_ptr_type_and_name(g_addr, "gRT", type_name_str);
        if (!efi_utils::addr_in_vec(m_rt_list, g_addr)) {
          m_rt_list.push_back(g_addr);
        }
      }
    }

    if (local_var) {
      // set the Hex-Rays variable type
      auto name = efi_utils::type_to_name(type_name.c_str());
      efi_utils::log("found %s at 0x%" PRIx64 " (function: 0x%" PRIx64 ")\n",
                     name.c_str(), u64_addr(e->ea), u64_addr(m_func_ea));
    }

    return 0;
  }

protected:
  bool m_debug = true;
  ea_t m_func_ea = BADADDR;
};

class services_detector_t : public ctree_visitor_t {
  // detect all services (Boot services, Runtime services, etc)
public:
  services_detector_t() : ctree_visitor_t(CV_FAST) {}

  json_list_t m_services;

  // this is the callback function that Hex-Rays invokes for every expression
  // in the CTREE
  int visit_expr(cexpr_t *e) {
    if (e->op != cot_call) {
      return 0;
    }

    if (e->x->op != cot_cast) {
      return 0;
    }

    // extract function type
    auto e_func = e->x->x;
    tinfo_t func_type;
    tinfo_t func_type_no_ptr;

    func_type = e_func->type;

    if (func_type.is_ptr()) {
      func_type_no_ptr = remove_pointer(func_type);
    }

    qstring type_name;
    bool is_ptr = false;
    if (!func_type.get_type_name(&type_name)) {
```

```
      if (!func_type_no_ptr.get_type_name(&type_name)) {
        return 0;
      }
      is_ptr = 0;
    }

    auto service_name = efi_utils::type_to_name(type_name.c_str());
    if (service_name.rfind("Efi", 0) == 0) {
      service_name = service_name.substr(3);
      if (service_name == "RaiseTpl") {
        service_name = "RaiseTPL";
      }
      if (service_name == "RestoreTpl") {
        service_name = "RestoreTPL";
      }
    }
    if (m_debug) {
      efi_utils::log("address: 0x%" PRIx64
                     ", service type: %s, service name: %s\n",
                     u64_addr(e->ea), type_name.c_str(), service_name.c_str());
    }

    json s;
    s["address"] = e->ea;
    s["service_name"] = service_name;
    s["table_name"] = efi_utils::get_table_name(service_name);

    if (!efi_utils::json_in_vec(m_services, s)) {
      m_services.push_back(s);
    }

    return 0;
  }

protected:
  bool m_debug = true;
};

class pei_services_detector_t : public ctree_visitor_t {
  // detect and mark all PEI services
public:
  pei_services_detector_t() : ctree_visitor_t(CV_FAST) {}

  bool make_shifted_ptr(tinfo_t outer, tinfo_t inner, int32 offset,
                        tinfo_t *shifted_tif) {
    ptr_type_data_t pi;
    pi.taptr_bits = TAPTR_SHIFTED;
    pi.delta = offset;
    pi.parent = outer;
    pi.obj_type = inner;
    shifted_tif->create_ptr(pi);
    return shifted_tif->is_correct();
  }

  bool set_var_type(ea_t func_ea, lvar_t lvar, tinfo_t tif) {
    lvar_saved_info_t lsi;
    lsi.ll = lvar;
    lsi.type = tif;
    return modify_user_lvar_info(func_ea, MLI_TYPE, lsi);
  }

  // this is the callback function that Hex-Rays invokes for every expression
  // in the CTREE
  int visit_expr(cexpr_t *e) {
```

```
    auto pointer_offset = BADADDR;
    auto service_offset = BADADDR;
    bool call = false;
    var_ref_t var_ref;
    if (e->op == cot_ptr && e->x->op == cot_cast && e->x->x->op == cot_add &&
        e->x->x->x->op == cot_ptr && e->x->x->x->x->op == cot_ptr &&
        e->x->x->x->x->x->op == cot_cast &&
        e->x->x->x->x->x->x->op == cot_sub &&
        e->x->x->x->x->x->x->x->op == cot_var &&
        e->x->x->x->x->x->x->x->y->op == cot_num && e->x->x->y->op == cot_num) {
      // (*ADJ(v2)->PeiServices)->GetHobList(
      // (const EFI_PEI_SERVICES**)ADJ(v2)->PeiServices, HobList);
      service_offset = e->x->x->y->numval();
      pointer_offset = e->x->x->x->x->x->x->x->y->numval();
      var_ref = e->x->x->x->x->x->x->x->v;
      call = true;
    } else if (e->op == cot_asg && e->x->op == cot_var && e->y->op == cot_ptr &&
               e->y->x->op == cot_cast && e->y->x->x->op == cot_sub &&
               e->y->x->x->x->op == cot_var && e->y->x->x->y->op == cot_num) {
      // __sidt(v6);
      // PeiServices = ADJ(v7)->PeiServices;
      pointer_offset = e->y->x->x->y->numval();
      var_ref = e->y->x->x->x->v;
    } else {
      return 0;
    }

    if (pointer_offset != 4) {
      return 0;
    }

    efi_utils::log("PEI service detected at 0x%" PRIx64 "\n", u64_addr(e->ea));

    tinfo_t outer;
    if (!outer.get_named_type(get_idati(), "EFI_PEI_SERVICES_4", BTF_STRUCT)) {
      return 0;
    }

    tinfo_t shifted_tif;
    if (!make_shifted_ptr(outer, outer, pointer_offset, &shifted_tif)) {
      return 0;
    }

    lvar_t &dest_var = var_ref.mba->vars[var_ref.idx];
    func_t *func = get_func(e->ea);
    if (func == nullptr) {
      return 0;
    }
    if (set_var_type(func->start_ea, dest_var, shifted_tif)) {
      efi_utils::log("shifted pointer applied at 0x%" PRIx64 "\n",
                     u64_addr(e->ea));
    }

    if (call) {
      efi_utils::op_stroff(e->ea, "EFI_PEI_SERVICES");
    }

    return 0;
  }

protected:
  bool m_debug = true;
};
```

```
class pei_services_detector_arm_t : public ctree_visitor_t {
  // detect and mark all PEI services for ARM firmware
  // tested on Ampere firmware that contains small PEI stack
public:
  pei_services_detector_arm_t() : ctree_visitor_t(CV_FAST) {}

  json_list_t m_services;

  // this is the callback function that Hex-Rays invokes for every expression
  // in the CTREE
  int visit_expr(cexpr_t *e) {
    if (!(e->op == cot_call && e->x->op == cot_memptr &&
          e->x->x->op == cot_ptr && e->x->x->x->op == cot_var)) {
      return 0;
    }
    ea_t offset = e->x->m;

    // check if service from EFI_PEI_SERVICES
    tinfo_t table_type = e->x->x->type;
    tinfo_t table_type_no_ptr;
    qstring table_type_name;
    if (table_type.is_ptr()) {
      table_type_no_ptr = remove_pointer(table_type);
      table_type_no_ptr.get_type_name(&table_type_name);
    } else {
      table_type.get_type_name(&table_type_name);
    }

    // get service name from function type
    std::string service_name;
    if (table_type_name != "EFI_PEI_SERVICES") {
      qstring func_type_name;
      tinfo_t service_type = e->x->type;
      service_type.get_type_name(&func_type_name);
      std::string func_type = func_type_name.c_str();
      std::string prefix = "EFI_PEI_";
      if (func_type.substr(0, prefix.length()) == prefix) {
        func_type.erase(0, prefix.length());
      }
      service_name = efi_utils::type_to_name(func_type);
    } else {
      auto s = m_pei_services.find(offset);
      if (s == m_pei_services.end()) {
        return 0;
      }
      service_name = s->second;
    }
    if (m_debug) {
      efi_utils::log("0x%" PRIx64 ": %s service detected (offset: %d): %s\n",
                     u64_addr(e->ea), table_type_name.c_str(), u32_addr(offset),
                     service_name.c_str());
    }

    json s;
    s["address"] = e->ea;
    s["service_name"] = service_name;
    s["table_name"] = table_type_name.c_str();

    if (!efi_utils::json_in_vec(m_services, s)) {
      m_services.push_back(s);
    }

    return 0;
  }
```

```
protected:
  bool m_debug = true;
  std::map<ea_t, std::string> m_pei_services = {
      {0x18, "InstallPpi"},
      {0x20, "ReInstallPpi"},
      {0x28, "LocatePpi"},
      {0x30, "NotifyPpi"},
      {0x38, "GetBootMode"},
      {0x40, "SetBootMode"},
      {0x48, "GetHobList"},
      {0x50, "CreateHob"},
      {0x58, "FfsFindNextVolume"},
      {0x60, "FfsFindNextFile"},
      {0x68, "FfsFindSectionData"},
      {0x70, "InstallPeiMemory"},
      {0x78, "AllocatePages"},
      {0x80, "AllocatePool"},
      {0x88, "CopyMem"},
      {0x90, "SetMem"},
      {0x98, "ReportStatusCode"},
      {0xA0, "ResetSystem"},
      {0xA8, "CpuIo"},
      {0xB0, "PciCfg"},
      {0xB8, "FfsFindFileByName"},
      {0xC0, "FfsGetFileInfo"},
      {0xC8, "FfsGetVolumeInfo"},
      {0xD0, "RegisterForShadow"},
      {0xD8, "FindSectionData4"},
      {0xE0, "FfsGetFileInfo3"},
      {0xE8, "ResetSystem3"},
  };
};
} // namespace efi_hexrays
}
```

efiXplorer/efi_smm_utils.h

```
/*
 * efiXplorer
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include "efi_utils.h"
```

```
#include <string>

namespace efi_smm_utils {
ea_list_t find_smst_sw_dispatch(ea_list_t bs_list);
ea_list_t find_smst_smm_base(ea_list_t bs_list);
func_list_t find_smi_handlers(ea_t address, std::string prefix);
func_list_t find_smi_handlers_dispatch(efi_guid_t guid, std::string prefix);
func_list_t find_smi_handlers_dispatch_stack(json_list_t stack_guids,
                                             std::string prefix);
ea_list_t find_smm_get_variable_calls(segment_list_t data_segs,
                                      json_list_t *all_services);
ea_list_t resolve_efi_smm_cpu_protocol(json_list_t stack_guids,
                                       json_list_t data_guids,
                                       json_list_t *all_services);
ea_t mark_child_sw_smi_handlers(ea_t ea);
} // namespace efi_smm_utils
}
```

efiXplorer/efi_ui.h

```
/*
 * efiXplorer
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include "efi_utils.h"
#include <string>

//---------------------------------------------------------------------------
// vulns chooser
class vulns_chooser_t : public chooser_t {
public:
  eavec_t list;
  json chooser_vulns;

  // this object must be allocated using `new`
  vulns_chooser_t(const char *title, bool ok, json_list_t vulns);

  // function that is used to decide whether a new chooser should be opened or
  // we can use the existing one. The contents of the window are completely
  // determined by its title
  virtual const void *get_obj_id(size_t *len) const {
    *len = strlen(title);
    return title;
```

```
  }

  // function that returns number of lines in the list
  virtual size_t idaapi get_count() const { return list.size(); }

  // function that generates the list line
  virtual void idaapi get_row(qstrvec_t *cols, int *icon_,
                              chooser_item_attrs_t *attrs, size_t n) const;

  // function that is called when the user hits `Enter`
  virtual cbret_t idaapi enter(size_t n) {
    if (n < list.size())
      jumpto(list[n]);
    return cbret_t();
  }

protected:
  static const int widths_vulns[];
  static const char *const header_vulns[];

  void build_list(bool ok, json_list_t vulns) {
    size_t n = 0;
    for (auto vuln : vulns) {
      list.push_back(vuln["address"]);
      chooser_vulns[n] = vuln;
      n++;
    }
    ok = true;
  }
};

//-------------------------------------------------------------------------
// GUIDs chooser
class guids_chooser_t : public chooser_t {
protected:
  static const int widths_guids[];
  static const char *const header_guids[];

public:
  eavec_t list;
  json chooser_guids;

  // this object must be allocated using `new`
  guids_chooser_t(const char *title, bool ok, json_list_t guids);

  // function that is used to decide whether a new chooser should be opened or
  // we can use the existing one. The contents of the window are completely
  // determined by its title
  virtual const void *get_obj_id(size_t *len) const {
    *len = strlen(title);
    return title;
  }

  // function that returns number of lines in the list
  virtual size_t idaapi get_count() const { return list.size(); }

  // function that generates the list line
  virtual void idaapi get_row(qstrvec_t *cols, int *icon_,
                              chooser_item_attrs_t *attrs, size_t n) const;

  // function that is called when the user hits `Enter`
  virtual cbret_t idaapi enter(size_t n) {
    if (n < list.size())
      jumpto(list[n]);
```

```
    return cbret_t();
  }

protected:
  void build_list(bool ok, json_list_t guids) {
    size_t n = 0;
    for (auto guid : guids) {
      list.push_back(guid["address"]);
      chooser_guids[n] = guid;
      n++;
    }
    ok = true;
  }
};

//------------------------------------------------------------------------
// protocols chooser
class m_protocols_chooser_t : public chooser_t {
protected:
  static const int widths_protocols[];
  static const char *const header_protocols[];

public:
  eavec_t list;
  json chooser_protocols;
  std::string name_key;

  // this object must be allocated using `new`
  m_protocols_chooser_t(const char *title, bool ok, json_list_t interfaces,
                        std::string name_key);

  // function that is used to decide whether a new chooser should be opened or
  // we can use the existing one. The contents of the window are completely
  // determined by its title
  virtual const void *get_obj_id(size_t *len) const {
    *len = strlen(title);
    return title;
  }

  // function that returns number of lines in the list
  virtual size_t idaapi get_count() const { return list.size(); }

  // function that generates the list line
  virtual void idaapi get_row(qstrvec_t *cols, int *icon_,
                              chooser_item_attrs_t *attrs, size_t n) const;

  // function that is called when the user hits `Enter`
  virtual cbret_t idaapi enter(size_t n) {
    if (n < list.size())
      jumpto(list[n]);
    return cbret_t();
  }

protected:
  void build_list(bool ok, json_list_t protocols) {
    size_t n = 0;
    for (auto protocol : protocols) {
      list.push_back(protocol["xref"]);
      chooser_protocols[n] = protocol;
      n++;
    }
    ok = true;
  }
};
```

```
//------------------------------------------------------------------------
// service chooser (address : service_name)
class services_chooser_t : public chooser_t {
protected:
  static const int widths_s[];
  static const char *const header_s[];

public:
  eavec_t list;
  json chooser_s;

  // this object must be allocated using `new`
  services_chooser_t(const char *title, bool ok, json_list_t services);

  // function that is used to decide whether a new chooser should be opened or
  // we can use the existing one. The contents of the window are completely
  // determined by its title
  virtual const void *get_obj_id(size_t *len) const {
    *len = strlen(title);
    return title;
  }

  // function that returns number of lines in the list
  virtual size_t idaapi get_count() const { return list.size(); }

  // function that generates the list line
  virtual void idaapi get_row(qstrvec_t *cols, int *icon_,
                              chooser_item_attrs_t *attrs, size_t n) const;

  // function that is called when the user hits `Enter`
  virtual cbret_t idaapi enter(size_t n) {
    if (n < list.size())
      jumpto(list[n]);
    return cbret_t();
  }

protected:
  void build_list(bool ok, json_list_t services) {
    size_t n = 0;
    for (auto j_service : services) {
      list.push_back(j_service["address"]);
      chooser_s[n] = j_service;
      n++;
    }
    ok = true;
  }
};

//------------------------------------------------------------------------
// NVRAM chooser
class nvram_chooser_t : public chooser_t {
protected:
  static const int widths_nvram[];
  static const char *const header_nvram[];

public:
  eavec_t list;
  json chooser_nvram;

  // this object must be allocated using `new`
  nvram_chooser_t(const char *title, bool ok, json_list_t nvrams);

  // function that is used to decide whether a new chooser should be opened or
```

```
  // we can use the existing one. The contents of the window are completely
  // determined by its title
  virtual const void *get_obj_id(size_t *len) const {
    *len = strlen(title);
    return title;
  }

  // function that returns number of lines in the list
  virtual size_t idaapi get_count() const { return list.size(); }

  // function that generates the list line
  virtual void idaapi get_row(qstrvec_t *cols, int *icon_,
                              chooser_item_attrs_t *attrs, size_t n) const;

  // function that is called when the user hits `Enter`
  virtual cbret_t idaapi enter(size_t n) {
    if (n < list.size())
      jumpto(list[n]);
    return cbret_t();
  }

protected:
  void build_list(bool ok, json_list_t nvrams) {
    size_t n = 0;
    for (auto nvram : nvrams) {
      list.push_back(nvram["addr"]);
      chooser_nvram[n] = nvram;
      n++;
    }
    ok = true;
  }
};

extern action_desc_t action_load_report;

bool show_nvram(json_list_t nvram, qstring title);
bool show_vulns(json_list_t vulns, qstring title);
bool show_guids(json_list_t guid, qstring title);
bool show_protocols(json_list_t protocols, qstring title);
bool show_ppis(json_list_t protocols, qstring title);
bool show_services(json_list_t services, qstring title);
void attach_action_protocols_deps();
void attach_action_modules_seq();
}
```

efiXplorer/efi_utils.h

```
/*
 * efiXplorer
 * Copyright (C) 2020-2024 Binarly
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
```

```
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 *
 */

#pragma once

#include "efi_defs.h"
#include <string>

namespace efi_utils {
arch_file_type_t input_file_type();

bool add_struct_for_shifted_ptr();
bool addr_in_tables(ea_list_t t1, ea_list_t t2, ea_list_t t3, ea_t ea);
bool addr_in_tables(ea_list_t t1, ea_list_t t2, ea_t ea);
bool addr_in_vec(ea_list_t vec, ea_t addr);
bool check_boot_service_protocol_xrefs(ea_t call_addr);
bool check_boot_service_protocol(ea_t call_addr);
bool check_install_protocol(ea_t ea);
bool json_in_vec(json_list_t vec, json item);
bool mark_copies_for_gvars(ea_list_t gvars, std::string type);
bool op_stroff(ea_t addr, std::string type);
bool set_ptr_type(ea_t addr, std::string type);
bool set_ret_to_pei_svc(ea_t start_ea);
bool summary_json_exists();
bool uint64_in_vec(uint64_list_t vec, uint64_t value);
bool valid_guid(json guid);

ea_list_t find_data(ea_t start_ea, ea_t end_ea, uchar *data, size_t len);
ea_list_t get_xrefs_to_array(ea_t addr);
ea_list_t get_xrefs(ea_t addr);
ea_list_t search_protocol(std::string protocol);

ea_t find_unknown_bs_var64(ea_t ea);

efi_guid_t get_global_guid(ea_t addr);
efi_guid_t get_local_guid(func_t *f, uint64_t offset);

ffs_file_type_t ask_file_type(json_list_t *m_all_guids);

json get_guid_by_address(ea_t addr);

qstring get_module_name_loader(ea_t addr);

std::filesystem::path get_guids_json_file();
std::filesystem::path get_summary_file();

std::string as_hex(uint64_t value);
std::string get_table_name(std::string service_name);
std::string get_wide_string(ea_t addr);
std::string guid_to_string(json guid);
std::string lookup_boot_service_name(uint64_t offset);
std::string lookup_runtime_service_name(uint64_t offset);
std::string type_to_name(std::string type);

uint8_list_t unpack_guid(std::string guid);

void op_stroff_for_global_interface(ea_list_t xrefs, qstring type_name);
void op_stroff_for_interface(xreflist_t local_xrefs, qstring type_name);
void set_const_char16_type(ea_t ea);
void set_entry_arg_to_pei_svc();
void set_guid_type(ea_t ea);
```

```
void set_ptr_type_and_name(ea_t ea, std::string name, std::string type);
void set_type_and_name(ea_t ea, std::string name, std::string type);

int log(const char *fmt, ...);
} // namespace efi_utils

uint16_t get_machine_type();
uint32_t u32_addr(ea_t addr);
uint64_t u64_addr(ea_t addr);
size_t get_ptrsize();

#if IDA_SDK_VERSION >= 900
tid_t import_type(const til_t *til, int _idx, const char *name);
#endif
}
```

The End

This PDF was generated by gitprint.me