BLOG

Riding Dragons: capa Harnesses Ghidra

COLTON GABERTAN, MIKE HUNHOFF, MORITZ RAABE, WILLI BALLENTHIN

FEB 14, 2024 6 MIN READ

FLARE MALWARE REVERSE ENGINEERING

capa is the FLARE team's open source tool that detects capabilities in executable files. Ghidra is an open source software reverse engineering framework created and maintained by the National Security Agency Research Directorate. With the release of capa v7, we have integrated capa with Ghidra, bringing capa's detection capabilities directly to Ghidra's user interface. With this integration, we hope to positively impact the workflows of Ghidra's large user base by helping Ghidra users quickly identify code that suggests an interesting behavior. We are excited to share this work with the community!

This integration was implemented by Colton Gabertan as part of a Google Summer of Code (GSoC) project that the Mandiant FLARE team mentored in 2023. To learn more about the program and our open source contributors check out the introductory post. See our previous capa v7 blog post to learn more about other new features and improvements included in the release.

Integrating capa with Ghidra

Our integration includes two Python 3 scripts, capa_explorer.py and capa_ghidra.py, that you can execute to display capa results in Ghidra. You may be asking yourself, "Python 3 scripts in Ghidra?". You read that correctly. This implementation is written entirely in Python 3 and relies on Ghidrathon, an open source Ghidra extension that adds Python 3 scripting to Ghidra.

Installation

You can find the source code for this integration on our GitHub page. After completing the installation steps outlined in the README, copy capa_explorer.py and capa_ghidra.py to your ghidra_scripts directory or manually add the parent directory of each script using Ghidra's Script Manager window. You can then execute the scripts and interact with the output as described in the following.

Ghidra UI Integration

capa_explorer.py renders capa results in Ghidra's UI to help you quickly navigate to them. This includes adding noteworthy functions to Ghidra's Symbol Tree and Bookmarks windows and adding comments to functions that indicate matched capabilities and features. The bookmarks make it really easy to jump to the interesting part of a program while the comments help to summarize the function you're currently studying. You can execute this script using Ghidra's Script Manager window.

Symbol Tree Window

Matched functions are displayed in Ghidra's Symbol Tree window under custom namespaces that align with capa's rule namespaces as shown in Figure 1. You can navigate directly to a matched function by double-clicking an entry. We like to use this to quickly navigate to functions that interact with the host, such as by setting a persistence mechanism, so that we can extract Indicators of Compromise.

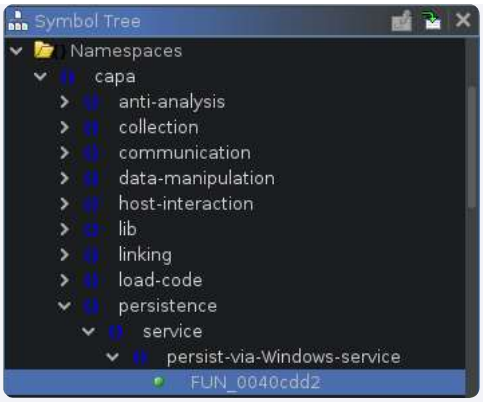

Figure 1: Custom namespaces in Ghidra's Symbol Tree window display matched functions

Comments

Comments are added that show which capabilities and features match within a function. Matched capabilities are commented at the start of a function and matched features are commented inline. By reviewing these comments from capa, you can get a sense for the functionality for a region of code with just a glance, making it easier to find the important parts. You can view comments in Ghidra's Disassembly Listing and Decompile windows as shown in Figure 2.

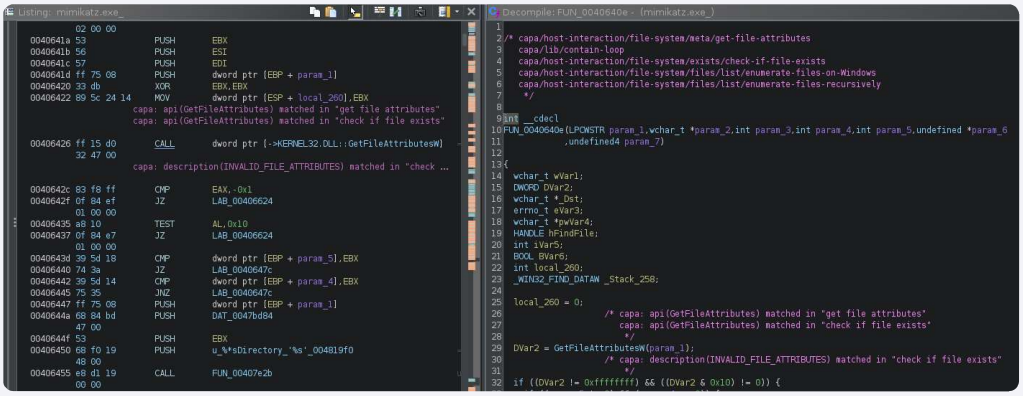

Figure 2: Comments in Ghidra's Disassembly Listing and Decompile windows indicate matched capabilities and features

Bookmarks

capa rules may map to one or more techniques described by the MITRE ATT&CK Framework and Malware Behaviour Catalog (MBC). Bookmarks are added to functions that match a capa rule that is mapped to one or both of these frameworks. You can view these bookmarks in Ghidra's Bookmarks window, including

descriptions that help you pivot to the appropriate MITRE ATT&CK and MBC documentation. An example is shown in Figure 3.

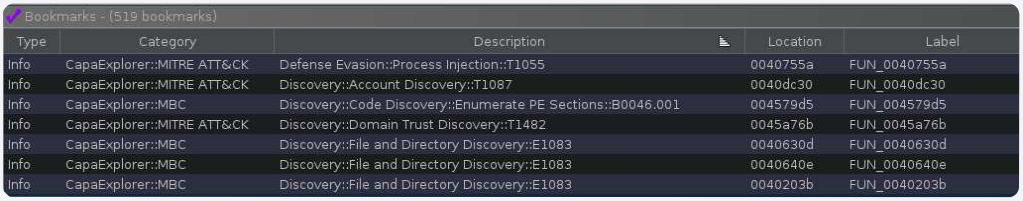

Figure 3: Bookmarks in Ghidra's Bookmarks window highlight functions that match a capa rule mapped to MITRE ATT&CK or MBC

Ghidra Text-based Integration

capa_ghidra.py outputs text-based capa results that mirror the output of capa's standalone tool. Note that this takes into account any interactive changes you've made to the database, which is particularly useful after you've unpacked a malware sample and uncovered some hidden code. You can execute this script using Ghidra's Script Manager to view its output in Ghidra's Console window as shown in Figure 4.

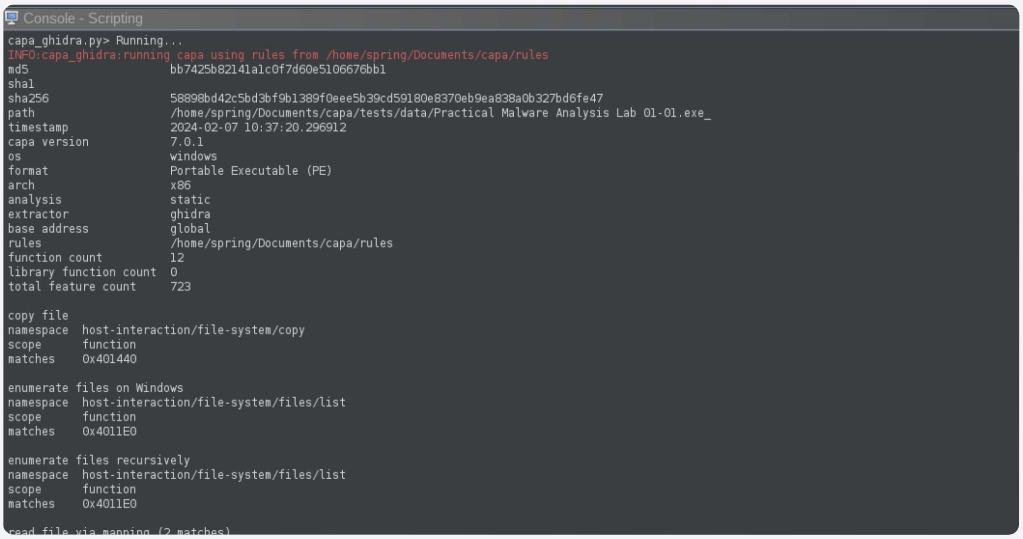

Figure 4: Viewing capa_ghidra.py output in Ghidra's Console window

You can also execute capa_ghidra.py using Ghidra's Headless Analyzer to view its output in a terminal window. This option lends itself nicely to automation and batch processing. The following is an example Ghidra Headless Analyzer command used to execute the script:

$ analyzeHeadless <project_path> <project_name> -Import <sample_path> -PostScript capa_ghidra.py "<capa_rules_path>"

Figure 5 shows output from a variant of the aforementioned command. Check out the Ghidra Headless Analyzer usage section of our README for more information.

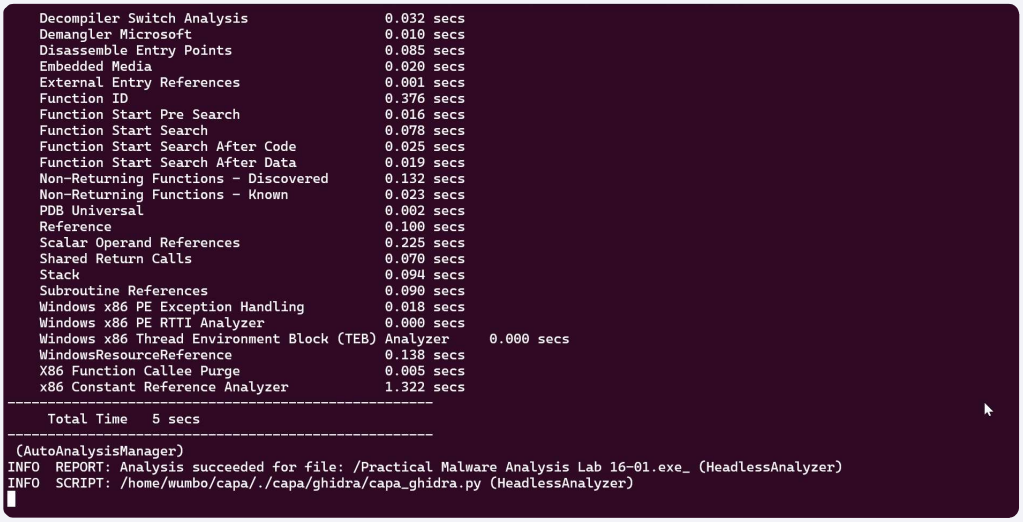

Figure 5: Executing capa_ghidra.py using Ghidra's Headless Analyzer

Implementation

capa can be extended to support new analysis backends. An analysis backend is responsible for extracting program features such as strings, disassembly, and control flow used to detect capabilities. The newly developed Ghidra backend implements all functionality to extract program features from Ghidra's program analysis database. We also created Continuous Integration (CI) workflows that check the correctness of the code.

Feature Extraction

Program features are extracted using Ghidra's API to access functions, basic blocks, and instructions. For example, Ghidra's FunctionManagerDB class is used to identify imports, exports, and user-defined functions and the FunctionDB class is used to identify basic blocks and instructions. We found the Ghidra Javadoc to be useful when experimenting with these classes and other Ghidra APIs. Since capa focuses on key features of programs we hope that our code also serves as inspiration and reference on how to extract relevant data using the API.

Testing and Continuous Integration (CI)

Creating quality tests is important when contributing to a large open source project like capa. Automated testing can help identify issues that would otherwise be hard to detect via manual testing and contribute greatly to the continuity of the project. We created a CI workflow that tests the capa Ghidra integration across multiple Python 3 versions. This workflow uses Github Actions to install capa, Ghidra, and Ghidrathon and then executes a series of tests that ensure our code functions correctly. An example is shown in Figure 6.

```
INFO  REPORT: Analysis succeeded for file: /home/runner/work/capa/capa/./tests/data/mimikatz.exe_ (HeadlessAnalyzer)
INFO  SCRIPT: /home/runner/work/capa/capa/./tests/test_ghidra_features.py (HeadlessAnalyzer)
INFO  Addings configuration to user settings at /home/runner/.ghidra/.ghidra_10.3_PUBLIC/GhidrathonConfig.xml
(GhidrathonUtils)
============================= test session starts ==============================
platform linux -- Python 3.8.18, pytest-7.4.2, pluggy-1.3.0
rootdir: /home/runner/work/capa/capa
plugins: instafail-0.5.0, cov-4.1.0, sugar-0.9.7
collected 161 items

tests/test_ghidra_features.py ssssssssssssssssssssssssssssssssss..... [ 26%]
.................................................................... [ 70%]
.........................sssssssssssssss..                          [100%]

====================== 109 passed, 52 skipped in 8.08s =======================
Script /home/runner/work/capa/capa/./tests/test_ghidra_features.py called exit with code 0
INFO  ANALYZING changes made by post scripts: /home/runner/work/capa/capa/./tests/data/mimikatz.exe_ (HeadlessAnalyzer)
INFO  REPORT: Post-analysis succeeded for file: /home/runner/work/capa/capa/./tests/data/mimikatz.exe_
(HeadlessAnalyzer)
INFO  REPORT: Save succeeded for: /mimikatz.exe_ (ghidra_test:/mimikatz.exe_) (HeadlessAnalyzer)
INFO  REPORT: Import succeeded (HeadlessAnalyzer)
```

Figure 6: GitHub Action CI workflow output for capa Ghidra integration

Future Work

There are multiple steps needed to configure capa for use with Ghidra as this integration relies on several projects including capa, Ghidra, and Ghidrathon. We would like to create a Ghidra extension to simplify the installation process and mirror the functionality of the existing capa explorer plugin for IDA Pro.

Ghidra implements its FunctionID analyzer to identify library functions. This is a similar concept to IDA's Fast Library Identification and Recognition Technology (FLIRT) technology. The capa standalone tool identifies library functions using a FLIRT matching engine implemented in Rust and an open-source FLIRT signature set that covers many library functions encountered in programs compiled using Visual Studio. With these signatures, the capa standalone tool can better detect and ignore library functions than without signatures, improving performance. We would like

to integrate this FLIRT matching engine with Ghidra to complement its FunctionID analyzer. This could expand Ghidra's ability to identify library functions benefiting our capa Ghidra integration and others looking to use FLIRT signatures with Ghidra's analysis.

Conclusion

With the release of capa v7, we bring capa's detection capabilities directly to Ghidra's UI. Our goal with this integration is to enhance your Ghidra workflows by showing you what parts of a program suggest an interesting behavior. We hope that after reading this post you are excited to try this integration and look forward to hearing your feedback and ideas.

Colton's Acknowledgements

Overall, GSoC 2023 was an invaluable experience and it immensely improved my programming skills, development methodology, and collaboration ability in the world of open source software. It further aligned my passion for binary analysis as well as software engineering by providing the best environment possible to learn something new and grow each day. It is always a pleasure to work with the FLARE team, and my GSoC mentor, Mike Hunhoff, who was able to reinforce each core competency needed for a successful outcome! Furthermore, his support of the adjacent project, Ghidrathon, was an integral aspect in completing my GSoC project. I hope that the Ghidra integration continues to mature and I am excited to see what capa has in store for the future.