

AUGUST 3-8, 2019
MANDALAY BAY / LAS VEGAS

# BLACKPHENIX

**MALWARE ANALYSIS + AUTOMATION FRAMEWORK** 

Presented by Chris Navarrete
Senior Security Researcher @ Fortiguard Labs

#### Disclaimer

• All content derives from my own opinions and does not represent my employer's views or opinions



### BLACKPHENIX – Malware Analysis & Automation Framework

- **BLACKPHENIX** is an open source malware analysis automation framework composed of services, scripts, plug-ins, and tools and is based on a Command-and-Control (C&C) architecture
- Adapted many of the well-known malware analysis tools into Python modules
  - PEiD can be seen now as: Peld("MyMalwareSample.exe")
- What this framework is not?
  - A Sandbox solution or replacement, but a complement Why a complement?
- What you can do with it?
  - Automate any GUI or console tool and implement it easily into the framework
  - Develop BPH (Python) scripts to perform a single or bundled execution of tools within the Virtual Machine, to later work on the generated data through BPH Analysis modules

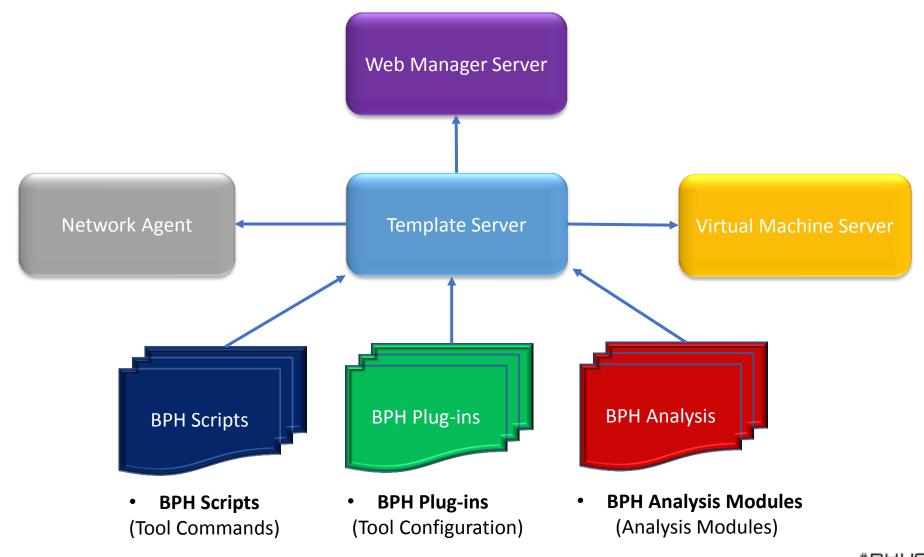


#### BLACKPHENIX – CUSTOM ANALYSIS CONDITIONALS

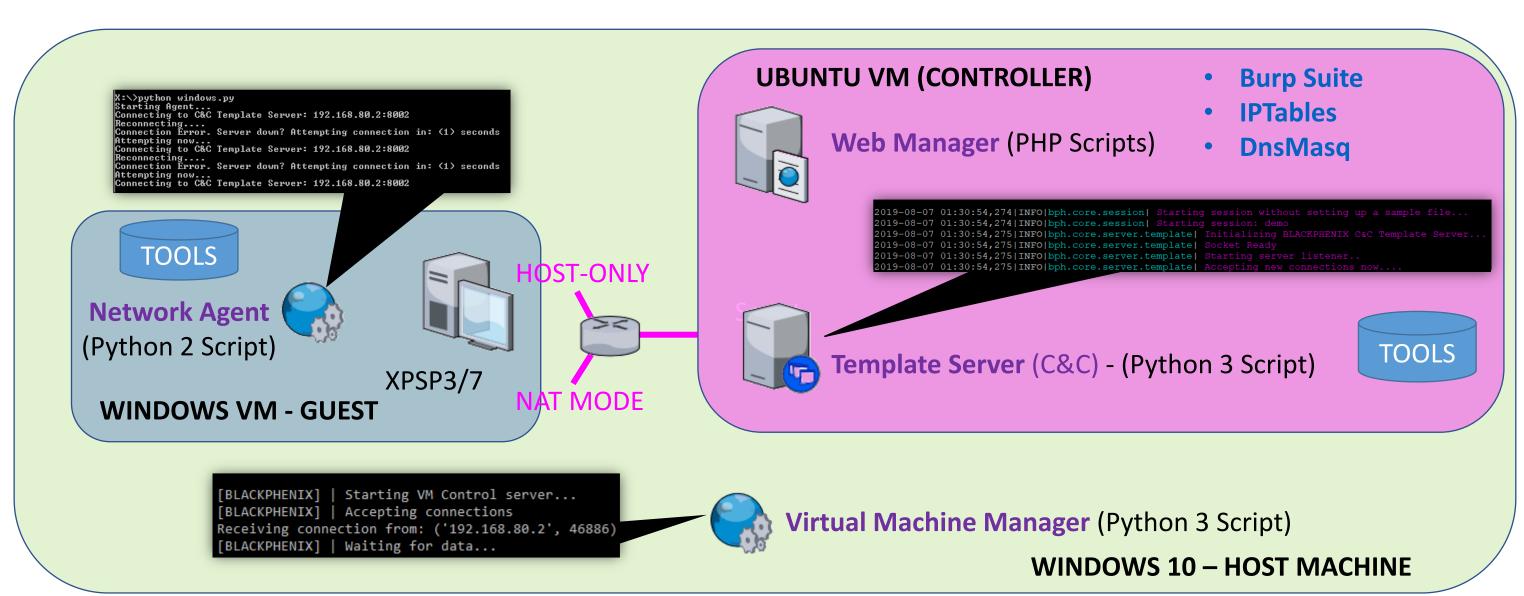
If malware sample...

- is Packed/Protected/Compressed with "EvilPacker\_v1337", then run:
  - A. Stand-Alone unpacker software (FUU-Faster-Universal-Unpacker, UPX, ...)
  - B. Stand-Alone Python unpacker Script
  - C. Debugger/Disassembler Script (OllyDbg, IDA Python, WinDBG, ...)
  - D. App's Own Scripting Engine Script (010 Editor, CFF Explorer, CheatEngine, ...)
- Connects to remote services to fingerprint its external IP, then restart and switch the
  guest VM to connect through a TOR (anonymous) communication channel
- **Downloads malicious components**, then pause its execution (before, during, or after) to generate a memory dump and perform further automated analysis

## High-Level Overview



#### Main Architecture



#### **Features**

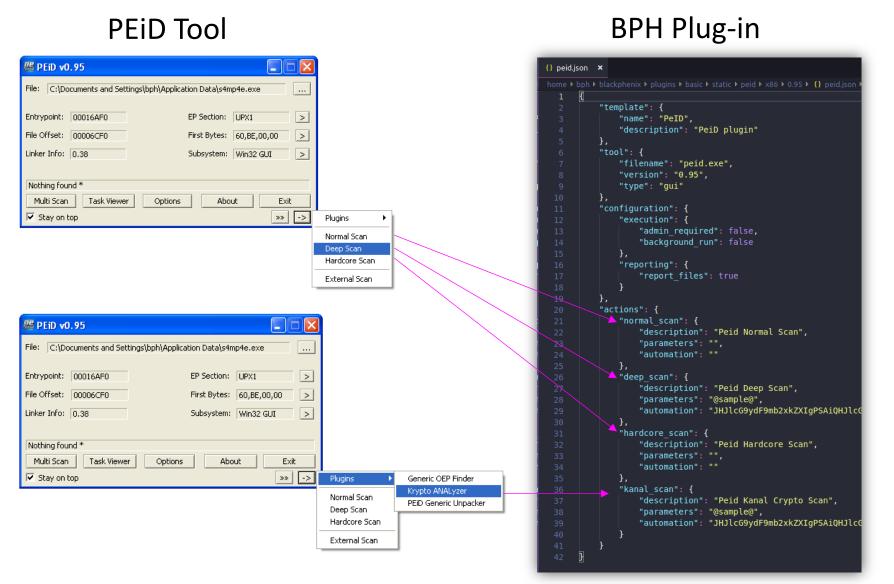
- Easy Installation & Deployment
- Python automation modules for malware analysis tools
- Virtual Machine Management
- Scripting support (BPH Scripts)
- Internet emulation
- Traffic redirection
- TOR support

```
from bph.tools.windows.peid import BphPeid as Peid
from bph.tools.windows.capturebat import BphCaptureBat as CaptureBat
from bph.tools.windows.regshot import BphRegShot as RegShot
from bph.tools.windows.procmon import BphProcMon as ProcMon
from bph.tools.windows.upx import BphUpx as Upx
from bph.tools.windows.pestudio import BphPeStudio as PeStudio
from bph.tools.windows.pd import BphPd as Pd
from bph.tools.windows.dumppe import BphDumpPE as DumpPE
from bph.tools.windows.depends import BphDepends as Depends
from bph.tools.windows.autoruns import BphAutoruns as Autoruns
from bph.tools.windows.exeinfope import BphExeInfoPe as ExeInfoPe
from bph.tools.windows.sysinspector import BphSysInspector as SysInspector
from bph.tools.windows.strings import BphStrings as Strings
from bph.tools.windows.floss import BphFloss as Floss
from bph.tools.windows.bintext import BphBinText as BinText
from bph.tools.windows.nircmd import BphNirCmd as NirCmd
from bph.tools.windows.ollydbg import BphOllyDbg as OllyDbg
from bph.tools.windows.resourcehacker import BphResourceHacker as ResourceHacker
from bph.tools.windows.xorsearch import BphXorSearch as XorSearch
from bph.tools.windows.xorstrings import BphXorStrings as XorStrings
from bph.tools.windows.networktrafficview import BphNetworkTrafficView as NetworkTrafficView
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.sample import BphSample as Sample
from bph.core.sample import BphLabFile as LabFile
from bph.core.session import BphSession as Session
from bph.core.vm import BphVmControl as VmControl
from bph.core.network.sniffer import BphSniffer as Sniffer
from bph.analysis.basic.static import BphStringAnalysis as StringAnalysis
from bph.analysis.network import BphNetworkAnalysis as NetworkAnalysis
```



# BPH Plug-In, Module & Script Structure Composition (PEiD GUI - Tool Automation)

#### BPH Plug-In, Module & Script Structure – PEiD GUI Elements



#### BPH Plug-In

```
me ▶ bph ▶ blackphenix ▶ plugins ▶ basic ▶ static ▶ peid ▶ x86 ▶ 0.95 ▶ {} peid.is
         "template": {
            "name": "PeID",
            "description": "PeiD plugin"
            "filename": "peid.exe",
            "version": "0.95",
            "type": "gui"
                "admin required": false,
                 "background run": false
            "reporting": {
                 "report files": true
        "actions": {
            "normal scan": {
                 "description": "Peid Normal Scan",
                 "parameters": "",
                 "automation": ""
             "deep scan": {
                 "description": "Peid Deep Scan",
                "parameters": "@sample@",
                "automation": "JHJlcG9ydF9mb2xkZXIgPSAiQHJlcG
             "hardcore scan": {
                "description": "Peid Hardcore Scan",
                 "parameters": "",
                 "automation": ""
                 "description": "Peid Kanal Crypto Scan",
                 "parameters": "@sample@",
                 "automation": "JHJlcG9ydF9mb2xkZXIgPSAiQHJlcG
```

#### BPH Plug-In == Tool's JSON Configuration file

- Template & Tool Meta-data information
- Configuration Execution & Reporting information
- Actions Tool's customized options
  - Description Action's information
  - Parameters Local or Remote variables (@sample@ = Sample's location inside of the VM)
  - Automation Base64 Auto-It (automation) encoded data

JHJlcG9ydF9mb2xkZXIgPSAiQHJlcG9ydF9mb2xkZXJAIg0KJGxvZ2ZpbGUgPSA
kcmVwb3J0X2ZvbGRlciAmICJccGVpZc5sb2ciDQpGaWxlRGVsZXRLKCRsb2dmaW
xlKQ0KDQpHbG9iYWwgJHBlaWQgPSBXaW5BY3RpdmF0ZSgiUEVpRCIpDQpQZWlkU
nVuQXV0b21hdGlvbigpDQpHbG9iYWwgJHQgPSBDb250cm9sR2V0VGV4dCgkcGVp
ZCwgIiIsICJbQ0xBU1M6RWRpdDsgSU5TVEFOQ0U6Ml0iKQ0KUGVpZFdyaXRlTG9
nKCkNClNlbmRLZWVwQWNOaXZLKCRwZWlkLCAiIikNClNlbmQoIiF7eH0iKQ0KV2
luQ2xvc2UoJHBlaWQpDQoNCkZ1bmMgUGVpZFdyaXRlTG9nKckNCiAgIERpclJlb
W92ZSgkcmVwb3J0X2ZvbGRlciwgMSkNCiAgIERpckNyZWF0ZSgkcmVwb3J0X2Zv
bGRlcikNCiAgICRmID0gRmlsZU9wZW4oJGxvZ2ZpbGUsIDIpDQogICBGaWXlV3J
pdGUoJGYSICR0KQ0KRW5kRnVuYw0KDQpGdW5jIFBlaWRSdW5BdXRvbWF0aW9uKC
kNCiAgIENvbnRyb2xDbGljaygkcGVpZCwgIi0+IiwgMTAyNiwgImxlZnQiKQ0KI
CAgU2VuZCgie0RPV059IikNCiAgIFNlbmQoIntET1d0fSIpDQogICBTZW5kKCJ7
RE9XTn0iKQ0KICAgU2VuZCgie0V0VEVSfSIpDQpFbmRGdW5j

```
(Untitled) * SciTE-Lite [11 of 11]
File Edit Search View Tools Options Language Buffers Help
1 first.au3 | 2 second.au3 | 3 compare.au3 | 4 aa1.au3 | 5 zz.au3 | 6 a12.au3 | 7 ee.au3 | 8 rgcomp1.au3 | 9
          $report folder = "@report folder@"
         $logfile = $report_folder & "\peid.log"
         FileDelete($logfile)
          Global &peid = WinActivate("PEiD")
         PeidRunAutomation
          Global &t = ControlGetText(&peid, "", "[CLASS:Edit; INSTANCE:2]")
         PeidWriteLog(
         SendKeepActive($peid, ""
  11
         WinClose($peid
          Func PeidWriteLog(
            DirCreate($report_folder)
  16
            $f = FileOpen($logfile, 2)
  17
            FileWrite($f, $t)
  19
  20
         Func PeidRunAutomation(
            ControlClick($peid, "->", 1026, "left")
  22
  23
            Send("(DOWN)"
  24
            Send (" (DOMN) "
  25
            Send ("{ENTER}"
```

#### **BPH Module**

```
ne 🕨 bph 🕨 blackphenix 🕨 bph 🕨 tools 🕨 windows 🕨 🍨 peid.py 🕨
   from termcolor import colored
   from bph.core.template import BphToolTemplateExecutor
   class BphPeid(BphToolTemplateExecutor):
       def init (self, target file=None, tool name='peid', arch='x86', version='0.95'):
           super(). init () ◀
           self.load tool config file(tool name, arch, version, target file=target file)
       def normal scan(self):
           self.logger.log(colored('EXECUTING NORMAL SCAN', 'yellow'))
           self.actions.action = "normal scan"
       def deep scan(self):
           self.logger.log(colored('EXECUTING DEEP SCAN', 'yellow'))
           self.actions.action = "deep scan"
       def hardcore scan(self):
           self.logger.log(colored('EXECUTING HARDCORE SCAN', 'yellow'))
           self.actions.action = "hardcore scan"
       def kanal scan(self):
           self.logger.log(colored(|EXECUTING KANAL SCAN', 'yellow'))
           self.actions.action = "kanal scan"
```

- BPH Module == Tool's Python Class
  - A Class should be created: BphMyTool()
    - This class inherits from *BphToolTemplateExecutor* class, which is a class that acts as interface between BPH plug-ins, scripts a Virtual Machine Network Agent
  - Initialization function ("\_\_init\_\_" method) is in charge of:
    - Initializing the class with tool's name, architecture, and version number
    - Calling *super()* to inherit methods and attributes from its parent class
    - Loading BPH-Plug-In JSON configuration file by calling self.load\_tool\_config\_file()
      method and tool's data as parameters
  - For each Tool's feature, a class method will be used
    - Each method must enable the action that will be called/executed
    - It modifies values inside the JSON (BPH Plug-In) configuration data
  - In BLACKPHENIX, \*all\* tools are Python objects (Classes) with its correspondent attributes!



#### **BPH Script**

```
peid.py ~/.../windows
home

# Tool imports
from bph.tools.windows.peid import BphPeid as Peid

# Core Imports
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile

session = Session(project_name='pony8')
session.start()

templateserver = TemplateServer()
templateserver.start()

sample_file = LabFile('/home/bph/blackphenix/session/pony8/launcher/s4mp4e.exe')

peid = Peid(sample_file)
peid.execute()
peid.output()

peid.output()
```

- BPH Script == Execution Instructions
  - The tool's Class should be imported
    - From bph.tools.<os>.<tool\_name> import BphMyTool as MyTool
  - The tool's object can be created as a regular Python object
    - Some tools require initialization parameters, especially the ones that work with samples or files
    - Some other tools (such as Process Monitor or CaptureBAT are not tied to a sample, but collecting data from a virtualized sample execution
  - After object's creation, then its methods can be called
    - Tools that run inside the Virtual Machine requires a call to the execute() method
    - BPH uses two main methods: output() and files()
      - Output() returns the tool's output
      - Files() returns the folder location where the collected files resides on
  - On a single BPH Scripts a user can choose a single tool or a bundle to perform different actions and use the data generated by all of them!

#### BPH Plug-In, Module & Script Structure – All together

BPH Plug-in

```
"template": {
   "name": "PeID",
   "description": "PeiD plugin"
   "filename": "peid.exe",
   "version": "0.95",
"configuration":
        "admin required": false,
        "background run": false
        "report files": true
    "normal scan": ·
        "description": "Peid Normal Scan",
        "parameters": "".
        "description": "Peid Deep Scan",
        "automation": "JHJlcG9ydF9mb2xkZXIgPSAiQHJlc
    "hardcore scan": {
        "description": "Peid Hardcore Scap",
        "parameters": "",
        "automation": ""
    "kanal scan": {
        "description": "Peid Kanal Crypto Scan",
        "parameters": "@sample@",
        "automation": "JHJlcG9ydF9mb2xkZXIgPSAiQHJlc
```

```
BPH Module
```

```
me 🕨 bph 🕨 blackphenix 🕨 bph 🕨 tools 🕨 windows 🕨 🏺 peid.py 🕨
    from termcolor import colored
    from bph.cdre.template import BphToolTemplateExecutor
    class BphPeid(BphToolTemplateExecutor):
        def init (self, target file=None, tool name='peid', arch='x86', version='0.95')
            self.load tool config file(tool_name, arch, version, target_file=target_file)
     def normal scan(self):
            self.logger.log(colored('EXECUTING NORMAL SCAN', 'yellow'))
            self.actions.action = "normal scan"
            self.logger.log(colored('EXECUTING DEEP SCAN', 'yellow'))
            self.actions.action = "deep scan"
        def hardcore scan(self):
            self.logger.log(colored('EXECUTING HARDCORE SCAN', 'yellow'))
            self.actions.action = "hardcore scan"
      __def kanal scan(self):
            self.logger.log(colored('EXECUTING KANAL SCAN', 'yellow'))
            self.actions.action = "kanal_scan"
```

#### **BPH Script**

```
peid.py ~/.../windows
home b bob b blackohenix b scripts b custom b tools b peid.py b

# Tool imports
from bph.tools.windows.peid import BphPeid as Peid

# Core Imports
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile

# session = Session(project_name='pony8')
session.start()

# templateserver = TemplateServer()
templateserver.start()

# sample_file = LabFile('/home/bph/blackphenix/session/pony8/launcher/s4mp4e.exe')

# peid = Peid(sample_file)
# peid.execute()
# peid.execute()
# peid.output()
# peid.output()
```

Only three files (one JSON configuration file and two Python scripts) are required to have a fully functional and integrated tool



# PEID DEMO



## BPH Analysis Modules – NetworkTrafficView Tool

 BPH Analysis scripts are used to consume the data collected by the tools, or in other words, the results given by the BPH scripts

1. Import network analysis – **NetworkAnalysisCsvReader** class

```
# Analysis Imports
from bph.analysis.network import BphNetworkAnalysisCsvReader as NetworkAnalysisCsvReader
```

2. Extract (ntv.fetch()) all domains from the collected CSV file

```
ntv = NetworkTrafficView()
ntv.start()
ntv.execute()

sample_exec = NirCmd(LabFile(session.launcher_abs_path))
sample_exec.configuration.execution.background_run = True
sample_exec.start_process(program='@sample@')
sample_exec.execute(delay=15)
ntv.stop()
ntv.execute()

for csv_file in ntv.files():
    ntv = NetworkAnalysisCsvReader(tool_name='networktrafficview', csv_file=csv_file)
    ntv.fetch(data_type='domains')
```

```
class BphNetworkAnalysisCsvReader(BphNetworkAnalysis):
   def init (self, tool name=None, csv file=None):
       super(). init ()
       self.tool name = tool name
       self.csv file = csv file
   def fetch(self, data type=None):
       with open(self.csv file) as csv file h:
           csv_reader = csv.reader(csv_file h, delimiter=',')
           domains = []
            for row in csv reader:
               if self.tool name == "networktrafficview":
                   if "Destination" not in str(row):
                       if "[" in str(row[3]):
                           domain = row[3].split(' ')[0]
                           if domain not in domains:
                                domains.append(domain)
           print(domains)
            return domains
```



## NetworkTrafficView + Analysis Module + Python One-liner

```
from bph.tools.windows.networktrafficview import BphNetworkTrafficView as NetworkTrafficView
from bph.tools.windows.nircmd import BphNirCmd as NirCmd
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile
from bph.analysis.network import BphNetworkAnalysisCsvReader as NetworkAnalysisCsvReader
import time
session = Session(project name='demo')
session.start()
templateserver = TemplateServer()
templateserver.start()
ntv = NetworkTrafficView()
ntv.start()
ntv.execute()
nircmd = NirCmd()
nircmd.configuration.reporting.report files = True
nircmd.start process(program=r'python -c "import urllib2; print(urllib2.urlopen(\"http://icanhazip.com\").read().strip())" > @report_folder@\\nircmd.log')
nircmd.execute(delay=5)
nircmd.output()
ntv.stop()
ntv.execute()
for csv file in ntv.files():
    ntv = NetworkAnalysisCsvReader(tool name='networktrafficview', csv file=csv file)
    ntv.fetch(data_type='domains')
```



# **UPX** Decompression

```
home ▶ bph ▶ blackphenix ▶ scripts ▶ custom ▶ tools ▶ 🏶 upx.py ▶ ...
      from bph.tools.windows.upx import BphUpx as Upx
      from bph.core.server.template import BphTemplateServer as TemplateServer
      from bph.core.session import BphSession as Session
      from bph.core.sample import BphLabFile as LabFile
      session = Session(project name='pony8')
      session.start()
      session.set_launcher(move_sample=False)
      templateserver = TemplateServer()
 10
      templateserver.start()
 12
 13
      upx = Upx(LabFile(session.launcher abs path))
 14
      upx.decompress()
      upx.execute()
      upx.output()
 16
      upx.files()
```

# TOR + Python Support

```
from bph.core.vm import BphVmControl as VmControl
from bph.core.network.sniffer import BphSniffer as Sniffer
session = Session(project name='pony8')
session.start()
vm = VmControl()
vm.set_vm('basic_static', network_id=1)
vm.start()
templateserver = TemplateServer()
templateserver.start()
time.sleep(10)
nircmd = NirCmd()
nircmd.configuration.reporting.report files = True
nircmd.start process(program=r'python -c "import urllib2; print(urllib2.urlopen(\"http://icanhazip.com\").read().strip())" > @report folder@\\nircmd.log')
nircmd.output()
templateserver.stop()
vm.stop()
vm = VmControl()
vm.set_vm('basic_static', network_id=2)
vm.start()
templateserver = TemplateServer()
templateserver.start()
time.sleep(20)
nircmd = NirCmd()
nircmd.configuration.reporting.report_files = True
nircmd.start_process(program=r'python -c "import urllib2; print(urllib2.urlopen(\"http://icanhazip.com\").read().strip())" > @report_folder@\\nircmd.log')
nircmd.execute(delay=3)
nircmd.output()
templateserver.stop()
vm.stop()
print("Done")
```



# ExeInfoPe + UPX Decompression + Imports Data

```
from bph.tools.windows.exeinfope import BphExeInfoPe as ExeInfoPe
from bph.tools.windows.upx import BphUpx as Upx
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session from bph.core.sample import BphLabFile as LabFile
session = Session(project name='pony8')
session.start()
session.set launcher(move sample=False)
templateserver = TemplateServer()
templateserver.start()
sample file = LabFile(session.launcher abs path)
exeinfope = ExeInfoPe(sample file)
exeinfope.execute()
sig match = re.search(r'(upx.*3.91.*)', "".join(exeinfope.output()), re.I)
    print("SAMPLE IS UPX PACKED => VERSION 3.91. DECOMPRESSING...")
    for symbol, function_data in sample_file.symbols(type='imports').items():
         for data in function data:
            print(data)
    upx = Upx(sample file)
    upx.execute()
    unpack result = upx.output()
    if "Unpacked 1 file." in unpack result:
        print("SAMPLE WAS UNPACKED SUCCESSFULLY...!!!11")
        files found = upx.files()
        for file found in files found:
            if "unpacked.exe" in file found:
                unpacked_file = LabFile(file_found)
                exeinfope = ExeInfoPe(unpacked file)
                exeinfope.default()
                exeinfope.execute()
                 for symbol, function data in unpacked file.symbols(type='imports').items():
                    print(symbol)
                     for data in function data:
                         print(data)
```



#### MANDALAY BAY / LAS VEGAS

## DUMPPE + WINAPI FUNCTION PARSING

```
# Tool Modules
from bph.tools.windows.dumppe import BphDumpPE as DumpPE

# Core Modules
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile

session = Session(project_name='pony8')
session.start()
session.set_launcher(move_sample=False)

templateserver = TemplateServer()
templateserver.start()

dumppe = DumpPE(LabFile(session.launcher_abs_path))
dumppe.default()
dumppe.execute()
dumppe.output()

dumppe.win_apis()
```

```
import re

# Custom imports
from bph.core.constants import *
from bph.core.template import BphToolTemplateExecutor
from bph.analysis.basic.static import BphWinApiStrings as WinApiStringAnalysis

class BphDumpPE(BphToolTemplateExecutor):

def __init__(self, target_file=None, tool_name='dumppe', arch='x86', version='2.32'):
    super().__init__()
    self.load_tool_config_file(tool_name, arch, version, target_file=target_file)

def default(self):
    print("Default")
    self.configuration.reporting.report_files = True
    self.actions.action = "default"

def win_apis(self):
    #self._search(search_type='domains')
for file in self.files():
    if "dumppe.log" in file and self.rid in file:

    a = WinApiStringAnalysis()
    a.set_log(file)
    a.search()
```

```
class BphWinApiStrings(BphStringAnalysis):

def search(self):
    self.logger.log("Searching for API function names...")

apis = ['LoadLibrary', 'GetProcAddress']

for d_string in BphStringAnalysis.strings_file.readlines():
    d_string = d_string.strip()

if len(d_string) > 5:
    search = re.search(r'([A-Z]([A-Z]*[a-z]*[A-Z]*[a-z]*[A-Z]*[a-z])[A-Za-z]*)', d_string)

if search and len(search.group()) > 5:
    string_found = search.group()

if string_found in apis:
    self.logger.log(colored("Found: {}*.format(string found), 'green'))
```

## BinText

```
from bph.tools.windows.bintext import BphBinText as BinText
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile
session = Session(project name='pony8')
session.start()
session.set launcher(move sample=False)
templateserver = TemplateServer()
templateserver.start()
bintext = BinText(LabFile(session.launcher abs path))
bintext.default()
bintext.execute(delay=3)
bintext.output()
bintext.files()
```



XOR Search **XOR Strings** PEID KANAL Scan **FLOSS** 

```
from bph.tools.windows.peid import BphPeid as Peid
from bph.tools.windows.floss import BphFloss as Floss
from bph.tools.windows.xorsearch import BphXorSearch as XorSearch
from bph.tools.windows.xorstrings import BphXorStrings as XorStrings
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile
session = Session(project name='pony8')
session.start()
session.set launcher(move sample=False)
templateserver = TemplateServer()
templateserver.start()
xorsearch = XorSearch(LabFile(session.launcher abs path))
xorsearch.search www()
xorsearch.execute()
xorsearch.output()
xorstrings = XorStrings(LabFile(session.launcher abs path))
xorstrings.search xored()
xorstrings.execute()
xorstrings.output()
peid = Peid(LabFile(session.launcher abs path))
peid.kanal scan()
peid.execute()
peid.output()
floss = Floss(LabFile(session.launcher abs path))
floss.search()
floss.execute()
floss.output()
```



# CaptureBAT

```
from bph.tools.windows.capturebat import BphCaptureBat as CaptureBat
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.sample import BphSample as Sample
from bph.core.sample import BphLabFile as LabFile
from bph.core.session import BphSession as Session
import time
session = Session(project name='pony8')
session.start()
templateserver = TemplateServer()
templateserver.start()
capturebat = CaptureBat()
capturebat.cleanup()
capturebat.execute()
capturebat.start()
capturebat.execute(delay=15)
capturebat.stop()
capturebat.execute()
capturebat.collect()
capturebat.execute()
capturebat.files()
```

# RegShot

```
# Tool Modules
     from bph.tools.windows.regshot import BphRegShot as RegShot
     # Core Modules
     from bph.core.server.template import BphTemplateServer as TemplateServer
     from bph.core.session import BphSession as Session
     from bph.core.sample import BphLabFile as LabFile
     session = Session(project name='pony8')
     session.start()
11
     templateserver = TemplateServer()
     templateserver.start()
     regshot = RegShot()
     regshot.first shot()
     regshot.execute()
     regshot.second shot()
19
     regshot.execute()
     regshot.compare()
     regshot.execute()
     regshot.files()
```

## **Process Monitor**

```
# Tool Modules
     from bph.tools.windows.procmon import BphProcMon as ProcMon
     from bph.core.server.template import BphTemplateServer as TemplateServer
     from bph.core.session import BphSession as Session
     from bph.core.sample import BphLabFile as LabFile
     session = Session(project name='pony8')
     session.start()
     templateserver = TemplateServer()
     templateserver.start()
     procmon = ProcMon()
     procmon.capture xp()
     procmon.execute(delay=10)
     procmon.terminate()
     procmon.execute(delay=15)
21
     procmon.export()
     procmon.execute(delay=10)
     procmon.files()
```

# NirCmd – OS Command (PING)

```
# Tool Modules
from bph.tools.windows.nircmd import BphNirCmd as NirCmd
# Core Modules
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
session = Session(project name='pony8')
session.start()
templateserver = TemplateServer()
templateserver.start()
nircmd = NirCmd()
nircmd.configuration.reporting.report files = True
nircmd.start process(program='ping 4.2.2.2 > @report folder@\\nircmd.log')
nircmd.execute(delay=5)
nircmd.output()
nircmd.files()
```

# OllyDbg - OllyScript

```
Tool Modules
from bph.tools.windows.ollydbg import BphOllyDbg as OllyDbg
# Core Modules
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile
session = Session(project name='pony8')
session.start()
session.set launcher(move sample=False)
templateserver = TemplateServer()
templateserver.start()
ollydbg = OllyDbg(LabFile(session.launcher abs path))
ollydbg.configuration.execution.background run = True
ollydbg.ollyscript(ollyscript file name='logapicall.osc')
ollydbg.execute()
```



## Resource Hacker

```
# Tool Modules
from bph.tools.windows.resourcehacker import BphResourceHacker as ResourceHacker
# Core Modules
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile
session = Session(project name='pony8')
session.start()
session.set launcher(move sample=False)
templateserver = TemplateServer()
templateserver.start()
reshacker = ResourceHacker(LabFile(session.launcher abs path))
reshacker.extract resources()
reshacker.execute()
reshacker.files()
```

## Autoruns

```
# Tool Modules
from bph.tools.windows.autoruns import BphAutoruns as Autoruns
# Core Modules
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile
session = Session(project_name='pony8')
session.start()
templateserver = TemplateServer()
templateserver.start()
autoruns = Autoruns()
autoruns.analysis basic()
autoruns.execute(delay=5)
autoruns.files()
```



# Malware Execution + CaptureBAT

```
from bph.tools.windows.nircmd import BphNirCmd as NirCmd
from bph.tools.windows.pd import BphPd as Pd
from bph.tools.windows.capturebat import BphCaptureBat as CaptureBat
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile
session = Session(project name='aop')
session.start()
session.set launcher(move sample=False)
templateserver = TemplateServer()
templateserver.start()
capturebat = CaptureBat()
capturebat.cleanup()
capturebat.execute()
capturebat.start()
capturebat.execute()
sample exec = NirCmd(LabFile(session.launcher abs path))
sample exec.configuration.execution.background run = False
sample exec.start process(program='@sample@')
sample exec.execute()
capturebat.stop()
capturebat.execute()
capturebat.collect()
capturebat.execute()
capturebat.files()
```



# Malware Execution + pd32

```
from bph.tools.windows.nircmd import BphNirCmd as NirCmd
from bph.tools.windows.pd import BphPd as Pd
from bph.core.server.template import BphTemplateServer as TemplateServer
from bph.core.session import BphSession as Session
from bph.core.sample import BphLabFile as LabFile
session = Session(project name='aop')
session.start()
session.set launcher(move sample=False)
templateserver = TemplateServer()
templateserver.start()
sample exec = NirCmd(LabFile(session.launcher abs path))
sample exec.configuration.execution.background run = True
sample exec.start process(program='@sample@')
sample exec.execute()
pd = Pd()
pd.dump process(process name='@sample filename@')
pd.execute(delay=5)
files found = pd.files()
for file found in files found:
    if file found.endswith('.exe'):
        dumped file = LabFile(file found)
        dumped file imports = dumped file.symbols(type='imports')
        if dumped file imports is not False:
            for symbol, function data in dumped file imports.items():
                print(symbol)
                for data in function data:
                    print(data)
       print(dumped file.abs path)
kill process = NirCmd()
kill process.kill process(program='@sample filename@')
kill process.execute()
```



# **Supported Tool Automation**

BASIC STATIC ANALYSIS	BASIC DYNAMIC ANALYSIS (BEHAVIORAL ANALYSIS)
• PeID (+KANAL)	<ul> <li>RegShot</li> </ul>
• ExeInfoPe	<ul> <li>Process Monitor</li> </ul>
• BinText	<ul> <li>Autoruns</li> </ul>
• Floss	<ul> <li>NirCmd</li> </ul>
Dependency Walker	<ul> <li>CaptureBAT</li> </ul>
• DumpPE	
• UPX	
XOR Search/Strings	
Resource Hacker	

ADVANCED DYNAMIC	ADVANCED STATIC
OllyDbg (OllyScript)	<ul> <li>*IDA Pro (IDA Python)</li> </ul>
*Cheat Engine (LUA)	

MEMORY ANALYSIS	NETWORK ANALYSIS
• Pd32/64	<ul> <li>NetworkTrafficView</li> </ul>
	• Tcpdump



#### How to Contribute

- Developing BPH Scripts or Analysis Modules and push them into the official GitHub repository
- Introducing new tools to the Framework Check the tool's licensing terms before starting!
- Fixing bugs, and/or adding new features
- If you have doubts, don't hesitate to ask! (See contact details)
- Have a cool tool in mind? I will guide you to integrate it!



# Contact

Name: Chris Navarrete

Personal Email: acid.ctrl.ak@gmail.com

BPH Email: bph\_framework@fortinet.com

Twitter: @ctrl\_ak

IRC@FREENODE: #bph-framework