Function Identification and Recovery Signature Tool

Angel M. Villegas
Cisco Systems Inc., Talos
anvilleg@cisco.com

Abstract

Reverse Engineering benign or malicious samples can take a considerable amount of time. Reversing many samples, or tracking changes in malware families, can cause an analyst to see similar or even the same functions used over and over. The similar, or same, functions could be seen recently, allowing the analyst to recall the metadata they associated with it. However, most likely, the disassembly will be familiar but the analyst will need to review the function to associate metadata with it. Broadening this to a team setting, time and effort is required to keep everyone's metadata sync-ed up between the same or similar samples. Reverse engineering the same routines is a waste of time and can be reduced by applying the right reverse engineering collaborative framework. In this paper a solution is provided for transferring knowledge to similar functions by introducing a new reverse engineering tool, named FIRST (Function Identification and Recovery Signature Tool), to reduce analysis time and enable information sharing.

Keywords: Reverse engineering, Hex-Rays IDA, disassembly, Python, reverse engineering collaboration, client-server framework, FIRST, Function Identification and Recovery Signature Tool

1. Introduction

With the computer security industry expanding, more occupations leverage reverse engineering: computer researcher, cyber security analyst, exploitation developer, vulnerability analyst, etc. Though the day to day of each occupation can differ, many find themselves looking at source code for scripting languages, disassembly for compiled languages, or both. The leading industry tool for analyzing disassembly is Hex-Rays' IDA. IDA is a multi-processor disassembler and debugger that allows users to analyze software, associate metadata with routines within the software and saves progress to a project file (IDB). The IDB allows users to share their analysis via sending others their IDB file.

The ability for users to markup functions with useful information like the function name, its prototype, comments, etc. greatly impacts the ability to analyze software quickly and efficiently. Though IDA has many features and supports developer created plugins to expand its capabilities, it does not provide an easy way to work with others.

Analyzing disassembly for a given binary is typically an individual task versus a group effort. However, several reverse engineers can work together to quickly analyze a binary or others could benefit from the knowledge gleamed and documented by another analyst. An IDA IDB contains function annotations, but those annotations cannot be shared or restored outside of the IDB. Several use cases exist for the need to share work among analysts:

- User A analyzes a function in one sample and re-analyzes it in another sample. Time is wasted either via the user re-analyzing the routine or attempting to find where the function was analyzed before.
- User A and User B analyze the same function. Duplication of effort occurs and time is wasted.
- A team is working together to analyze software. Team members need to work hard to keep synchronized with everyone's annotations.
- User A and User B are using different versions of IDA. Newer versions of IDA create IDBs that cannot be distributed and used by older versions due to version restrictions. Completed work is inaccessible to users resulting in wasted time.

Other tools exist that provide a subset or similar functionality to IDA[1], whereas some analysts use proprietary solutions, scripts, and/or procedures to accomplish similar results. However, much like IDA, they do not provide an easy way to add, update, or synchronize function annotations. Function

[1] Radare2, Binary Ninja, Online Disassembler (ODA) to name a few

Identification and Recovery Signature Tool (FIRST) is framework. This paper focuses on the client IDA plugin and alludes to the standalone API that allows users to add function metadata to a repository and search the repository for function metadata similar to the function(s) they are searching for. A function's name, prototype, and repeatable comment are saved and made available for others.

2. Related Work

There has been several attempts to create a full collaborative framework for IDA and its user base. As of today, many of these projects are abandoned, no longer actively developed and thus do not support newer versions of IDA. Additionally, the IDA software development kit (SDK) and exposed Python application program interface (API) are actively being developed. Each new version of IDA supplies developers with more ways to integrate with IDA, however, the functionality for many of the related works was not available at the time of their creation. A major change occurred from IDA 6.8 to IDA 6.9 when the creators migrated to using Qt 5.4.1 instead of Qt 4.8.4, thereby forcing all developers using IDAPython with PySide to update their plugins to use PyQt5 [1].

The IDA Pro Book 2nd Edition showcases the author's (Chris Eagle) collaboration tool, collabREate [2]. The project is open source and hosted on GitHub[2], but has not been updated since May 2014. collabREate uses a client-server model that stores data out-of-band (external from the IDB). The server stores projects that allow users to enter/exit projects to synchronize efforts for the samples in the project. The framework broadcasts messages and updates to each client connected to the server to force synchronization. Complications arise when analysts disconnect from the server update or add content and then reconnect. The server contains a management application to see server connections, manage users and projects. collabREate supports IDA 4.9 through 6.5 (including IDA freeware 5.0) by leveraging the SDK and compiling plugin versions for each platform and architecture. Users are required to setup a Java/JDBC and PostgreSQL or MySQL database. Users cannot opt-out of any updates, either they get all updates by entering the project or they get no updates by exiting the project.

CrowdStrike released a free Crowdsource Reverse Engineering (CrowdRE) service June 2011 at RECon[3] [3]. CrowdRE uses a client-server model too, however, it is a proprietary product of CrowdStrike. The IDA plugin component is a compiled binary for each platform and architecture supported. The server component is a service offered by CrowdStrike and doesn't allow users to setup their own server or environment in a closed network. The service had great promise when it debuted; however, the service has been unavailable for quite some time.

3. Framework Description

FIRST is an open source client-server framework. It was designed to reduce duplication of effort, decrease time wasted by synchronization, provide a workflow with minimal additional overhead, and integrate into existing disassemblers, scripts, and/or procedures. The client side includes an IDAPython plugin and a standalone API. The server side includes a web site, a representational state transfer (REST) API, and a framework to allow extensibility via a modular plugin design. FIRST provides users with a framework to:

- Add, modify, delete their function annotations within FIRST
- Search for function annotations within FIRST
- Update applied function annotations from FIRST
- View a function's metadata history within FIRST
- Develop server side plugins (Engines and DBs) to expand FIRST's functionality

There are a couple of design decisions that affect the requirements for different components of the framework. The client side standalone API is a Python 2.7 module that requires Python Requests module (can be installed via "pip install requests"). Python 2.7 was chosen since IDA leverages Python 2.7 and to provide a way for earlier versions of IDA to interact with the FIRST server. The client side IDAPython plugin also requires the Python Requests module, and was created as an IDAPython plugin instead of a compiled plugin to prevent the need for different builds for each platform and architecture. An IDAPython plugin also brings a sense of openness, making it easier for people to contribute to the project or even adapt FIRST to their environment. To minimize the workflow and prevent varying levels of functionality the minimum version of IDA is 6.9 sp1 (released 22 Feb. 2016). The freeware version of IDA (currently 5.0) does not include IDAPython, thus it is not supported. As of the Beta release, FIRST server side framework leverages Apache, Python, Django, and MongoDB.

The complete source for FIRST is hosted on GitHub. The quickest way to get going is to use the

[2] https://github.com/cseagle/collabREate

[3] https://recon.cx/2012/schedule/attachments/51_recon-crowdre-final-120621174609-phpapp02.pdf

publicly available FIRST server (http://first-plugin.us) and install the IDAPython plugin. Installation of the plugin requires the user to add the first.py file to their IDA plugins directory (ex. for Windows 10, C:\Program Files (x86)\IDA 6.9\plugins) and install the Requests Python module. Users need to register on the website before they can start using FIRST. Currently, user registration is handled by OAuth2 services. For the Beta version of FIRST, only Google OAuth2 is supported. FIRST only obtains the user's email address and name from the OAuth service. Once registered, the user receives a unique API key that is required by the standalone API and IDAPython plugin. If FIRST is not configured, the user will be presented with a Welcome screen. The welcome screen asks the user to provide FIRST connection details and their API key. Before applying the configuration to FIRST users can test if the configuration connects to the server.

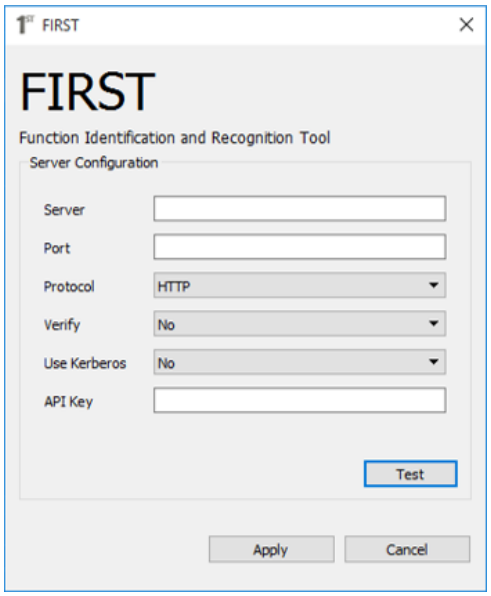

Figure 1. FIRST welcome screen

The installation is now complete. The configuration can be modified at any time by opening the plugin management window (IDA Menu: Edit > Plugins > FIRST) and selecting "Configuration" from the left panel.

The IDAPython plugin allows users to add their annotations (function's name, prototype, and IDA repeatable function comment) to the server. With the plugin, a user can add one or more function annotations at a time, search for existing metadata that is similar to the functions being queried, view function metadata history, delete metadata the user has created and check for updated versions of the metadata currently applied in their IDB. The plugin integrates with IDA to provide the user with a graphical user interface (GUI) to perform these operations, send the data required by the server, display results back to the user, and apply metadata to the IDB. The plugin tries to add as little additional overhead to the reversing process by not requiring the end user to perform additional steps when adding their metadata to the IDB. Instead, an end user will perform their analysis as usual, making sure to leverage repeatable function comments, and then specify which functions they want to add to FIRST. All features are user driven (on-demand) instead of forcing automatic synchronization. Operations are available via the right click menu in IDA's Disassembly View window.

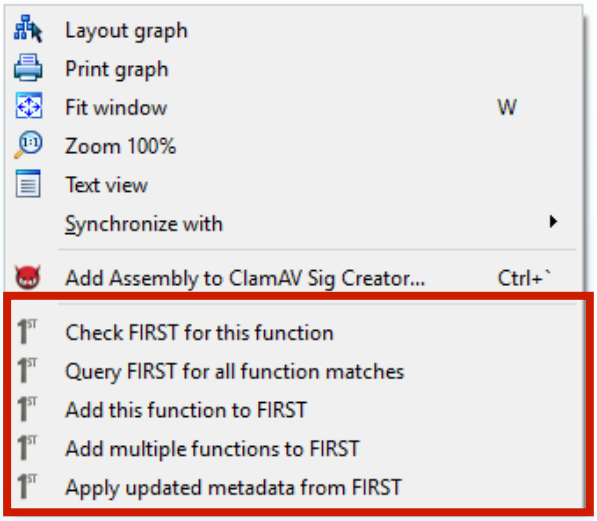

Figure 2. Right click menu: FIRST operations

Each operation will interact with the user through GUI dialog boxes and output summarization of results or errors to the Output Window. The IDAPython plugin works with various threads within IDA to interact with the server, the underlying IDB, and the user via GUIs while ensuring critical non thread-safe functionality is executed within the context of IDA's main thread. Each operation will be discussed further in the Evaluation section.

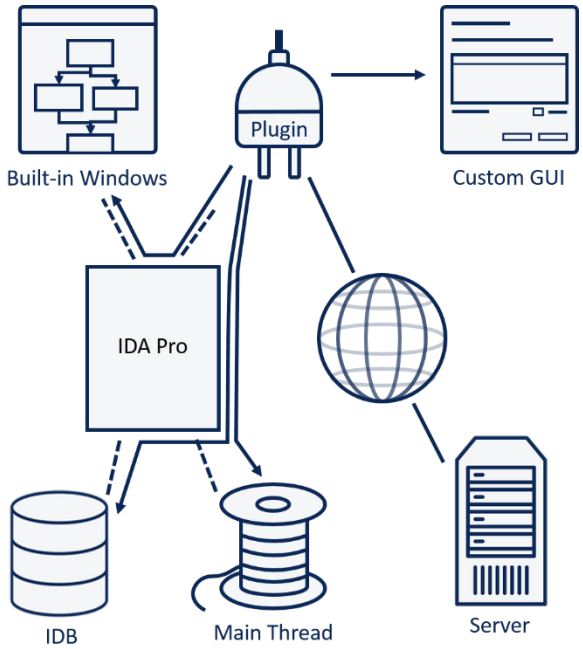

Figure 3. FIRST plugin diagram

The second part of the client side component is a standalone Python module that implements functionality to expose the REST API to the user or scripts utilizing the module. IDA is not the only disassembler publicly available. The Python module was facilitated to enable FIRST integration with other software systems. Developers in different environments will be able to interface with the server in their own environments, even if that environment does not include IDA.

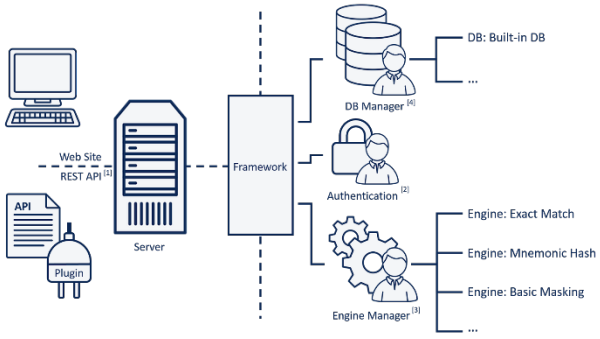

Figure 4. FIRST server diagram

The server side component is a framework incorporating a web service for user registration and interacting with the plugin. Once registered, the user will receive a unique API key that is required by the plugin. The plugin will interact with the server via an exposed REST API. Any data sent to the server will enter the modular framework. The framework includes several components: [1] a web REST API (responsible for providing a RESTful API to the client, validating input data, and returning results or error messages to the client) [2] an authentication module (validate logins via the web user interface and/or checking API keys) [3] an Engine manager (responsible for interacting with the various installed engines to add new function metadata or retrieve it), [4] a database (DB) manager (responsible for providing a layer of abstraction for engines to interact with various databases without implementing specific functionality in the engine themselves).

The framework supplies an abstract implementation of a DB object, allowing developers to create their own DB modules that can be incorporated with the engines. The default installation of FIRST includes a DB module for getting data from the database FIRST uses to store all of its information. Developers are not required to only use FIRST's database but can leverage completely different databases for storing data. This allows FIRST to be integrated into pre-existing workflows without engineering a completely new system.

The framework supplies an abstract implementation of an Engine object, allowing developers to create their own engines to expand and/or enrich FIRST's capabilities. The Engine Manager will dynamically load installed engines. Once installed, an engine will be provided with any incoming function metadata for processing. Adding to the engine's system for storing relevant information (whether that leverages a DB or another means of storage). Engines are given very specific input and expect specific output for the Engine Manager to handle requests from the client. The abstract Engine Class defines what methods and class variables are required by the Engine Manager and includes many wrapper functions to ensure correct data is returned to the Engine Manager. Upon initialization, the Engine Manager will initialize all installed engines by calling their constructors and passing them the DB manager. If the DB manager contains the database connections required by the engine then the engine is operational and added to the list of engines the framework will use for processing requests from the client. However, if the engine does not have the required connections or dependencies, then the engine will be excluded from the framework's list of engines. The Engine is implemented by the developer, and can use any system for finding similar function as long as it implements two required methods (Add and Scan as labeled in the diagram). The framework treats every engine as a blackbox that expects predefined input and outputs predefined data.

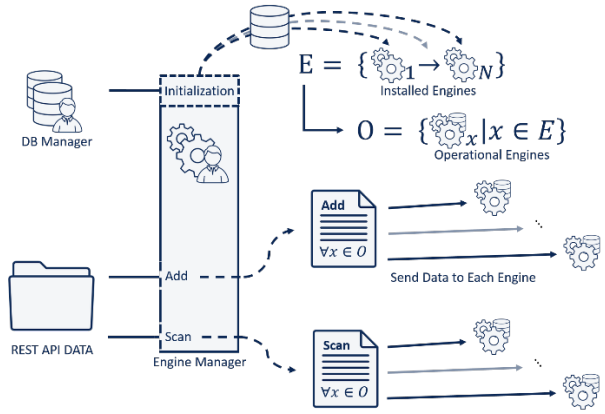

Figure 5. FIRST engine manager diagram

The Beta version of FIRST comes with three Engines built-in: Exact Match, Mnemonic Hashing, and Basic Masking.

Exact Match relies on all opcodes associated with the function. The opcodes are used to create a SHA256 hash for quicker lookups.

```
55 89 E5 53 57 56 83 E4 F8 83 EC 40
8B 45 08 C7 44 24 24 C5 5D 00 00 89 → SHA256
44 24 20 66 C7 44 24 2A F5 27
```

Figure 6. Engine: Exact matching logic

Mnemonic Hashing relies on converting the opcodes associated with the function to their respective disassembly. FIRST currently uses distorm3 to disassemble Intel x86 and x64 opcodes. Once the disassembly is obtained, all operands are stripped away, leaving only the function mnemonics. Those mnemonics are used to create a SHA256 hash for quicker lookups.

Figure 7. Engine: Mnemonic hashing logic

Basic Masking relies on converting the opcodes associated with the function to their respective disassembly. Then the disassembly is normalized by removing ESP/EBP/RSP offsets, absolute calls, relative calls and jump offsets, and global offsets.

Figure 8. Engine: Basic masking logic

4. Evaluation

Registration is required to use FIRST. If a private FIRST server is setup then substitute http://first-plugin.us for the location of the private FIRST server. By default FIRST uses Google's OAuth 2 service to register users. Once an account is created, via providing a handle, an API key is assigned to the user. If using a private FIRST server, then adapt the authentication module to fit the environment. After the API key is received, the user can configure FIRST either through the Welcome screen (if FIRST is not configured) or through the plugin management window (accessible via Edit > Plugins > FIRST). With the plugin configured, the user can access several operations via the right click menu:

- Check FIRST for this function
  Queries FIRST for the function associated with the current location of the cursor
- Query FIRST for all function matches
  Searches FIRST for all defined functions that are more than just a JMP wrapper function
- Add this function to FIRST
  Adds the function associated with the current location of the cursor to FIRST

- Add multiple functions to FIRST
  Allows the user to select multiple functions to upload to FIRST in a single operation. The function list contains all functions that are more than just a JMP wrapper function
- Apply updated metadata from FIRST
  Gets all functions with FIRST metadata applied in the IDB, queries the server for the latest annotations, and applies annotations to the IDB
- View metadata history
  Displays the annotation changes over the lifetime of the function selected

4.1. Checking FIRST for a function, check for multiple functions at once

The IDAPython plugin allows the user to check FIRST for a single function or multiple functions at once. The GUI organizes functions into a tree structure. For checking a single function, the top level represents the metadata that matched. The user is shown the function name, rank, similarity, prototype, the engines that matched, and the creator of the metadata. Expanding the tree will show the comment associated with the function. Checking multiple functions at once will result in the same tree structure, however, the top level represents the function address and current function name in the IDB, with the number of matches. Expanding the tree will show the same information that was available when checking a single function.

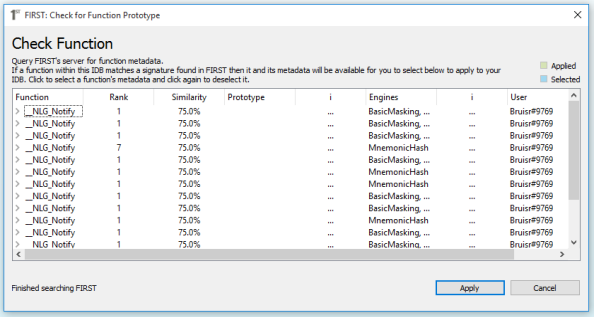

Figure 9. Check a Single Function

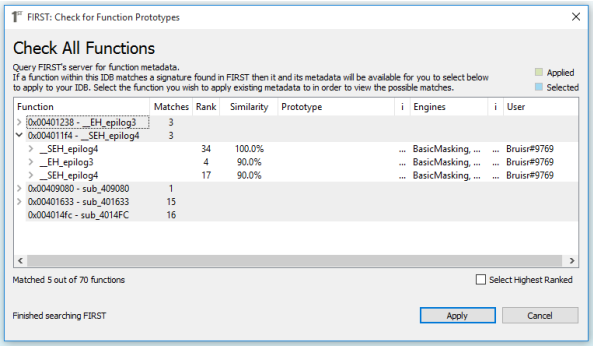

Figure 10. Check Multiple Functions at Once

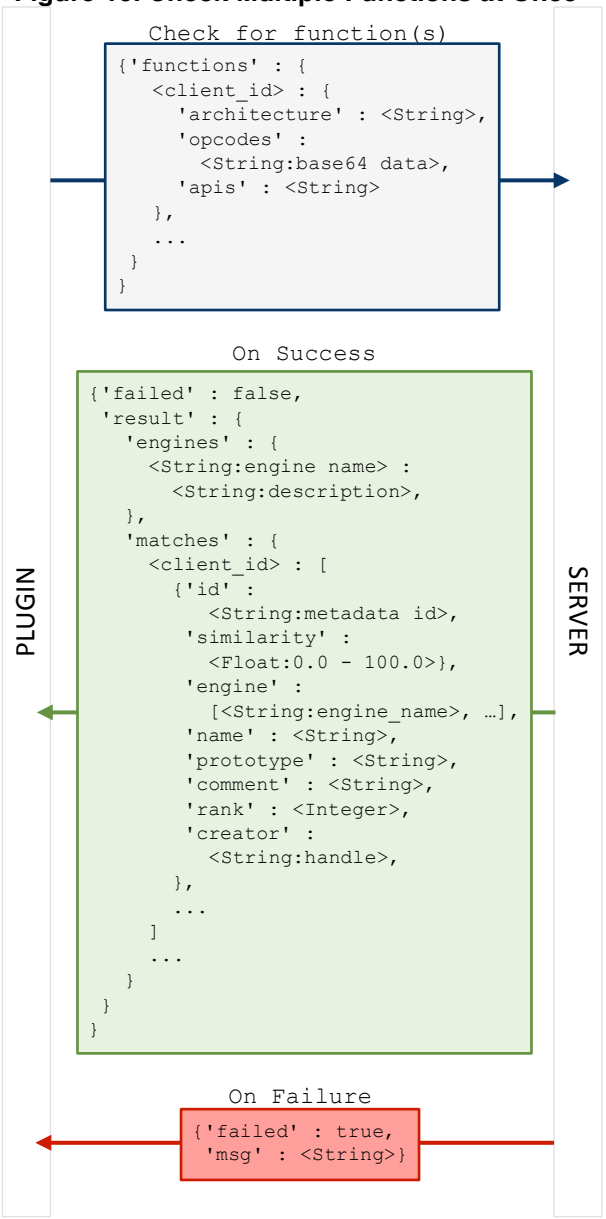

Figure 11. Checking FIRST network flow

4.2. Add function to FIRST, upload multiple functions at once

Uploading a single function will render the GUI on the left, displaying the annotations that will be sent to FIRST. The form is not editable and will require the user to close the dialog and edit the metadata in IDA. Adding multiple functions at once will list all functions that can be added to FIRST, the user will be able to select all the functions (via the checkbox) or click each function they want to add to FIRST. If a function's metadata has been added to FIRST and has

changed since the last upload then the row's background will be a light red color.

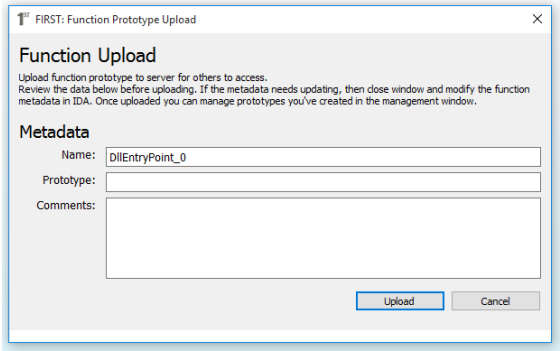

Figure 12. Add a Single Function

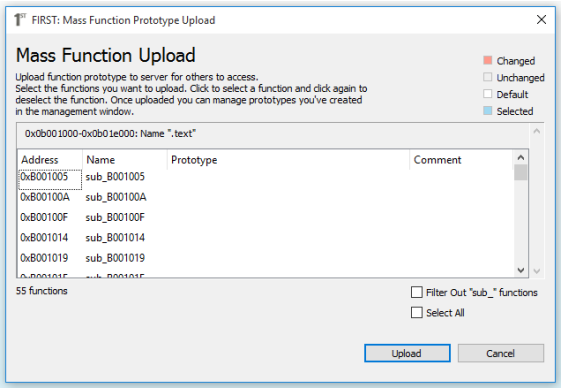

Figure 13. Add Multiple Functions at Once

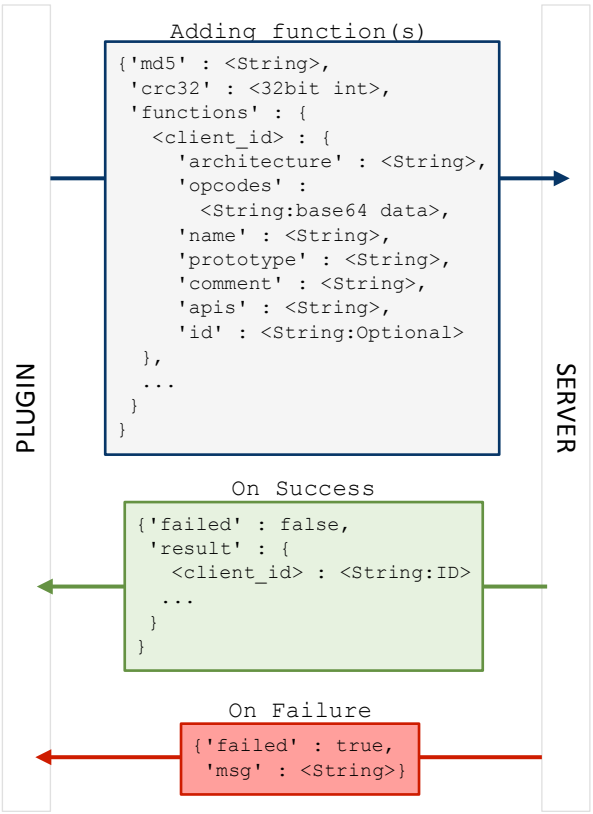

Figure 14. Adding to FIRST network flow

4.3. Updating metadata from FIRST

There is no GUI associated with updating. A user simply right clicks and selects "Apply updated metadata from FIRST."

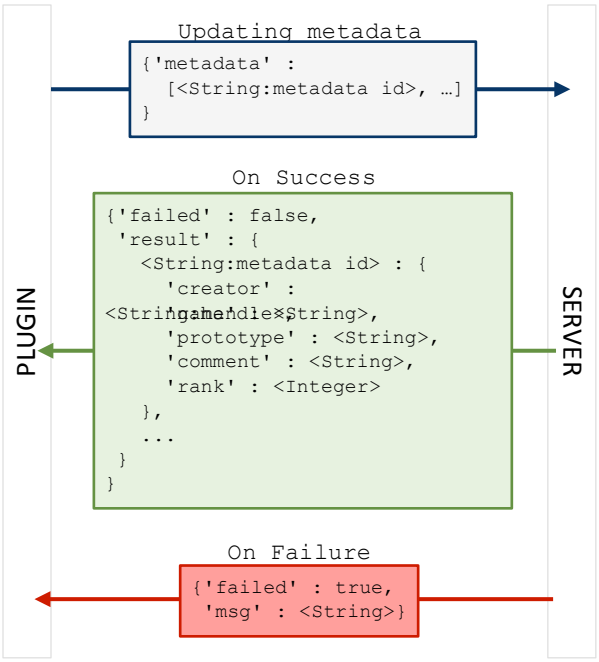

Figure 15. Updating metadata network flow

4.4. Viewing changes to metadata

When metadata from FIRST has been applied to a function in the IDB, a user will be able to right click and select "View metadata history." This view is also available in other dialog boxes via right clicking a row of metadata.

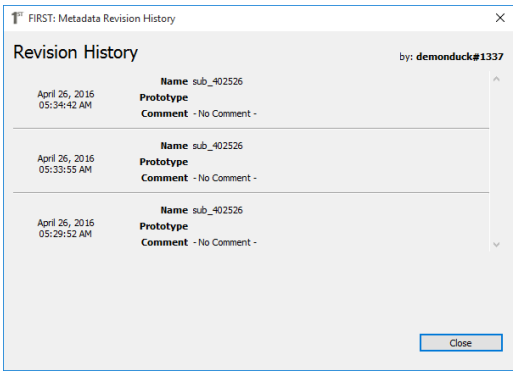

Figure 16. Metadata Revision History

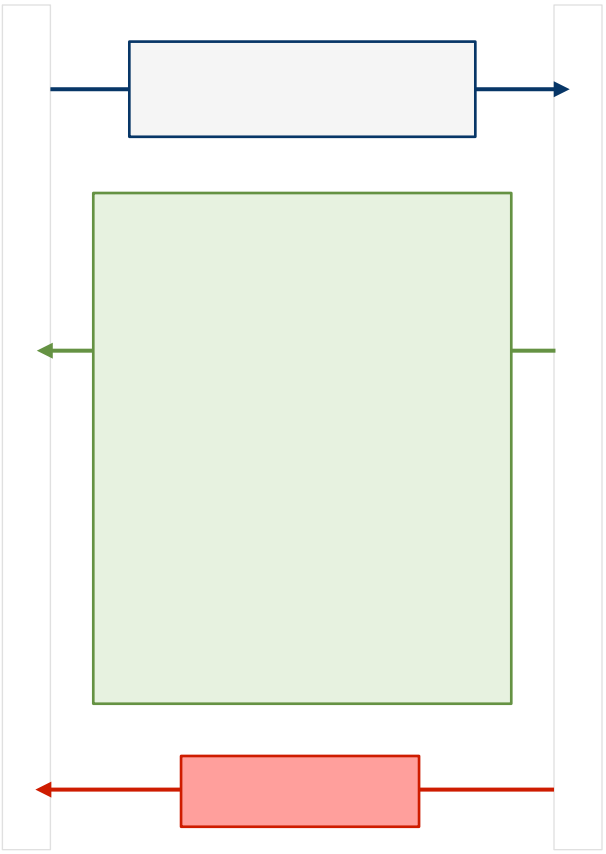

Figure 16. Metadata history network flow

5. Conclusion

Reverse Engineering and all related efforts are time consuming endeavors. That fact alone and the pure number of new benign and malicious samples makes time a valuable resource. To save the time and effort analysts use requires more automation and tools to leverage knowledge.

FIRST's modular framework allows developers to create new Engine components and integrate them into the framework to gain better detection of similar functions. These user developed Engines can be added to the project and made available to everyone or controlled/maintained privately. Additionally, developers can create custom database (DB) components to allow access to other resources (local or external from the FIRST installation). The pairing of Engine and DB modules allow customization not provided by other known prior alternative solutions.

It is the hope of the creators of FIRST to allow users to quickly and easily reduce their workload and/or the amount of time analysts requires by enabling:

- Users with different version of IDA (6.9 and higher) to share annotations without encountering issues due to version changes.
- Users to retrieve functions they have already analyzed in another sample, preventing duplication of work and making synchronization efforts easier.
- Anyone to host their own instance in a public or private space, thereby allowing collaboration on small to large scales.
- Users of varying levels of skill to gain helpful information from community contributions.
- Users to leverage FIRST outside of IDA, gaining the ability to utilize its server side capabilities, thereby removing the restriction of information and allowing pre-existing systems to use FIRST's framework.

6. Acknowledgements

This project was made possible by Talos' Malware Research Team at Cisco Systems, Inc. Any opinions, findings, or conclusions are those of the author and does not necessarily reflect the views of the author's employer.

7. References

[1] Diederen, Arnaud. "IDAPython: migrating PySide code to PyQt5." Hex Blog: State-of-the-art code analysis. Hex Rays, 30 Dec. 2015. Web. 5 July 2016. http://www.hexblog.com/?p=975

[2] Eagle, Chris. "The collabREate Plugin for IDA Pro." No Starch Press. 3 Sep. 2008. Web. 5 July 2016. http://www.idabook.com/collabreate/

[3] Geffner, Jason. "Streamlining the Reverse Engineering Process with CrowdRE." CrowdStrike. 22 June 2012. Web. 5 July 2016. https://www.crowdstrike.com/blog/streamlining-reverse-engineering-process-crowdre/