MASTIFF

Automated Static Analysis Framework

KoreLogic SECURITY

Tyler Hudak
KoreLogic Security
OSDF 2013

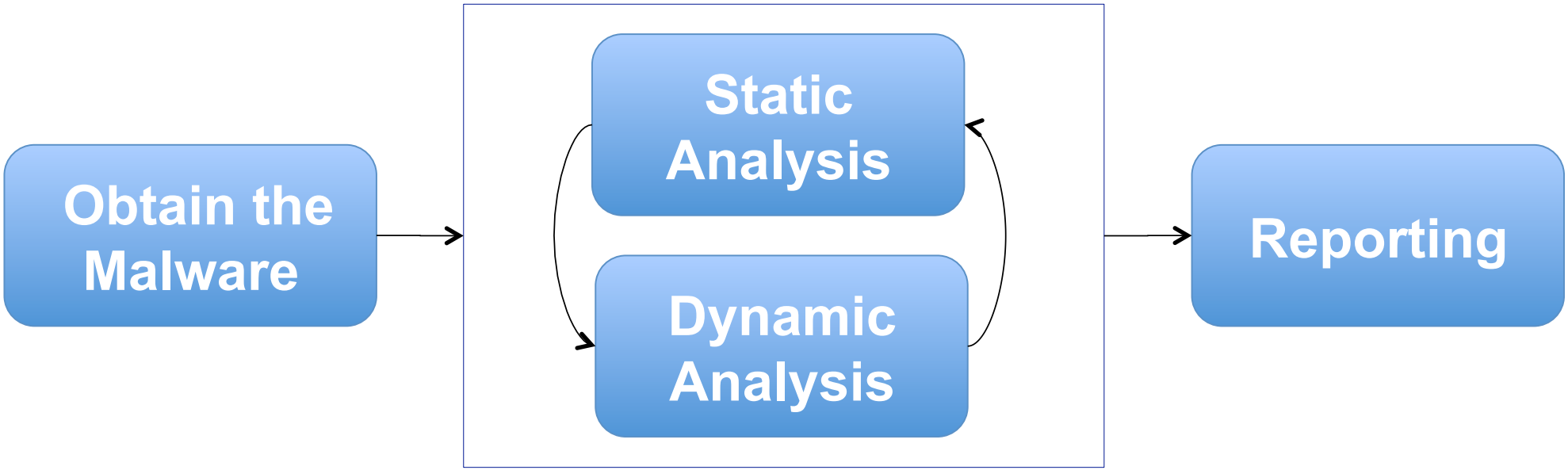
Introduction

- Contact info:

  thudak@korelogic.com
  http://blog.korelogic.com
  http://securityshoggoth.blogspot.com
  @secshoggoth

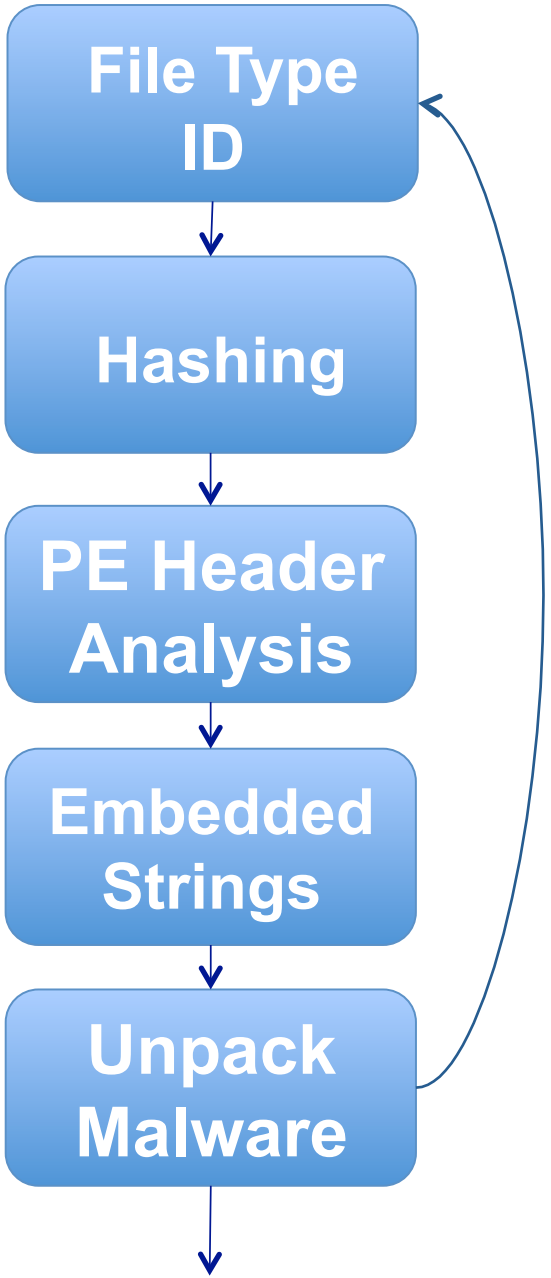
- 15 years in Information Security
- Lead malware analyst/reverse engineer for KoreLogic
- Teach malware analysis courses
- Just a big ol' nerd that loves what he does.

Malware Analysis Process



Static Analysis examines malware characteristics.

- Finds clues as to what the malware does
- Can find:
  - Who created it?
  - What it does?
  - Any exploits within it?
  - Is this old/new?
  - …
- Variety of techniques available



Automating Malware Analysis

- Faster, less prone to mistakes
- Most frameworks focus on dynamic analysis
  - Hardest to do, get the most info out of
  - Some static analysis, but not a lot

There is nothing that fully automates static analysis!

Automation Issues

Automation scripts must be modified for every file type.

```
#!/bin/bash

# set up globals
PECHECK=/usr/local/bin/pecheck.py
PEID=/usr/local/etc/UserDB.TXT
TRID=/usr/local/bin/trid
TRIDDEFS=/usr/local/etc/TrIDDefs.TRD
STRINGS=/usr/local/bin/mystrings
SINGLESTR=/usr/local/bin/singleString.py
```

Some static analysis techniques are universal.
Some are not.

```
$ pdf-parser.py ./VPTray.exe  | more
PDF Comment '%\x00\x00\x00\x00\x83\xec(SVW\x8b]\x083\xff;\xdfu\x0e\xffu\x0c\xe8\xfb\x02\x00\x00Y\xe
00\x8bu\x0c;\xf7u\x0cS\xe8\xde\x1d\x00\x00Y\xe9\xd1\x02\x00\x00\xa1|\xb4@\x00\x83\xf8\x03\x0f\x859\
}\xdc\x83\xfe\xe0\x0f\x87\xf1\x00\x00\x00j\t\xe8\x10\x18\x00\x00Y\x89}\xfcS\xe8\x11\x06\x00\x00Y\x8
\x84\xa4\x00\x00\x00;5\x98\xb4@\x00wLVSP\xe8\xfe\r'

PDF Comment '%\x00\x00\x00\x00\x83\xec\x0cSVW\xa1|\xb4@\x00\x83\xf8\x03uC\x8bu\x08;5\x98\xb4@\x00\x
x00\x00j\t\xe8\xce\x14\x00\x00Y\x83e\xfc\x00V\xe8"\x06\x00\x00Y\x89E\xe4\x83M\xfc\xff\xe8\x0c\x00\x
\x85\xc0tm\xe9\x86\x00\x00\x00j\t\xe8\x04\x15\x00\x00Y\xc3\x83\xf8\x02uZ\x8bE\x08\x85\xc0t\x08\x8dp
0\xeb\x03j\x10^\x89u\x08;5|\xa3@\x00w.j\t\xe8x\x14\x00\x00Y\xc7E\xfc\x01\x00\x00\x00\x8b\xc6\xc1\xe
\x00\x00Y\x89E\xe4\x83M\xfc\xff\xe8\r'

PDF Comment '%\x00\x00\x00\x00\x83\xecXSVW\x89e\xe8\xff\x15Pp@\x003\xd2\x8a\xd4\x89\x15`\xae@\x00\x
xff\x00\x00\x00\x89\r'

PDF Comment '%\x00\x00\x85\xc0u\x08j\x10\xe8\xb2\x00\x00\x00Y3\xf6\x89u\xfc\xe8^#\x00\x00\xff\x15Lp
b4@\x00\xe8\x1c"\x00\x00\xa3\xd4\xad@\x00\xe8\xc5\x1f\x00\x00\xe8\x07\x1f\x00\x00\xe8\x12\x1c\x00\x
E\xa4P\xff\x15Hp@\x00\xe8\x98\x1e\x00\x00\x89E\x9c\xf6E\xd0\x01t\x06\x0f\xb7E\xd4\xeb\x03j\n'
```

Using tools for file-type detection.

UNIX *file* command is most common...

… but not everything is detected correctly.

How do you handle:

A single type used for multiple formats?
Files contained within other files?
Files that are multiple types?

Corkamix -

http://code.google.com/p/corkami/downloads/detail?name=CorkaMIX.zip

A single file that is a valid:

- PDF
- PE Executable
- Java Jar / Python Script
- HTML (w JavaScript)

MASTIFF

Framework that automates the static analysis process.

- Automatically determines the file type.
- Applies only techniques for those file types.

- Open-source
  - http://sourceforge.net/projects/mastiff
- Written in Python
- Extensible using plug-ins

Plug-ins

In order to be easily expandable, MASTIFF utilizes plug-ins for file-type detection and static analysis techniques.

Category Plug-ins

Analysis Plug-ins

Implemented using Yapsy (http://yapsy.sourceforge.net/).

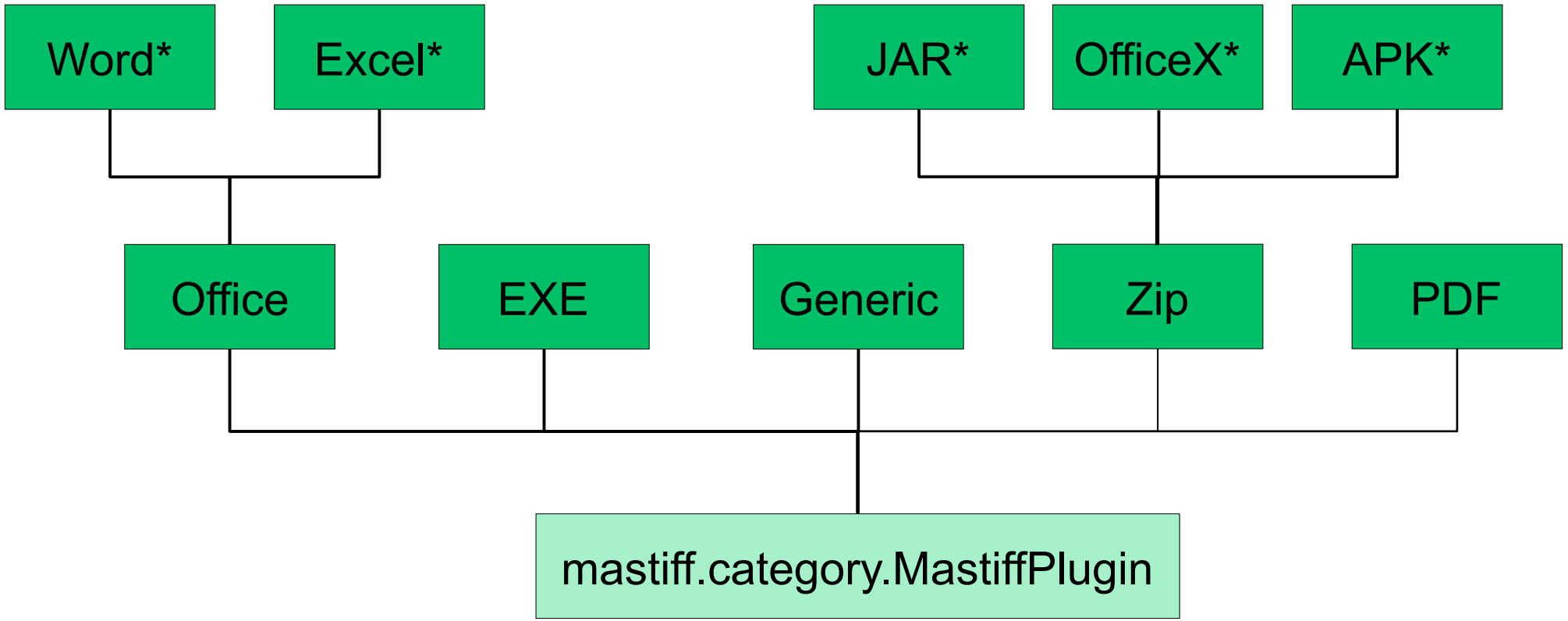
Category Plug-ins

- Determine the type of file that is being examined
- Group analysis plug-ins into file type categories

Current category plug-ins:

- Windows Executables
- PDF Documents
- Office Documents
- Zip Archives
- ”Generic”

Category Plug-in Architecture

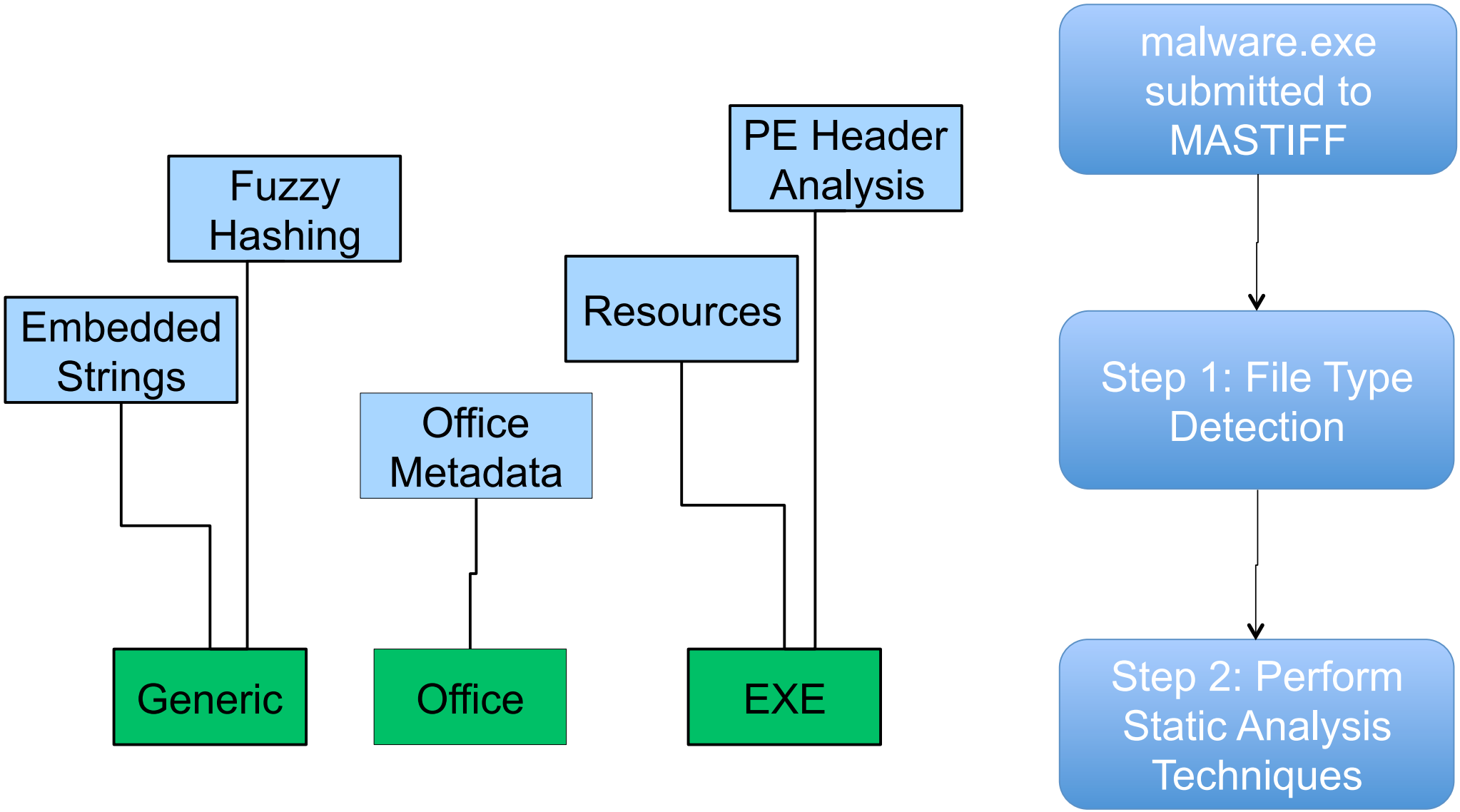


Analysis Plug-ins

Code that implements static analysis techniques.

- Associated with a specific file type
- Will only run if that type of file is being examined
- Handles its own output (text file or DB)
- Two types:
  - Technique implemented within the Python code
  - Calls a 3rd party program that performs analysis

MASTIFF Process



Available Analysis Plug-ins (1)

Generic Plug-ins	
File Info	Obtains file information (name, size) and stored in database.
Fuzzy Hashing	Generates fuzzy hash and compares to other known fuzzy hashes.
Embedded Strings	Obtains ASCII and UNICODE embedded strings.
VirusTotal	Checks file hash against VirusTotal.
Yara	Runs Yara signatures against file.

Windows EXE Plug-ins	
PE Info	Dumps information found in the PE header.
Resources	Obtains information on and extracts resources.
Digital Signature	Extracts PE digital signature.
Single-byte Strings	Obtains and single-byte strings.

Available Analysis Plug-ins (2)

PDF Plug-ins	
pdfid	Runs Didier Steven's pdfid.py on document.
pdf-parser	Uncompresses document and finds interesting objects using Didier Steven's pdf-parser.py.
PDF Metadata	Extracts metadata from document.

MS Office Plug-ins	
Office Metadata	Extracts metadata from document.
pyOLEScanner	Runs Evilcry's pyOLEScanner.py on the document.

Zip Archive Plug-ins	
ZipInfo	Extracts zip metadata and document information.
ZipExtract	Extracts zip archive contents.

Running MASTIFF – mas.py

Sets up the framework and runs the analysis.

```
mas.py -c CONFIG_FILE [FILE|DIR]
```

```
$ mas.py -c ./mastiff.conf ./VPTray.exe
[2013-02-25 14:46:52,793] [INFO] [Mastiff] : Starting analysis on ./VPTray.exe
[2013-02-25 14:46:52,794] [INFO] [Mastiff.Init_File] : Analyzing ./VPTray.exe.
[2013-02-25 14:46:52,794] [INFO] [Mastiff.Init_File] : Log Directory: ./work/log/9c3afcc11e337ca839c696677140dd21
[2013-02-25 14:46:52,870] [INFO] [Mastiff.DB.Insert] : Adding ['EXE', 'Generic']
[2013-02-25 14:46:52,915] [INFO] [Mastiff.Analysis] : File categories are ['EXE', 'Generic'].
[2013-02-25 14:46:52,918] [INFO] [Mastiff.Plugins.Resources] : Starting execution.
[2013-02-25 14:46:52,949] [INFO] [Mastiff.Plugins.Single-Byte Strings] : Starting execution.
[2013-02-25 14:46:52,980] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:52,998] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:52,999] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:53,057] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:53,096] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:53,575] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:53,575] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:53,612] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:53,614] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:54,136] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:54,137] [INFO] [Mastiff.Plugin
[2013-02-25 14:46:54,158] [INFO] [Mastiff.Analys
```

```
$ pwd
/tmp/malware/work/log/9c3afcc11e337ca839c696677140dd21
$ ls -l
total 212
-rw-r--r-- 1 tyler tyler    110 Feb 25 14:46 fuzzy.txt
-rw-r--r-- 1 tyler tyler   3280 Feb 25 14:46 mastiff.log
-rw-r--r-- 1 tyler tyler   1068 Feb 25 14:46 mastiff-run.config
-rw-r--r-- 1 tyler tyler  24381 Feb 25 14:46 peinfo-full.txt
-rw-r--r-- 1 tyler tyler   5597 Feb 25 14:46 peinfo-quick.txt
drwxr-xr-x 2 tyler tyler   4096 Feb 25 14:46 resources
-rw-r--r-- 1 tyler tyler    737 Feb 25 14:46 resources.txt
-rw-r--r-- 1 tyler tyler  40394 Feb 25 14:46 strings.txt
-rw-r--r-- 1 tyler tyler     48 Feb 25 14:46 virustotal.txt
-rw-r--r-- 1 tyler tyler 110592 Feb 25 14:46 VPTray.exe.VIR
-rw-r--r-- 1 tyler tyler   6305 Feb 25 14:46 yara.txt
```

Configuration File

Windows INI-formatted config file that sets options.

Options can be over-ridden at the CLI with the -o option.

```
# This is the configuration file for mastiff.
#
# Comments are preceded by a # or ;
#

[Dir]
# log_dir is the base directory where the logs generated will
# be placed in.
log_dir = /usr/local/mastiff/log

# plugin_dir is a list of directories plugins may be present in.
# should be comma-separated.
plugin_dir = /usr/local/mastiff/plugins, /etc/mastiff

[Misc]
# verbose = [on|off]
verbose = off

[Sqlite]
# Sqlite database options
# db_file = Name of the database file
db_file = mastiff.db

[File ID]
# trid is the location of the TrID binary
# trid_db is the location of the TrID database
trid = /usr/local/bin/trid
trid_db = /usr/local/etc/triddefs.trd

[Embedded Strings Plugin]
# Options for the Embedded Strings Plugin.
# strcmd is the path to the strings command
strcmd = /usr/bin/strings
```

Development Information

- Plug-in templates available for all plug-in types
- APIs available for all tasks within framework
- Coding standards and information in project documentation

```
import mastiff.category.generic as gen

# Change the class name and the base class
class GenSkeleton(gen.GenericCat):
  """Skeleton generic plugin code."""

  def __init__(self):
    """Initialize the plugin."""
    gen.GenericCat.__init__(self)

  def activate(self):
    """Activate the plugin."""
    gen.GenericCat.activate(self)

  def deactivate(self):
    """Deactivate the plugin."""
    gen.GenericCat.deactivate(self)

  def analyze(self, config, filename):
    """Analyze the file."""

    # sanity check to make sure we can run
    if self.is_activated == False:
      return False
    log = logging.getLogger('Mastiff.Plugins.' + self.name)
    log.info('Starting execution.')

    # Add analysis code here. data var below would be changed.

    data = "\"Generic Plugin Example Skeleton data from %s\"" % filename
    self.output_file(config.get_var('Dir','log_dir'), data)
    return True

  def output_file(self, outdir, data):
    """Print output from analysis to a file."""

    # Code to log to file goes here
```

Future Enhancements

- More category plug-ins (APK, JAR, OfficeX)
- More analysis plug-ins
- Tagging
- Output plug-ins
- Windows/OSX compatibility

Community Involvement

MASTIFF was made to be extended.

Send me your category or analysis plug-ins.

Can't code? Let me know what you want to see.

http://sourceforge.net/projects/mastiff

mastiff-project@korelogic.com


Demo

Thank you!
Questions?

http://sourceforge.net/projects/mastiff

mastiff-project@korelogic.com

Tyler Hudak
thudak@korelogic.com
@secshoggoth