PAIMEIdiff BETA 1

Introduction

PAIMEIdiff is an open-source binary differ. It can be used to identify changes in patched binaries. Its other uses could include identifying families of malware. PAIMEIdiff is similar to Sabre-Security BinDiff (http://www.sabre-security.com) except PAIMEIdiff is free and saber-security uses some proprietary diffing algorithms. PAIMEIdiff is set apart from BinDiff in the fact that PAIMEIdiff can be configured on every matching/diffing level. It is the authors hope that the RE community will contribute patches and fixes to help develop PAIMEIdiff move from beta to stable and into a more refined tool.

Thanks to: Pedram, Halvar, Ero

Usage

To use PAIMEIdiff the user first must have pida files generated by pida_dump.py. Once the files are generated the user will need to load the files into PAIMEIdiff. To do this click the Load button the user will know the files are loaded when the user sees the module added to the module pane. Be sure to load the modules in different panes.

The user will then have to configure PAIMEIdiff for diffing. There are three phases to be configured, matching on the function level, matching on the basic block level, and diffing on the function level. The way the configuration works is the user selects the algorithm and order that the algorithms are applied. To select an algorithm double click on the left side that says supported methods. This will add the algorithm to the right side in the order it will be applied. To remove an algorithm selected just double click on the right side. To further configure PAIMEIdiff the user can specify insignificant basic block and functions these definitions will be applied to basic blocks and functions any basic block and function that matches the criteria, will be ignored. This speeds up matching and therefore diffing greatly. The defaults are already set and should be used. There are two check

boxes one will tell PAIMEIdiff to use the insignificant settings this will speed up matching. The other checkbox will tell PAIMEIdiff to keep applying the algorithms in a loop until there are no more matches (changes) in the list of functions.
Once the user has setup their preferred matching profile the user can save it to disk. This makes it so the user doesn't have to go through the process every time.

Now to actually diff the files the user clicks Execute. This will execute the matching and diffing engines. The results will be shown in the two tabs. The matched tab will contain all the functions that are matched and any matched function that is different will be marked in red. The unmatched tab will contain functions that have not been matched.
Dealing with the matched tab first. Once a function is matched, the user has many options. Right clicking on that function will allow the user to:

- View the matched/diff function – this will bring up a dialog box that will display the two functions side by side any unmatched basic block will be marked in read. If the user is curious about the values of the basic block i.e. the instruction count, SPP value etc the user can right click and select view basic block stats or view function stats.
- Unmatch Function – this will remove the function and its corresponding match from the match tab and move it to the unmatched tab.
- Unmark as Different – if a function is falsely marked as different the user can change that.
- Mark as Different – if a function is not marked as different but should have been the user can change that as well.

With the unmatched tab the user has two options when right clicking on unmatched functions they can manually match them or view the function. If the user chooses to manually match the function they select the function on the left and right and then select manual match PAIMEIdiff will take the two selected functions and add them to the matched tab.
If the user chooses view function PAIMEIdiff will allow the user to see the disassembly of the function so that they might better figure out what function should be matched to it.

After the diffing process is complete the user can export the results to html. To do this they merely select export and export the files to the directory of their choice.

Important Note:
The diffing phase applies one diffing algorithm at a time when all the functions are either different or matched it will stop. Which ever diffing algorithm is applied first will probably be the only algorithm to diff. The author is aware of this issue and is working on it. Another caveat if the functions being diffed had all their basic blocks matched then no diffing is done, and if the functions being diffed have the same spp or smart md5 values then no diffing algorithm is applied.

The matching/diffing modules:

To ease the creation of methods to be used in the matching diffing phases PAIMEIdiff allows authors to simply drop files into the DiffModules directory. Once a file is dropped in the directory, it is added to the internal list and PAIMEIdiff displays it in the configuration.

To create a matching module the author must have an attributes dictionary. In the attributes dictionary the author must have Match, Diff, Level defined see below example. Another required value is module_name the rest of the variables are optional.

There are five functions used to register match/diff handlers. They are:
register_match_function – this function registers the method as being able match functions
register_match_basic_block – this function registers the method as being able to match basic blocks
register_diff_function – this function registers the method as being able to diff functions
register_module – this function registers the module.

To write a matching/diffing module the author must call register_module. After that the author can then call any of the other register_* functions.

The registered functions will receive the following parameters:
    def match_function_by_smart_md5(self, function_a, function_b):
This function receives the pida function passed to it. To access extended information the user can do function_a.ext[“PAIMEIDiffFunction”].smart_md5 this will access the smart_md5 value of the function.
    def match_basic_block_by_smart_md5(self, bb_a, bb_b):
This function receives the pida basic block passed to it. To access extended information the user can do bb_a.ext[“PAIMEIDiffBasicBlock”].smart_md5 this will access the smart_md5 value of the basic block.

Important Note:
Throughout the code the user may see references to basic block diffing this was an original feature that was removed. It will be re-implemented and some of the skeleton code for the feature was left in however it is never called.

```
self.attributes = {}                    # initialize attributes

        self.attributes["Match"] = 1            # Match attribute set to 1 tells the main program 
we can be used to match
        self.attributes["Diff"] = 1             # Diff  attribute set to 1 tells the main program we 
can be used to diff
        self.attributes["Level"] = FUNCTION_LEVEL | BASIC_BLOCK_LEVEL   # these 
flags indicated we can diff/match both functions and basic blocks
        self.parent = parent                    # set up the parent
        self.module_name = "Smart_MD5"                # give the module a name
        self.author      = "Peter Silberman"    # author name
        self.description = "Smart MD5  module implements an algorithm that tokenizes the
instructions to create a smart signature."
        self.date        = "09/08/06"
        self.homepage    = "http://www.openrce.org"
        self.contact     = "peter.silberman@gmail.com"
        self.accuracy    = ACCURACY_HIGH

        self.parent.register_match_function(    self.match_function_by_smart_md5,    self )
# register a function matching routine
        self.parent.register_match_basic_block( self.match_basic_block_by_smart_md5,
self )   # register a basic block matching routine
        self.parent.register_diff_function(     self.diff_function_by_smart_md5,     self )   #
register a function diffing routine

        self.parent.register_module(self)
```

Algorithms:

A description of each algorithm supplied by default in PAIMEIdiff (the original ideas are credited but every algorithm was implemented by Peter Silberman and in some cases ideas were modified):

- Arg Var
    - Diffing and Matching
    - Function and Basic Block Level
    - If it is function level Arg Var will compare the number of arguments and local variables between two functions. If it is basic block level it will see how many times a variable or argument is accessed within the basic block and try to find another basic block that matches.
- Name
    - Matching
    - Function Level
    - Uses symbols to match the names of functions together. Generally 100% accuracy but can only be used successfully on certain types of binaries.
- NECI (Node Edge Call Instruction Count)
    - Idea: Halvar Flake
    - Diffing and Matching
    - Function and Basic Block Level
    - If it is function level it uses the node edge call instruction count to match and diff. If it is basic block level it uses the edge call instruction count (ECI) to match and diff.
- SPP (Small Prime Product)
    - Idea: Halvar Flake

    - Diffing and Matching
    - Function and Basic Block Level
    - Assigns a prime number to every instruction and multiplies them together to come up with an integer signature. Very accurate.
- Smart MD5
    - Idea: Pedram Amini
    - Diffing and Matching
    - Function and Basic Block Level
    - Each mnemonic string is put in a list. The list is ordered alphabetically all j** are assigned either jmp_unsigned or jmp_signed call, ret, jmp are all assigned jmp. This is very accurate.
- CRC
    - Diffing and Matching
    - Function and Basic Block Level
    - A CRC signature at both the function and basic block level is taken. This is very accurate.
- API
    - Diffing and Matching
    - Function and Basic Block Level
    - At the function level a string is created representing all API calls made within the function. At the basic block level a string is created made up of the API calls within a basic block.
- Constants
    - Diffing and Matching
    - Function and Basic Block Level
    - At the function level a string is created representing all constants within a function. At the basic block level a string is created made up of constants within a basic block.
- Stack Frame
    - Diffing and Matching
    - Function Level
    - The size of the stack frame is used as a signature.
- Size
    - Diffing and Matching
    - Function and Basic Block Level
    - The size of the function in bytes is used at the function level. The size of the basic block in bytes is used at the basic block level.