Sample Scripts for Field

This is a listing of scripts provided by engineering to the field for various scenarios. The listing should be kept up to date with regards to file types and scripts and enable reuse of the scripts.

For more details on how to use the File Type Analyzer, please refer to the Custom File Type Detection Guide.

The File type Analyzer program can be found here

Note: Use the following in the "File Name Filter":

[\w\s]+.[\w]+ : this will ONLY be able to read ascii filenames

or

[^0x00]+.[\w]+ : This will be able to read non-ascii filenames

Password/Encrypted protected rar

Use Case The endpoint cannot detect password/encrypted protected rar through Verity due to the way Verity identifies password protected contents of a rar. This is not a problem on the server. This script allows us to workaround the issue.This will detect if the WHOLE file is encrypted OR if the first file entry is password protected.

Note: This detects password protected rar by seeing if the encryption bit is enabled on the first file entry. Rar allows encryption on a per entry basis OR Encryption on the WHOLE file, so this workaround is relatively weak when it comes to detecting if 1 of many files has a password protected file.

*** Make sure the brackets are on their own lines.

```
$First4rarTag=getHexStringValue('52617221');
$First4rarBytes=getBinaryValueAt($data, 0x0, 4);
assertTrue($First4rarTag == $First4rarBytes);
$Second3rarTag=getHexStringValue('1A0700');
$Second3rarBytes=getBinaryValueAt($data, 0x4, 3);
assertTrue($Second3rarTag == $Second3rarBytes);
$EncryptedrarTag=getHexStringValue('80');
$EncryptedrarBytes=getBinaryValueAt($data, 0xA, 1);
$FileHeaderFlags = getBinaryValueAt($data, 0x18, 1);
$modedValue = mod($FileHeaderFlags, 8);

if ($modedValue >=4)
{
assertTrue($modedValue >= 4);
}
else
{
assertTrue($EncryptedrarBytes == $EncryptedrarTag);
}
```

Password protected zip

Use Case The endpoint cannot detect password protected zip through Verity due to the way Verity identifies password protected contents of a zip. This is not a problem on the server. This script allows us to workaround the issue. However, note that this detects password protected zip by seeing if the encryption bit is enabled on the first file entry. Zip allows encryption on a per entry basis, so this workaround is relatively weak.

```
$pktag=ascii('PK');
$frecord=getHexStringValue('0304');
$pkbytes=getBinaryValueAt($data, 0x0, 2);
assertTrue($pktag == $pkbytes);
$recordbytes=getBinaryValueAt($data, 0x2, 2);
assertTrue($frecord == $recordbytes);
$cryptByte=getBinaryValueAt($data, 0x6, 1);
$encrypted=mod($cryptByte, 2);
assertTrue($encrypted == 1);
```

Symantec Endpoint Encryption

Use Case Detecting if data has been encrypted by SEE. The file type is executable but contains string resources reflecting *EPCL* in many predefined locations. This could be further validated by engaging with SEE engineers.

```
$exetag=getHexStringValue('4D5A9000');
$exebytes=getBinaryValueAt($data, 0x0, 4);
assertTrue($exetag == $exebytes);
$epcltag=ascii('EPCL');
$epclbytes=getBinaryValueAt($data, 0x228, 4);
assertTrue($epcltag == $epclbytes);
```

VmWare Disk Files (.vmdk)

Use Case Detecting if files are VmWare Disk Files. The file type is a .vdmk extension, though NOT all .vdmk files contain actual drive partitions, some are just xml descriptors, which are ignored by this file detection.

```
$vmdktag = ascii('KDMV');
$vmdkhextag=getHexStringValue('01');
$vmdktag2 = getBinaryValueAt($data, 0x0, 4);
$vmdkhextag2 = getBinaryValueAt($data, 0x4, 1);
assertTrue($vmdktag == $vmdktag2);
assertTrue($vmdkhextag == $vmdkhextag2);
```

Amazon Kindle

Details Mobi files are Amazon and have BOOKMOBI in ascii at 0x3c

```
$book=ascii('BOOK');
$mobi=ascii('MOBI');
$word1=getBinaryValueAt($data, 0x3c, 4);
$word2=getBinaryValueAt($data, 0x40, 4);
assertTrue($book == $word1);
assertTrue($mobi == $word2);
$null=getBinaryValueAt($data, 0x3b, 1);
assertTrue($null == 0);
$nullx=getBinaryValueAt($data, 0x44, 1);
assertTrue($nullx == 0);
```

Open Reader E-Book

Details Epub files are open book format using zip with a manifest containing application/epub+zip

```
$slash=getHexStringValue('2f');
$epub1=ascii('epub');
$epub2=ascii('zip');
$slash1=getBinaryValueAt($data, 0xb, 1);
assertTrue($slash == $slash1);
$word1=getBinaryValueAt($data, 0xc, 4);
assertTrue($word1 == $epub1);
$word2=getBinaryValueAt($data, 0x11, 3);
assertTrue($word2 == $epub2);
```

Oracle IRM

Details Oracle provides a DRM solution that prepends an ascii header to the encrypted content of whatever file type was in use. For instance, a doc file becomes .sdoc with the encrypted payload after the ascii header.

```
$soft=ascii('Soft');
$seal=ascii('SEAL');
$word1=getBinaryValueAt($data, 0x0, 4);
$word2=getBinaryValueAt($data, 0x4, 4);
assertTrue($soft == $word1);
assertTrue($seal == $word2);
```

Buffalo SLW Encrypted

Details Buffalo SLW Encrypted files (Used in APJ markets).

The Buffalo Encrypted file, is an Encryption SW that comes preloaded on Japanese or Asian purchased USB devices. We are using this as a possible exception to a rule or detection for blocking.
http://buffalo.jp/products/catalog/flash/ruf2-rvs-sv/
http://buffalo.jp/download/driver/hd/slm.html

```
$slwfirst4tag=getHexStringValue('534C5720');
$second3rartag=getHexStringValue('564F4C55');
$slwfirst4bytes=getBinaryValueAt($data, 0x0, 4);
$second3rarbytes=getBinaryValueAt($data, 0x4, 4);
assertTrue($slwfirst4tag == $slwfirst4bytes);
assertTrue($second3rartag == $second3rarbytes);
```

HWP Files (Hangul Word Processor (Korean))

Details Korea in many cases uses a locally developed word processor called Hangul. This script will allow the detection of these file types (.hwp).

```
$First4hwpTag=getHexStringValue('46006900');
$First4hwpBytes=getBinaryValueAt($data, 0x680, 4);
assertTrue($First4hwpTag == $First4hwpBytes);

$Second4hwpTag=getHexStringValue('6C006500');
$Second4hwpBytes=getBinaryValueAt($data, 0x684, 4);
assertTrue($Second4hwpTag == $Second4hwpBytes);

$Third4hwpTag=getHexStringValue('48006500');
$Third4hwpBytes=getBinaryValueAt($data, 0x688, 4);
assertTrue($Third4hwpTag == $Third4hwpBytes);

$2First4hwpTag=getHexStringValue('44006F00');
$2First4hwpBytes=getBinaryValueAt($data, 0x700, 4);
assertTrue($2First4hwpTag == $2First4hwpBytes);

$2Second4hwpTag=getHexStringValue('63004900');
$2Second4hwpBytes=getBinaryValueAt($data, 0x704, 4);
assertTrue($2Second4hwpTag == $2Second4hwpBytes);

$2Third4hwpTag=getHexStringValue('6E006600');
$2Third4hwpBytes=getBinaryValueAt($data, 0x708, 4);
assertTrue($2Third4hwpTag == $2Third4hwpBytes);
```

GUL Files (Hunmin Word Processor (Korean))

Details Korea in many cases uses a locally developed word processor called Hunmin. This script will allow the detection of these file types (.gul).

*** Make sure the brackets are on their own lines.

```
$4gulTag = getHexStringValue('53414D53');
$2nd4gulTag = getHexStringValue('554E47');

$old4gulBytes = getBinaryValueAt($data, 0x0, 4);
$new4gulBytes = getBinaryValueAt($data, 0x400, 4);

if ($4gulTag == $old4gulBytes)
{
$2Old4gulBytes=getBinaryValueAt($data, 0x4, 3);
assertTrue($2nd4gulTag == $2Old4gulBytes);
}
else if ($4gulTag == $new4gulBytes)
{
$2New4gulBytes=getBinaryValueAt($data, 0x404, 3);
assertTrue($2nd4gulTag == $2New4gulBytes);
}
else
{
$First4gulTag=getHexStringValue('4D006100');
$First4gulBytes=getBinaryValueAt($data, 0x780, 4);
assertTrue($First4gulTag == $First4gulBytes);

$Second4gulTag=getHexStringValue('73007400');
$Second4gulBytes=getBinaryValueAt($data, 0x784, 4);
assertTrue($Second4gulTag == $Second4gulBytes);

$Third4gulTag=getHexStringValue('65007200');
$Third4gulBytes=getBinaryValueAt($data, 0x788, 4);
assertTrue($Third4gulTag == $Third4gulBytes);

$2First4gulTag=getHexStringValue('50004100');
$2First4gulBytes=getBinaryValueAt($data, 0x78c, 4);
assertTrue($2First4gulTag == $2First4gulBytes);

$2Second4gulTag=getHexStringValue('47004500');
$2Second4gulBytes=getBinaryValueAt($data, 0x790, 4);
assertTrue($2Second4gulTag == $2Second4gulBytes);
}
```