

HW-03: Static Analysis Utility

Aim: To take the source code that was developed in Lecture and add an additional analysis to the program that will extract some useful data from the artifact(s). To perform static analysis of two or more malware samples (from the test_samples folder) highlighting why the information extracted is significant.

Given: In this lab I was provided with the python file metadata_import.py which is developed in the class which extracts the following data from the malware samples and puts it into a global object within the script, and finally commits it into the database.

- MD5, SHA-1, SHA-256 hashes
- File type (as reported by exiftool / “file magic”)
- File size (in bytes)
- File names
- Compile Time
- Creation Time
- Modify Time
- Author
- Company Name
- File Description
- List of sections (if it is a PE32-type file).

Rebuilding the mongodb database using the samples provided:

To rebuild the Mongodb database using the samples provided, I have implemented the following steps.

1. Moved all the files required for the lab from host machine to remnux VM using shared folder and changed the permissions of all the files and folders required.
2. After that I have installed the mongodb using the following commands:

```
sudo apt-get update
```

```
sudo apt-get install mongodb
```

```
mkdir -p /data/db
```

3. Start the mongodb server using the command “mongod”
4. Invoke mongo shell in another terminal using the command “mongo”
5. Installed pymongo module using pip through the following commands.

```
pip install --upgrade pip
```

```
sudo pip install pymongo
```

6.Run the python code provided on all the malware sample files which are present in test_samples folder using the command below and rebuild the database.

```
find test_samples/ | xargs -n 1 ./metadata_import.py -f
```

Analysis on meta data stored in mongoDB using JavaScript:

After successfully extracted all the metadata and stored in the database “cs7038” and the collection “malware” .I have conducted few analysis using the below commands in the mongo shell in order to see the content stored in the database.

```
show databases
```

```
use cs7038
```

```
show collections
```

```
db.malware.find().count()
```

```
db.malware.distinct('type',{})
```

```
var types = db.malware.distinct('type',{})
```

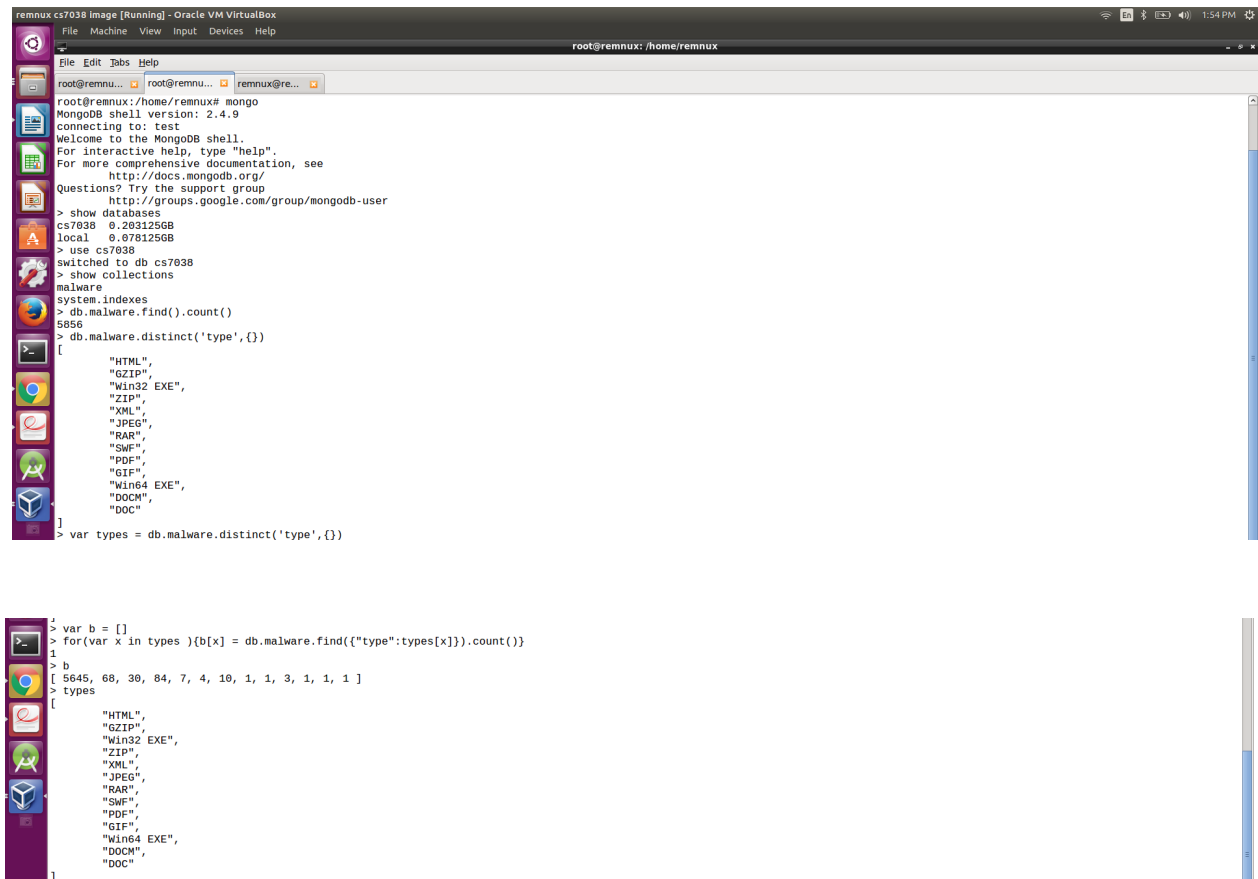
```
b = [ ]
```

```
for(var x in types ){b[x] = db.malware.find({"type":types[x]}).count() }
```

```
b
```

```
types
```

The result of all the analysis is shown in the below screenshots:



```
remnux cs7038 Image [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@remnux: /home/remnux

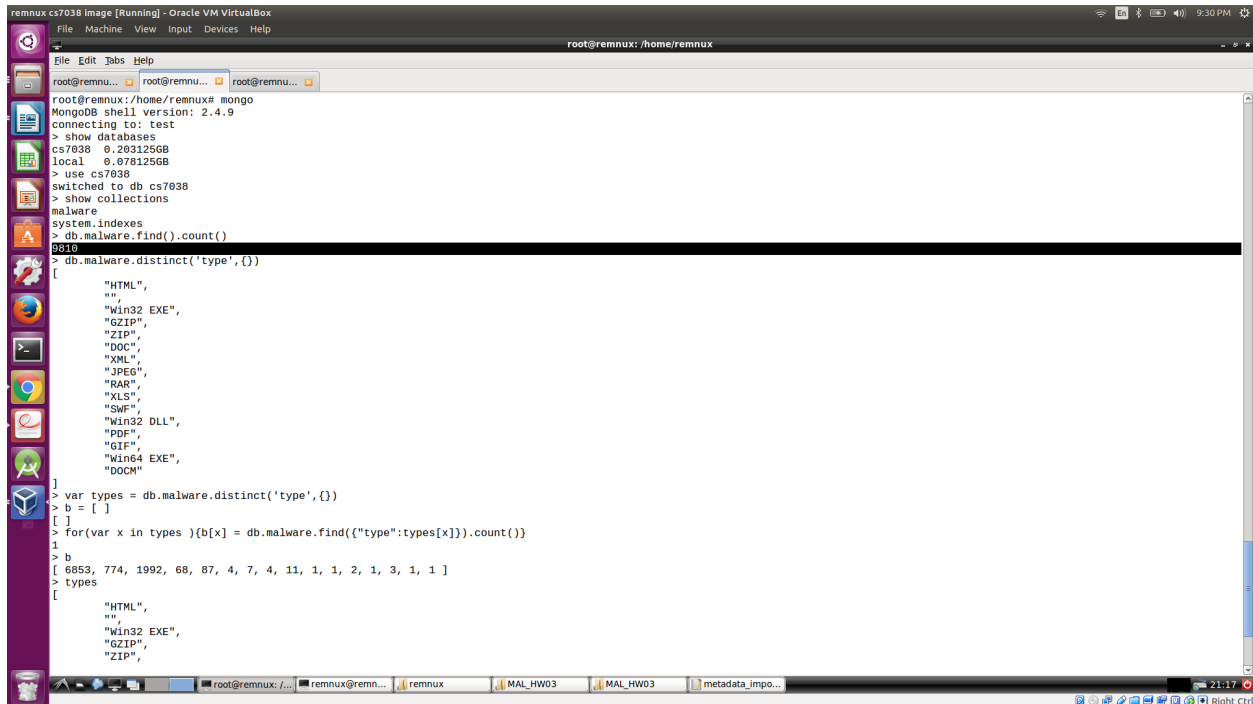
root@remnux:~# mongo
MongoDB shell version: 2.4.9
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
http://docs.mongodb.org/
Questions? Try the support group
http://groups.google.com/group/mongodb-user

> show databases
cs7038 0.203125GB
local 0.078125GB
> use cs7038
switched to db cs7038
> show collections
malware
system.indexes
> db.malware.find().count()
5856
> db.malware.distinct('type',{})
[
  "HTML",
  "GZIP",
  "Win32 EXE",
  "ZIP",
  "XML",
  "JPEG",
  "RAR",
  "SWF",
  "PDF",
  "GIF",
  "Win64 EXE",
  "DOCX",
  "DOC"
]
> var types = db.malware.distinct('type',{})

var b = [ ]
> for(var x in types ){b[x] = db.malware.find({"type":types[x]}).count() }
1
b
[ 5645, 68, 30, 84, 7, 4, 10, 1, 1, 3, 1, 1, 1 ]
> types
[
  "HTML",
  "GZIP",
  "Win32 EXE",
  "ZIP",
  "XML",
  "JPEG",
  "RAR",
  "SWF",
  "PDF",
  "GIF",
  "Win64 EXE",
  "DOCX",
  "DOC"
]
```

Adding the additional Analysis to the program :

After successfully conducting the analysis on the database I have dropped the current database and added few more analysis to the program and started extracting the new meta data from the malware samples and stored them in the database.



```
remnux cs7038 Image [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@remnux: /home/remnux
root@remnux:~# mongo
MongoDB shell version: 2.4.9
connecting to: test
> show databases
cs7038  0.283125GB
local  0.078125GB
> use cs7038
switched to db cs7038
> show collections
malware
system.indexes
> db.malware.find().count()
9810
> db.malware.distinct('type',{})
[
  "HTML",
  "Win32 EXE",
  "GZIP",
  "ZIP",
  "DOC",
  "XML",
  "JPEG",
  "RAR",
  "XLS",
  "SWF",
  "Win32 DLL",
  "PDF",
  "GIF",
  "Win64 EXE",
  "DOCM"
]
> var types = db.malware.distinct('type',{})
> b = [ ]
> for(var x in types ){b[x] = db.malware.find({"type":types[x]}).count()}
> b
[ 6853, 774, 1992, 68, 87, 4, 7, 4, 11, 1, 1, 2, 1, 3, 1, 1 ]
> types
[
  "HTML",
  "Win32 EXE",
  "GZIP",
  "ZIP",
  "DOC",
  "XML",
  "JPEG",
  "RAR",
  "XLS",
  "SWF",
  "Win32 DLL",
  "PDF",
  "GIF",
  "Win64 EXE",
  "DOCM"
]
```

Here I have noticed that more malware samples have been extracted compared to our previous extraction i.e 9810 (now) compared to 5856 (previous).

The additional analysis added to the program extracts the following information.

For html file, we are extracting the following meta data

- External JavaScript links
- External CSS links
- JavaScript function names
- SHA1
- SHA256
- md5
- size
- File names

For pdf file we are extracting the following metadata

- Encodings used in pdf

- URI's in pdf
- Filters in pdf
- SHA256
- SHA1
- md5
- size
- File names

For PE files we are extracting the following metadata

- Sections
- Sections size in order
- Sections address in order
- SHA256
- SHA1
- md5
- size
- File names
- type

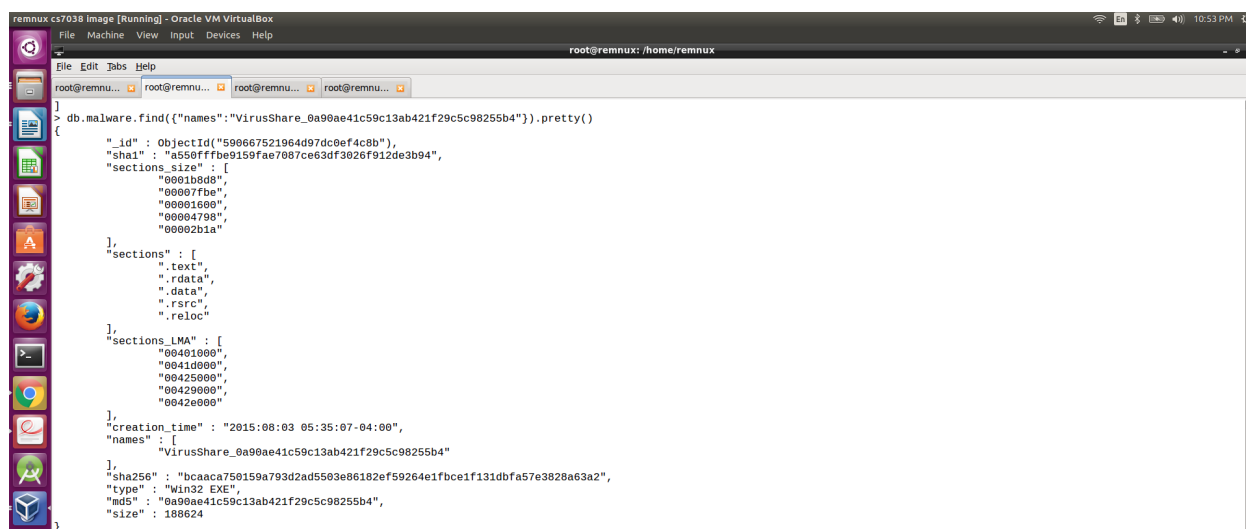
Static Analysis of three malware samples:

The three files that I chose for further Analysis are:

- 1) VirusShare_0a90ae41c59c13ab421f29c5c98255b4 (exe)
- 2) VirusShare_0b6fbf8c1ca8638f38997ecf505682b2 (html)
- 3) VirusShare_0be5918b98256ca07c42b35980806a7c (pdf)

Here I have taken few challenges like extracting the Sections Data,Section names,Sections sizes,Sections LMA from the executables ,javascript links ,javascript functions for html etc.

1) VirusShare_0a90ae41c59c13ab421f29c5c98255b4 (exe)



```
remnux cs7038 Image [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@remnux: /home/remnux

> db.malware.find({"names":"VirusShare_0a90ae41c59c13ab421f29c5c98255b4"}).pretty()
{
  "_id" : ObjectId("590667521964d97dc0ef4c8b"),
  "sha1" : "a550ffffbe9159fae7087ce63df3026f912de3b94",
  "sections_size" : [
    "0001b8d8",
    "00007fbb",
    "00001000",
    "00004798",
    "00002b1a"
  ],
  "sections" : [
    ".text",
    ".rdata",
    ".data",
    ".rsrc",
    ".reloc"
  ],
  "sections_LMA" : [
    "00401000",
    "0041d000",
    "00425000",
    "00429000",
    "0042e000"
  ],
  "creation_time" : "2015:08:03 05:35:07-04:00",
  "names" : [
    "VirusShare_0a90ae41c59c13ab421f29c5c98255b4"
  ],
  "sha256" : "bcaaca750159a793d2ad5503e86182ef59264e1fbce1f131dbfa57e3828a63a2",
  "type" : "Win32 EXE",
  "md5" : "0a90ae41c59c13ab421f29c5c98255b4",
  "size" : 188624
}
```

The hash values for this file are:

sha1: a550ffffbe9159fae7087ce63df3026f912de3b94

sha256: bcaaca750159a793d2ad5503e86182ef59264e1fbce1f131dbfa57e3828a63a2

md5: 0a90ae41c59c13ab421f29c5c98255b4

In the above screen-shot it is clear that sections data has section names which are meaningful and we can say that it may not be the malware. [Because The benign files often contain the basic section name such as .text, .data, .rsrc, etc, or meaningful name like .shared, .page, .init, winzip, etc].

From section sizes it is clear that the size of .rsrc (resources) section is 18328 (integer value). Then the ratio of (resource size/total size), i.e. (18328/188624) is around 0.097. As the ratio is comparatively normal there may not be any malicious code inside of the resource data. [Because attackers using the resources section as a storage place for additional execution code leads to more Resource size / sample size ratio which is susceptible.]

2) VirusShare_0b6fbf8c1ca8638f38997ecf505682b2 (html)

```

> db.malware.find({"names":"VirusShare_0b6fbf8c1ca8638f38997ecf505682b2"}).pretty()
{
  "_id" : ObjectId("50665a881964d92ffcbb02a4"),
  "externalScriptLinks" : [ ],
  "sha1" : "a6e246065805f30160ff039721e1a10e9767cdaf",
  "author" : "alexsys",
  "externalCSSLinks" : [
    "http://alfarealt.by/components/com_osproperty/js/bootstrap/css/bootstrap.css",
    "http://alfarealt.by/components/com_osproperty/js/bootstrap/css/bootstrap-responsive.css",
    "http://alfarealt.by/components/com_osproperty/js/bootstrap/css/bootstrap_adv.css",
    "http://alfarealt.by/modules/mod_ospropertyrandom/asset/style.css",
    "http://alfarealt.by/modules/mod_slogin/tmpl/compact/slogin.css",
    "/components/com_rform/assets/css/front.css",
    "/templates/bee2_20/css/layout.css",
    "/templates/bee2_20/css/jquery.formstyler.css",
    "/templates/bee2_20/css/reset.css",
    "http://fonts.googleapis.com/css?family=Roboto:400,100,500,700&subset=cyrillic,latin"
  ],
  "javascriptFunctions" : [ ],
  "names" : [
    "VirusShare_0b6fbf8c1ca8638f38997ecf505682b2"
  ],
  "sha256" : "823918d41fbf0684d75f33c83ee6ca63ec50909b29af4ebb8a54e83a8e4e8077",
  "type" : "HTML",
  "md5" : "0b6fbf8c1ca8638f38997ecf505682b2",
  "size" : 48573
}

```

The hash values for this file are:

sha1: a6e246065805f30160ff039721e1a10e9767cdaf

sha256: 823918d41fbf0684d75f33c83ee6ca63ec50909b29af4ebb8a54e83a8e4e8077

md5: 0b6fbf8c1ca8638f38997ecf505682b2

There are no external javascript links as in the above image it is empty. But it has external CSS links from the domain <http://alfarealt.by> which looks suspicious of cross site scripting attack. The file does not contain the javascript functions which wont allow attackers to inject the payload.

3) VirusShare_0be5918b98256ca07c42b35980806a7c (pdf)

```

> db.malware.find({"names":"VirusShare_0be5918b98256ca07c42b35980806a7c"}).pretty()
{
  "_id" : ObjectId("506662581964d927a4eac4"),
  "sha1" : "38b5006eec8e5444e83f9f9bd752cb7910f5cba7",
  "names" : [
    "VirusShare_0be5918b98256ca07c42b35980806a7c"
  ],
  "sha256" : "038f7294add219fa8b7ff149eae55b72b75176a04d5c1eff4972d9a40d1d2a3",
  "type" : "",
  "md5" : "0be5918b98256ca07c42b35980806a7c",
  "size" : 6419
}

```

The hash values for this file are:

sha1: 38b5006eec8e5444e83f9f9bd752cb7910f5cba7

sha256: 038f7294add219fa8b7ff149eae55b72b75176a04d5c1eff4972d9a40d1d2a3

md5: 0be5918b98256ca07c42b35980806a7c

The pdf file I have considered for analysis looks fine as it does not have any malicious objects in it.

Submitted by:

Naveen Reddy Aleti

M-ID: 10727908