< insert cover page >

Table of Contents

[PNG Fuzzing Pit 3](#_Toc312080410)

[Specification Coverage 3](#_Toc312080411)

[Step 0: Prerequisites 4](#_Toc312080412)

[Windows 4](#_Toc312080413)

[Python 2.7 : Required 4](#_Toc312080414)

[Windows Debugging Tools : Required 4](#_Toc312080415)

[!Exploitable : Strongly Recommended 5](#_Toc312080416)

[Wireshark : Recommended 5](#_Toc312080417)

[Linux 6](#_Toc312080418)

[Python 2.7 : Required 6](#_Toc312080419)

[Wireshark : Recommended 6](#_Toc312080420)

[OSX 7](#_Toc312080421)

[Python 2.7 : Required 7](#_Toc312080422)

[XCode : Strongly Recommended 7](#_Toc312080423)

[Crash Wrangler : Required 7](#_Toc312080424)

[Wireshark : Recommended 7](#_Toc312080425)

[Installing and Running Peach 8](#_Toc312080426)

[Unpacking Peach 8](#_Toc312080427)

[OSX and Linux 8](#_Toc312080428)

[Installing Required Python Packages 8](#_Toc312080429)

[Windows 8](#_Toc312080430)

[Linux and OSX 9](#_Toc312080431)

[vDebug on Linux 10](#_Toc312080432)

[Checking Peach 11](#_Toc312080433)

[Setting up the Pit 11](#_Toc312080434)

[Configuring a logging directory 11](#_Toc312080435)

[Configuring the Peach Agent 12](#_Toc312080436)

[Windows 12](#_Toc312080437)

[Fuzzing a command line target 12](#_Toc312080438)

[Fuzzing a GUI target (Windows Only) 12](#_Toc312080439)

[Linux (Command line and GUI) 13](#_Toc312080440)

[OSX (Command line and GUI) 13](#_Toc312080441)

[Supplying a Seed File 14](#_Toc312080442)

[Launch Peach 14](#_Toc312080443)

[Debugging FAQ 14](#_Toc312080444)

[Support Contact 14](#_Toc312080445)

# PNG Fuzzing Pit

The Déjà vu Security <Pit Name> provides file fuzzing capabilities that cover the

# Specification Coverage

# Step 0: Prerequisites

The following software utilities are required to be installed to use the Peach Fuzzing framework.

## Windows

### Python 2.7 : Required

It is recommended to use the latest 2.7 release from Active State which can be downloaded from Active State’s website. <http://www.activestate.com/activepython/downloads> Make sure to get the version that matches the architecture of your operating system x86 or x64.

To check the current installed version of Python run the following command.

C:\> cd \Python

C:\Python>python.exe --version

**Python 2.7.2**

C:\Python>

This command should return at least **2.7.1**

### Windows Debugging Tools : Required

Install the appropriate Windows Debugging Tools for the target platform. Make sure to match the architecture to the operating system x86 or x64. More information about downloading and installing the Debugging Tools can be found from MSDN.

<http://msdn.microsoft.com/en-us/windows/hardware/gg463009.aspx>

Windows debugging tools should be installed at one of the following paths.

%SYSTEMDRIVE%\Program Files\Debugging Tools for Windows (x64)\

%SYSTEMDRIVE%\Program Files\Debugging Tools for Windows (x86)\

%SYSTEMDRIVE%\Program Files\Debugging Tools for Windows \

%SYSTEMDRIVE%\Program Files (x86)\Debugging Tools for Windows (x86)\

%SYSTEMDRIVE%\Program Files (x86)\Debugging Tools for Windows (x64)\

%SYSTEMDRIVE%\Program Files (x86)\Debugging Tools for Windows\

%SYSTEMDRIVE%\Debugger

%SYSTEMDRIVE%\Debuggers

### !Exploitable : Strongly Recommended

“!exploitable (pronounced “bang exploitable”) is a Windows debugging extension (WinDBG) that provides automated crash analysis and security risk assessment. The tool first creates hashes to determine the uniqueness of a crash and then assigns an exploitability rating to the crash: Exploitable, Probably Exploitable, Probably Not Exploitable, or Unknown.” [1]

Download !Exploitable from <http://msecdbg.codeplex.com/>

Unzip the MSECExtensions\_1\_0\_6.zip file.

Copy the MSEC.dll from the correct architecture directory

Binaries\x64\MSEC.dll or Binaries\x86\MSEC.dll to the Windows debugger extensions folder path.

Example if you debugger is installed at

C:\Program Files\Debugging Tools for Windows (x64)\

Copy MSEC.dll to

C:\Program Files\Debugging Tools for Windows (x64)\winext\

### Wireshark : Recommended

For networking based pits it is recommended that you install Wireshark to capture network traces of crashes.

Download and install the latest stable build of Wireshark for Windows from <http://www.wireshark.org/download.html>

## Linux

### Python 2.7 : Required

To check the current installed version of Python run the following command.

peach@peach:~$ python --version

**Python 2.7.1+**

peach@peach:~$

This command should return at least **2.7.1**

If Python 2.7 is not installed use the latest 2.7 Python package that the distribution’s package manager supports.

### Wireshark : Recommended

For networking based pits it is recommended that you install Wireshark to capture network traces of crashes. Install the latest version of Wireshark supported by your package manager.

## OSX

### Python 2.7 : Required

Python 2.7 is installed with the default Snow Leopard and Lion media.

To check the current installed version of Python run the following command.

Peach:~ peach$ --version

**Python 2.7.1+**

Peach:~ peach$

This command should return at least **2.7.1**

### XCode : Required

Xcode is required for crash collection and debugging. It is recommended that the target system have XCode installed. Download and install XCode at <http://developer.apple.com/xcode/>

### Crash Wrangler : Required

Crash wrangler is a tool provided by Apple, Inc to detect the type, severity, and exploitability of an OSX program crash.

It can be downloaded from the developer <http://connect.apple.com/cgi-bin/WebObjects/MemberSite.woa/wa/getSoftware?bundleID=20390>

To install Crash Wrangler you must compile it for each ***Machine*** you want to bucket crashes on. Copying crash wrangler binaries from one machine to the other can result in misinterpreted crashes.

unzip 35497\_crashwrangler.zip

cd crashwrangler

make

cp exc\_handler /usr/local/peach

### Wireshark : Recommended

For networking based pits it is recommended that you install Wireshark to capture network traces of crashes.

Download and install the latest stable build of Wireshark for your architecture from [http://www.wireshark.org/download.htm](http://www.wireshark.org/download.html)l

# Installing and Running Peach

For maximum compatibility it is recommended that the bundled version of Peach is used.

## Unpacking Peach

Windows

Unzip the provided Peach Package into a directory of your choosing.

***C:\peach*** is recommended for Windows.

## OSX and Linux

***/usr/local/peach***is recommended for Linux.

unzip Peach.zip

mv peach /usr/local/peach

cd /usr/local/peach

chmod 755 peach.py peach.sh

# Installing Required Python Packages

## Windows

Peach comes with the required Python dependencies.

Choose the correct architecture to match the installed version of Python and run the included installer.bat file as Administrator.

C:\peach\dependencies\py2.7-win32\installer.bat

or

C:\peach\dependencies\py2.7-win64\installer.bat

To check that Peach is properly working run the following command.

C:\>C:\peach\peach.py -1 C:\peach\samples\HelloWorld.xml

] Peach 2.3.9 DEV Runtime

] Copyright (c) Michael Eddington

[\*] Performing single iteration

Warning: Run 'DefaultRun' does not have logging configured!

[\*] Starting run "DefaultRun"

[-] Test: "HelloWorldTest" (None)

[1:?:?] Element: N/A

Mutator: N/A

Hello World!

-- Completed our iteration range, exiting

[-] Test "HelloWorldTest" completed

[\*] Run "DefaultRun" completed…

## Linux and OSX

Peach comes with the required Python dependencies in source form. Run the following commands to install the following Python packages.

**To manually compile packages install python-dev or the equivalent package that contains python header files.**

cd /usr/local/peach

cd dependencies/src/4Suite-XML-1.0.2

sudo python setup.py install

cd ../cDeepCopy

sudo python setup.py install

cd ../cPeach

sudo python setup.py install

cd ..

tar –xzvf zope.interface-3.6.1.tar.gz

cd zope.interface-3.6.1

sudo python setup.py install

cd ..

tar –xjvf Twisted-10.2.0.tar.bz2

cd Twisted-10.2.0

sudo python setup.py install

cd ..

tar –xzvf psutil-0.2.0.tar.gz

cd psutil-0.2.0

sudo python setup.py install

cd ..

tar –xzvf pyasn1-0.0.13a.tar.gz

cd pyasn1-0.0.13a

sudo python setup.py install

### vDebug on Linux

To support debugging and crash collection on Linux vDebug will have to be altered and installed. Unpack vDebug in the dependencies directory.

cd /usr/local/peach/dependencies/src/

unzip vdebug-022710.zip

cd vdebug-022710

Some Linux kernel versions are out of sync with the packaged version of vDebug in Peach. Locate the sys/user.h file for your currently installed Linux kernel commonly located at /usr/include/sys/user.h.

Inspect both the vdebug-022710/vtrace/platforms/linux.py and the sys/user.h.

The sizes of variables across the following structures must match.

|  |  |
| --- | --- |
| **vtrace/platforms/linux.py** | **sys/user.h** |
| user\_regs\_i386 | user\_regs\_struct |
| user\_fpxregs | user\_fpxregs\_struct |
| USER\_i386 | User |

Example the size of ebx must be the same.

|  |  |
| --- | --- |
| Class user\_regs\_i386(Structure):  \_fields\_ = (  ("ebx", **c\_ulong**), | struct user\_regs\_struct  {  **long int** ebx; |

Example the total size of the split register ds and \_\_ds must be the same as xds.

|  |  |
| --- | --- |
| ("ds", c\_ushort),  ("\_\_ds", c\_ushort) | long int xds; |

Most of the sizes will already match, this is a spot check to confirm the target Linux kernel is not too far out of sync with vDebug. Once the sizes and structures are confirmed to be the same install vDebug with the following command.

sudo python setup.py install

### Checking Peach

To check that Peach is properly working run the following command.

peach@peach:~$ cd /usr/local/peach

peach@peach:/usr/local/peach $ ./peach.sh -1 samples/HelloWorld.xml

] Peach 2.3.9 DEV Runtime

] Copyright (c) Michael Eddington

[\*] Performing single iteration

Warning: Run 'DefaultRun' does not have logging configured!

[\*] Starting run "DefaultRun"

[-] Test: "HelloWorldTest" (None)

[1:?:?] Element: N/A

Mutator: N/A

Hello World!

-- Completed our iteration range, exiting

[-] Test "HelloWorldTest" completed

[\*] Run "DefaultRun" completed…

# Setting up the Pit

To custom tune the pit to your target and environment a few steps must be taken. First open the pit <insertpitname.xml> in a text editor.

## Configuring a logging directory

Logging must be configured to capture fuzzing data and crash information. By default Peach will log to a local directory called “logs”.

To change the default logger to use another directory:

1. Go to the RUN SECTION of the pit.
2. Change the value=”.\logs\” to the desired directory path. If on Linux or OS change the “\”s to “/”

<Run name=”DefaultRun”>

<Logger class="logger.Filesystem">

<Param name="path" **value=".\logs\"**/>

</Logger>

</Run>

## Configuring the Peach Agent

To run programs and detect crashes an Agent configuration must be in place.

Alter the default Agent field to update the configuration to the proper debugger for your operating system.

### Windows

### Fuzzing a command line target

Configure the Agent

1. Go to the AGENT SECTION of the pit.
2. To fuzz a Windows command line target remove the **<!-- Windows** line and the **-->**. Leave all other agents commented out.
3. Update the “CommandLine” value attribute and assign the full path of the target executable where **C:\target.exe** currently is.
4. Update the “Executable” path value **target.exe**.
5. The Agent configuration section should look like this with your values filled in for target.

<Agent name="Agent">

<Monitor class="debugger.WindowsDebugEngine">

<Param name="CommandLine" value="**C:\path\to\target.exe** c:\peach\fuzzed.png"/>

<Param name="StartOnCall" value="FuzzMe"/>

</Monitor>

<Monitor class="process.PageHeap">

<Param name="Executable" value="**target.exe**"/>

</Monitor>

### Fuzzing a GUI target (Windows Only)

1. Go to the TEST SECTION of the pit.
2. Comment out the Command Line Launcher Publisher. By adding <!—and --> around the Publisher tag.
3. Remove the comment markers for the GUI Launcher section.
4. Change **GUI Window Name** to the title of the application window at launch.
5. The Test section should look like this with your values filled in for GUI Window Name.

<!-- Command Line Launcher: Win/Lin/OSX -->

**<!--**

<Publisher name="launch" class="process.DebuggerLauncher"/> **-->**

<!-- GUI Launcher: WINDOWS ONLY -->

<Publisher name="launch" class="process.DebuggerLauncherGui">

<Param name="WindowName" value="**GUI Window Name**" />

</Publisher>

### Linux (Command line and GUI)

To fuzz a Linux target either command line or GUI update the agent configuration as follows.

1. Go to the AGENT SECTION and find the Linux agent config.
2. Remove the <!—Linux and the -->. Leave all other agents commented out.
3. Update the “CommandLine” value attribute “/bin/target” and assign the full path of the target executable.
4. Optional: Change the location attribute if the agent is on a different port or host.
5. The Agent section should now look like the following.

<Agent name="Agent" location=”http://**127.0.0.1:9000**”>

<Monitor class="debugger.UnixDebugger">

<Param name="Command" value="**/bin/target**" />

<Param name="Params" value="fuzzed.png" />

<Param name="StartOnCall" value="FuzzMe" />

</Monitor>

</Agent>

### OSX (Command line and GUI)

To fuzz an OSX target either command line or GUI update the agent configuration as follows.

1. Go to the AGENT SECTION and find the Linux agent config.
2. Remove the <!—OSX and the -->. Leave all other agents commented out.
3. Update the “Command” value attribute “/Applications/Target/TargetName” and assign the full path of the target executable.
4. Optional: Change the location attribute if the agent is on a different port or host.
5. The Agent section should now look like the following.

<Agent name="Agent" location=”http://127.0.0.1:9000”>

<Monitor class="osx.CrashWrangler">

<Param name="Command" value="**/Applications/Target/TargetName**" />

<Param name="Arguments" value="fuzzed.png" />

<Param name="ExecHandler" value="./exc\_handler" />

<Param name="StartOnCall" value="FuzzMe" />

</Monitor>

## Supplying a Seed File

To help exercise code paths and assign default values the pit cracks a seed file’s values into the DataModel. The seed file must be a valid and intact representation of the target file format, this helps in both testing and debugging your configuration.

1. Download or create a seed file. Copy that file local to where Peach is being run.
2. Go to the STATE MODEL SECTION of the Pit. s
3. Change the fileName parameter to the name of the SeedFile that was downloaded.

<Action name="WriteFileFromSample" type="output" publisher="writer">

<DataModel name="Png" ref="Png" />

**<Data name="data" fileName="SeedFile.png" />**

</Action>

## Launch Peach

### Windows

To test that everything is working correctly launch peach with the -1 option. The Windows version of Peach will automatically launch the Peach Agent for you.

C:\peach\peach.bat -1 --new PitName.xml

If this correctly launches the target program and loads a file run peach.

C:\peach\peach.bat --new PitName.xml

### Linux and OSX

To test that everything is working correctly launch Peach with the -1 option. The Linux and OSX versions of Peach require that you manually launch the Agent.

Launch Peach in Agent mode in a shell.

cd /usr/local/peach

./peach.sh –a

Launch Peach in a different shell.

cd /usr/local/peach

./peach.sh –1 –new PitName.xml

## Support Contact

For limited email support contact support@dejavusecurity.com

Please provide the following.

Pit Name:

De