AutoDiffer - AutoDiff + Web-UI + agents

Lib

Repositories addresses

AutoDiff-Web : https://bitbucket.org/icewall/autodiff-web

AutoDiff-Agent : https://bitbucket.org/icewall/autodiff-agent

Motivation

Recently I was working on AutoDiff script integration with Ryan's auto-downloader.
Generally integration was not hard to do but during the tests I discovered couple problems on which I started doing deeper research and later started codding AutoDiffer.

Problems

==========
1.1 Problem with AutoIt when more than one instance of IDA/BinDiff is executed.
Because AutoIt scripts in significant places based on window title launched two instances of BinDiff window can cause sending wrong sequences to wrong window in wrong moment.
Of course, we don't wanna limit ourselfs to one instances of BinDiff per time frame because for some files it can take a while and we want to finish process of bindiff as fast as is possible

1.2 Total automation
Parsing sites for each product can be problematic + for time to time it's a big probability that they gonna change their DOM structure and parser will need to be updated. Another IMO unsolvable problem is e.g for MS patches. In a lot of cases there is couple CVS assigned to one ms-patch, but we are interested in only one particular CVE which is related only with one specific version of product like, e.g : CVE XX-XXX for IE 8 for win XP ONLY.

1.3 "Dynamic file collection " - installation of patch is needed because there is no package version (e.g proper msi,msp,etc) which can be unpacked
For that kind of product like e.g flash there is need to create system which will be uninstalling product from system (or run "naked" VM), install old product version, collect PROPER files to diff, uninstall,install new version.....
I have already that script written for Flash.

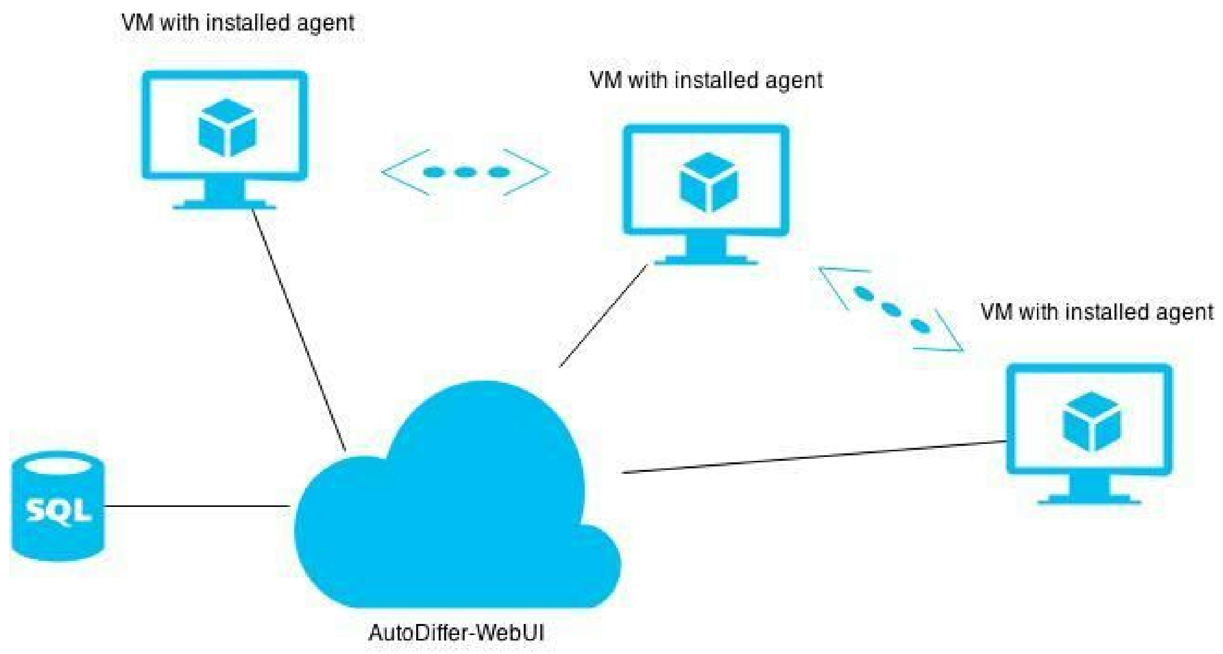
1.4 Finding significant files for diffing + finding proper pairs in old and new version.
Sometimes update package contains a lot of files e.g html, .dat which can be eliminated with simple filterring (I implemented proper system for that kind of filtering, it's configurable for each product), but real problem is when two package, old and new one contains files which names change in such way that automatic "searching for pairs" algorithm was not able to find. In that case manual intervention is needed.

Solution

========

I started working on solutions for above problems and it gonna be handle by web application which will be interface for "auto|semi-downloader" agents.

Architecture



Elements of architecture

1. AutoDiffer-WebUI
2. Database
3. Agent

How does it work ?

Operator/User find interesting for him patch to analysis, but before he starts analysis he deploys agent files to one of VMs, configure agent config file [config.py] and after that activate agent [execute agent.py]. Currently he gonna have one active agent to disposition. After this procedures, he is able to add patch to analysis via **Add** tab. Filling properly all fields in form, especially these ones pointing url with new and old patch he is able to add this patch to analysis. Pointed agent in this form received order to diff pointed packages. Thanks to specifying type of product to diff agent will launch proper module to handle entire process:

```
        self.__modules = {"any": CAny(),
                          "flash": CFlash(),
                          "IE": CAny()
                          }
        #additional initialization
        self.__modules["IE"].pairFinder.addSpecifiedFiles(["mshtml.dll"])
```

When diffing process ends, agent sends collected results to Web panel and there it is available on diffed tab for particular patch.

Functionality

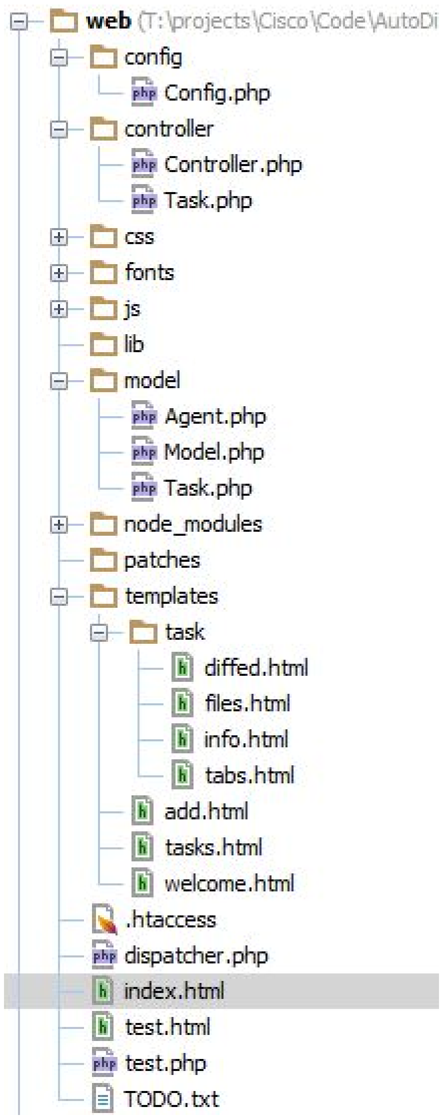
AutoDiffer-WebUI

General functionality

Using web-ui user is able to:

- add new patch for diffing
- choose manually files from patch for diffing
- monitor diffing process (observe in runtime log from diffing)
- control diffing process (stop, pause, remove,etc)
- get information about amount of changed functions in diffed files (BinDiff results and also AutoDiff ones)
- statistics, all kind of statistics

Components



```
web (T:\projects\Cisco\Code\AutoDi
    config
        Config.php
    controller
        Controller.php
        Task.php
    css
    fonts
    js
    lib
    model
        Agent.php
        Model.php
        Task.php
    node_modules
    patches
    templates
        task
            diffed.html
            files.html
            info.html
            tabs.html
        add.html
        tasks.html
        welcome.html
    .htaccess
    dispatcher.php
    index.html
    test.html
    test.php
    TODO.txt
```

Backend

PHP - MV pattern

MySQL

Sqlite - for storing information's about files contained in specified patch

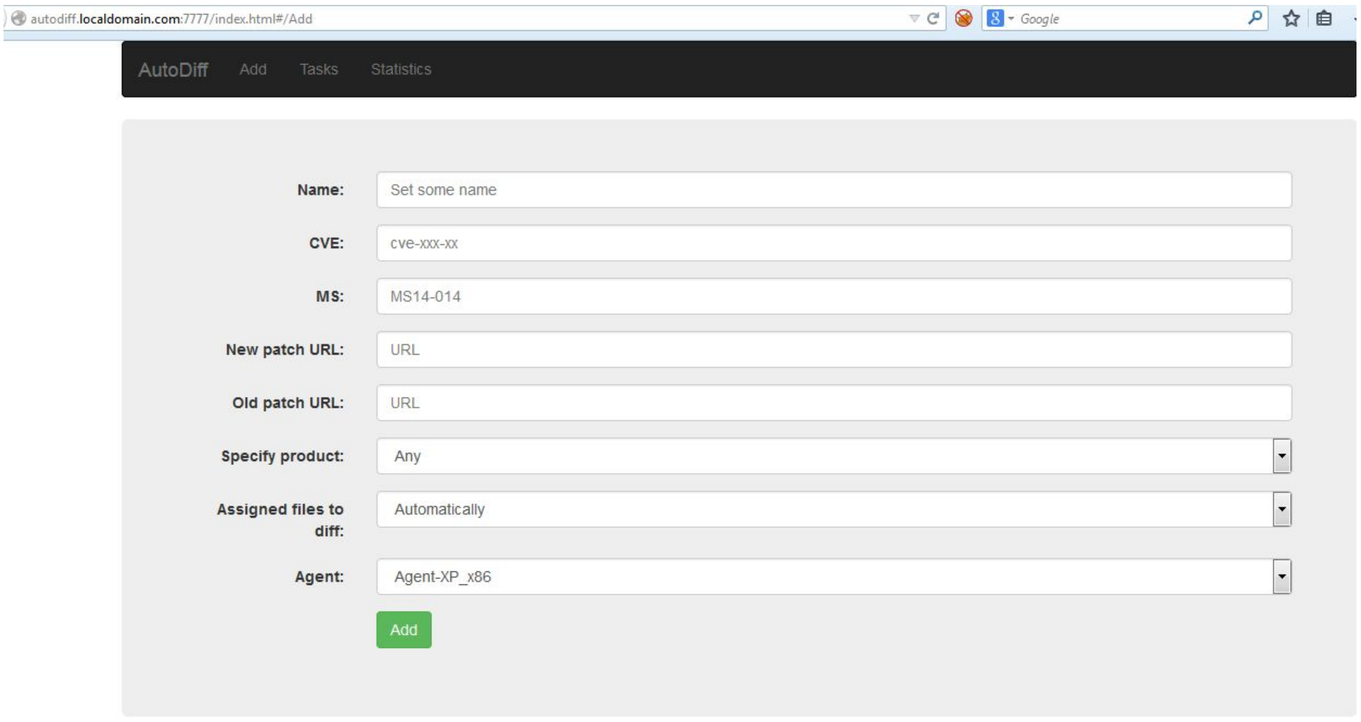
Frontend

AngularJS

Angular-UI (ui-bootstrap)

Angular-UI-Router

1. Add patch for diffing


autodiff.localdomain.com:7777/index.html#/Add

Name:	Set some name
CVE:	cve-xxx-xx
MS:	MS14-014
New patch URL:	URL
Old patch URL:	URL
Specify product:	Any
Assigned files to diff:	Automatically
Agent:	Agent-XP_x86

Here u need to specify couple things like:

- url to new version, url to old version, product* (auto|manual), files which should be diffed from this product(auto|manual),agent which will handle this analysis (agent means script runned on some VM | solution similar to cuckoo sandbox).

URLs new/old : in this point we handle manually this automatic process of parsing update web sites for different products which how I presented above is hard to do full-automaticly in some cases. This process is definitely not bottleneck of entire diffing process, so easily one person can one time in month collect manually all interesting for us urls to patches and put them into AutoDiff system in proper manner.

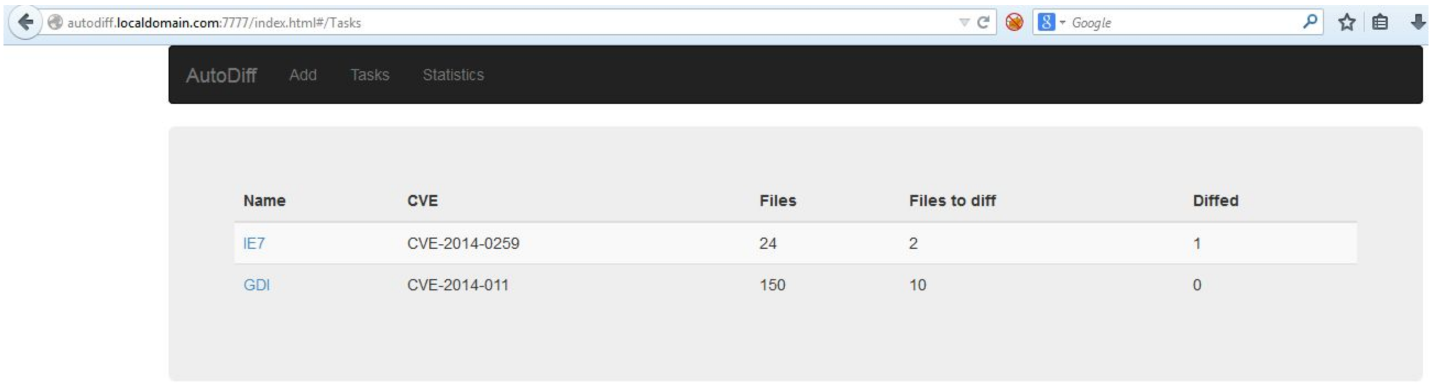
product: specifying product name like e.g Flash we give more specific information to agent how to handle package to diff. With product set to Flash, agent will launch module which specialize in Flash package diffing (has already pre-configured file list to diff, knows that should install old ver, collect files, uninst,inst new on,etc).

assigned files: maybe it's some new product to bindiff and we don't have any specialized agent + we know from some sources that package contains a lot of files and we will be interested only in few particular ones. Setting up this option on manual will give us possibility after extracting files from package to choose which files should be bindiffed.

manual: if automatic system for searching pairs for bindiffing, won't be able to find ones it gonna stop analysis and ask user for manual intervention of pointing which particular file should be diffed with which one.

agent: only available agents (scripts on different VMs) and free to work.

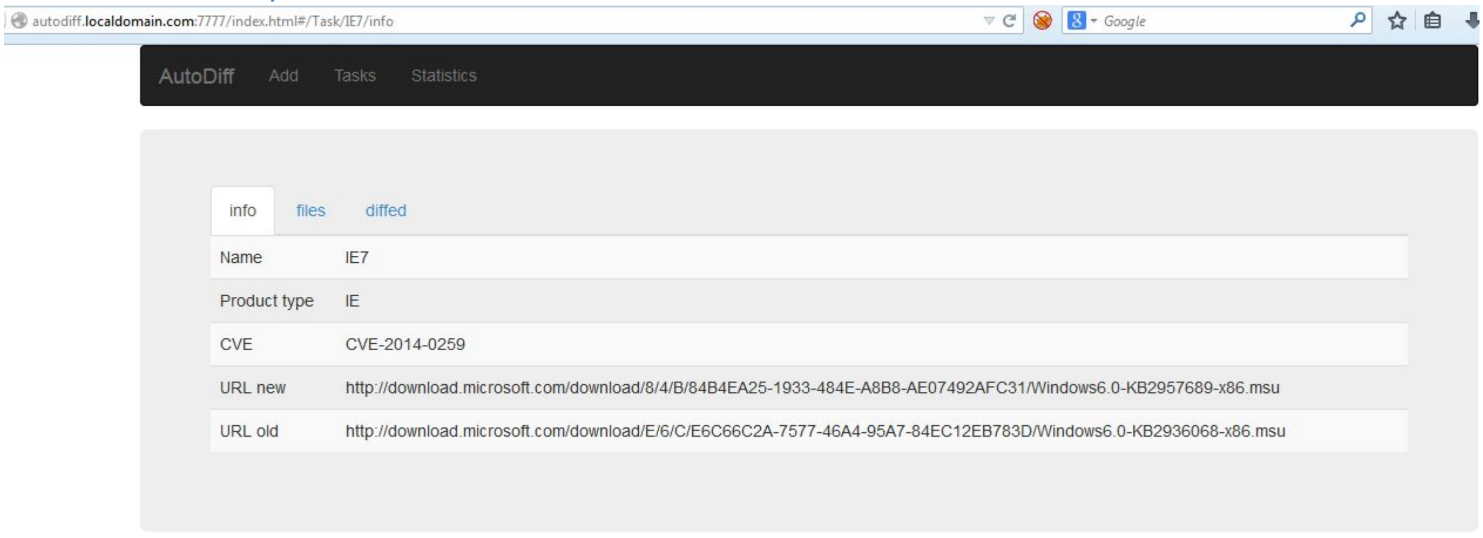
2. Tasks



Name	CVE	Files	Files to diff	Diffed
IE7	CVE-2014-0259	24	2	1
GDI	CVE-2014-011	150	10	0

Tasks tab presents information's about patches added to system which should be diffed or are already diffed. At the beginning of diffing process we can expect there only informations about:
Files - how many files contains this package
Files to diff - amount of files which **PairFinder** module found worthy/important to diff
Diffed - amount of already diffed files

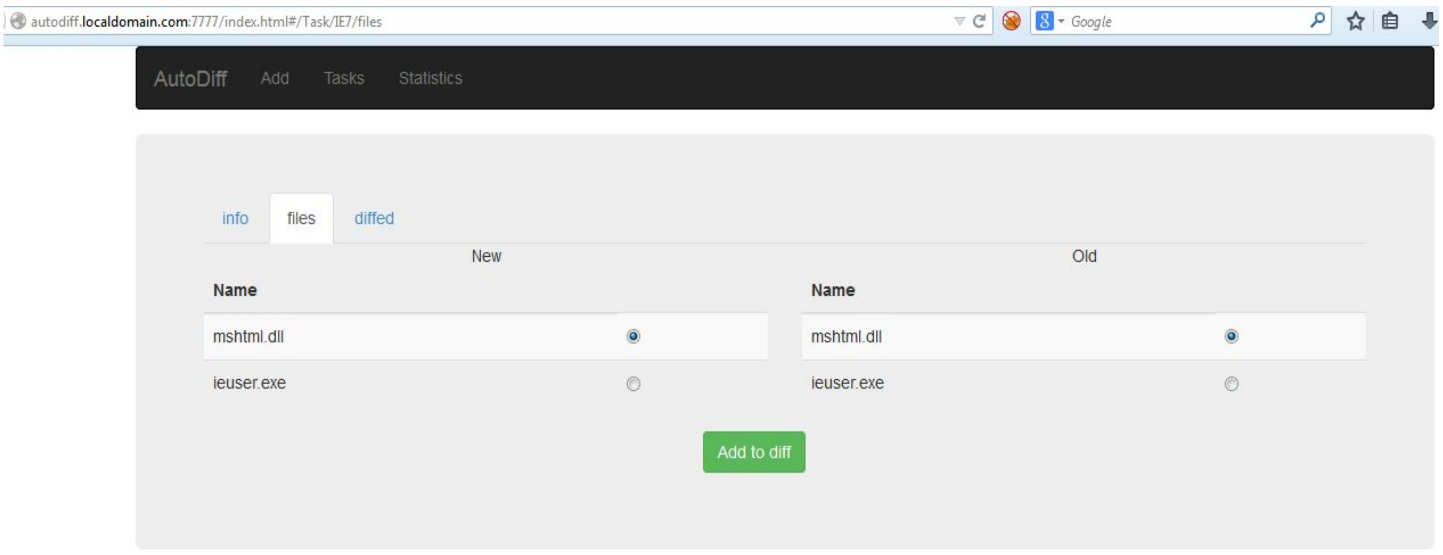
2.1 Tasks / info



Name	IE7
Product type	IE
CVE	CVE-2014-0259
URL new	http://download.microsoft.com/download/8/4/B/84B4EA25-1933-484E-A8B8-AE07492AFC31/Windows6.0-KB2957689-x86.msu
URL old	http://download.microsoft.com/download/E/6/C/E6C66C2A-7577-46A4-95A7-84EC12EB783D/Windows6.0-KB2936068-x86.msu

General informations about patch.

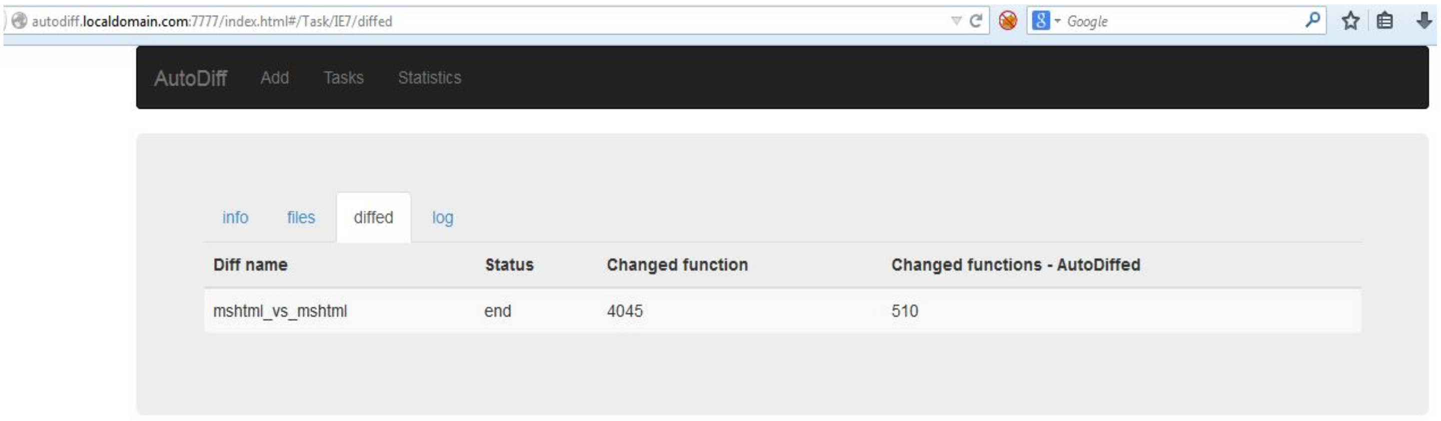
2.2 Tasks / files


autodiff.localdomain.com:7777/index.html#/Task/IE7/files

New		Old	
Name		Name	
mshtml.dll	●	mshtml.dll	●
ieuser.exe	○	ieuser.exe	○

List of files found by **PairFinder** worth diffing. You can choose manually files from this list for diffing process.

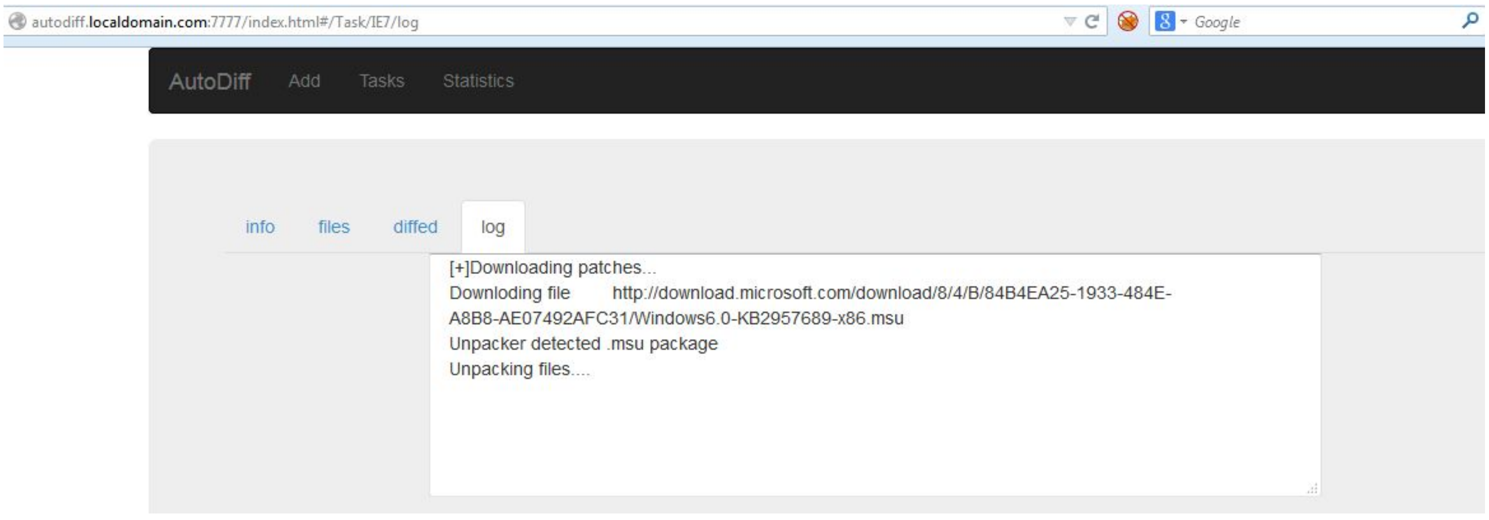
2.3 Tasks / diffed


autodiff.localdomain.com:7777/index.html#/Task/IE7/diffed

Diff name	Status	Changed function	Changed functions - AutoDiffed
mshtml_vs_mshtml	end	4045	510

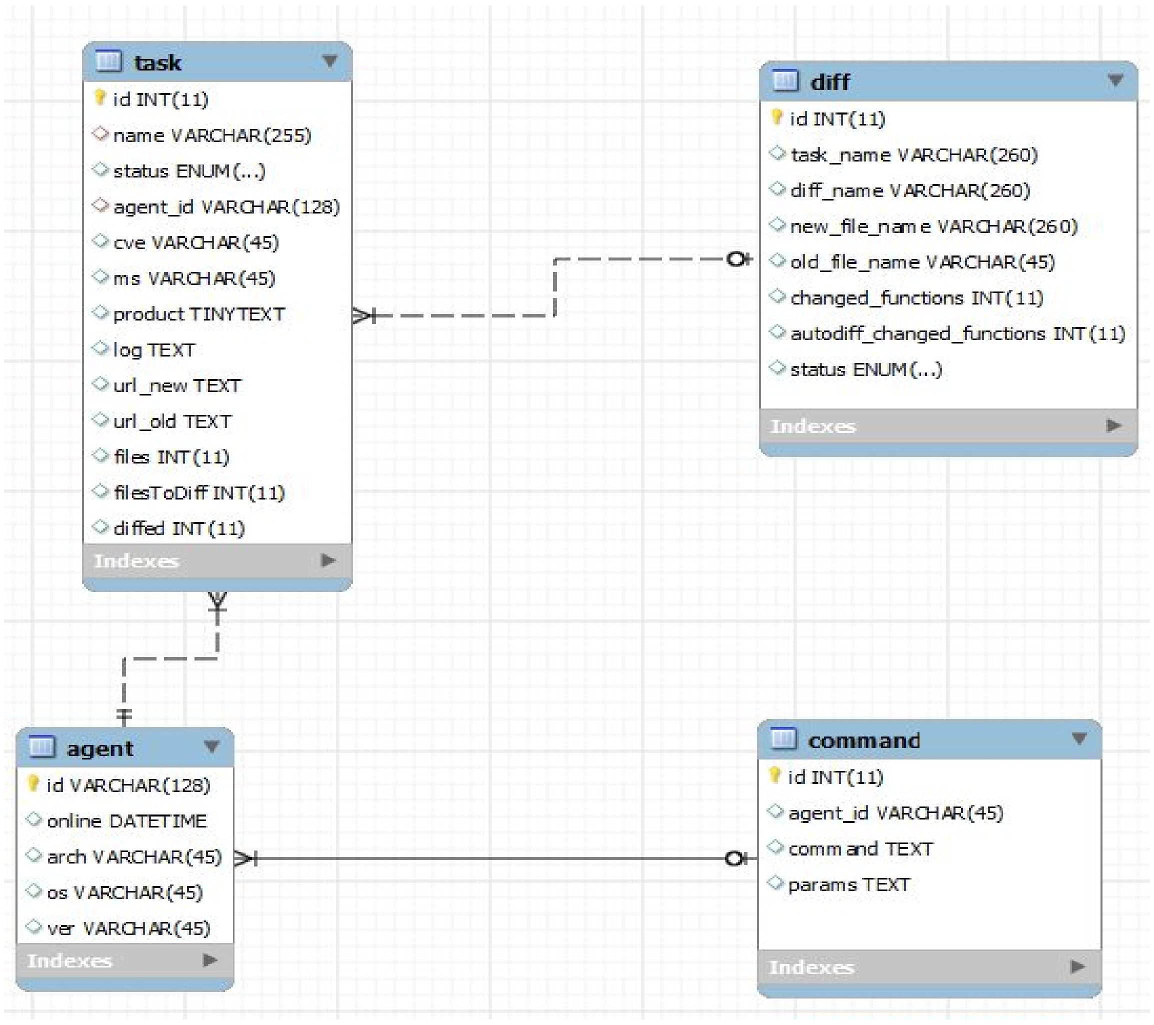
Here we can see current status of diffed process, which can be one of :
ENUM('progress','stop','end','manual')
Summary informations about how many functions BinDiff detected as changed - "**Changed functions**" and also amount of functions left to analysis after AutoDiff analysis "**Change functions - AutoDiffed**".

2.4 Tasks / log



```
[+]Downloading patches...
Downloding file      http://download.microsoft.com/download/8/4/B/84B4EA25-1933-484E-
A8B8-AE07492AFC31/Windows6.0-KB2957689-x86.msu
Unpacker detected .msu package
Unpacking files....
```

Database



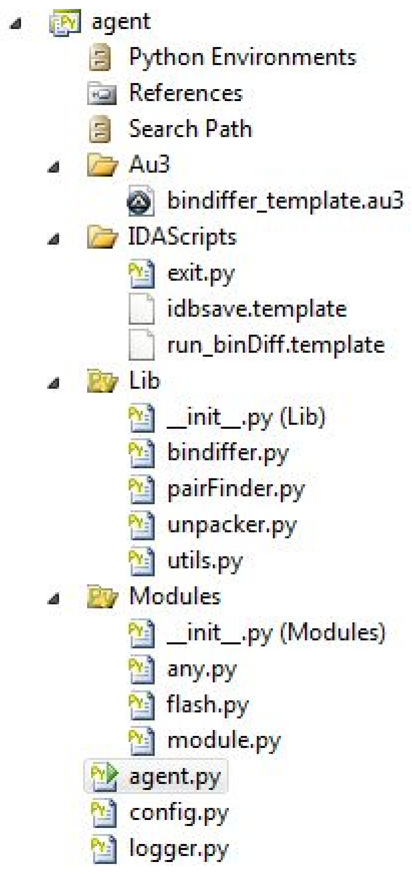
Agent

General functionality:

Agent is an Python script with proper modules and libraries responsible for all kind of things:
- receiving and executing commands from Web-UI
- patch downloading
- creating proper directory structure to keep all files related with patch analysis in clear order
- unpacking patch packages (.msu, .msi,.exe,...)
- painding pairs of files for diffing, based on their name, pattern, extensions,...
- executing IDA,BinDiff,AutoDiff script and collect results
- send logs from diffing process and all kinds of results to Web-UI

It's installation is easy and simple, u just need to copy agent files to chose VM,
set proper values in config.py file and execute major file which is agent.py.
agent.py gonna connect automaticly to Web-UI and waits for command to execute.

Components



```
agent
    Python Environments
    References
    Search Path
    Au3
        bindiffer_template.au3
    IDAScripts
        exit.py
        idbsave.template
        run_binDiff.template
    Lib
        __init__.py (Lib)
        bindiffer.py
        pairFinder.py
        unpacker.py
        utils.py
    Modules
        __init__.py (Modules)
        any.py
        flash.py
        module.py
    agent.py
    config.py
    logger.py
```

Modules

Depends on product type, there are specialized modules which handle process of unpacking and file collection for further diffing. For some unstandard patch package extraction like in example of flash were we need to install proper flash version to collect files, there are specialized modules for it - **flash.py** module.

Any.py module handle all sort of products because difference between handling their files is only in names/amount of files which we want to diff and eventually package type(msu,msi) , but generally downloading/unpacking/file collection routine is the same. Module of finding worth diffing files for specified product in described below.

Lib

PairFinder - module responsible of finding pairs of files for further diffing.
There is couple parameters which can be set in this module:
setting specified files - you can specify what kind of file names PairFinder should search for.
Usefull e.g in IE or flash example where we are interested in diffing only some specified files.
Examples of usage:

Example for IE:
```
PairFinder.addSpecifiedFiles(["mshtml.dll",])
```

and general filter by extension:
```
PairFinder.addSpecifiedExtensions([".exe",".dll"])
```

```
PairFinder = CPairFinder()
n = r"C:\flash\new_flash"
o = r"c:\flash\old_flash"

PairFinder.addSpecifiedFiles(["^NPSWF32_","^FlashUtil32_"])
PairFinder.collectFiles(n,o)
pairs = PairFinder.getPairs()
```

In above example **PairFinder** gonna find us only pair of files which names contain specified pattern (RegEx support).

Example of output for Flash:
```
Old files:
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\old_flash\FlashInstall.log
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\old_flash\flashplayer.xpt
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\old_flash\FlashPlayerUpdateService.exe
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\old_flash\FlashUtil32_14_0_0_145_Plugin.exe
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\old_flash\install.log
c:\Documents and Settings\virtual\My Documents\Downloads\flash\old_flash\mms.cfg
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\old_flash\NPSWF32_14_0_0_145.dll
c:\Documents and Settings\virtual\My Documents\Downloads\flash\old_flash\plugin.vch

New files:
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\new_flash\FlashInstall.log
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\new_flash\flashplayer.xpt
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\new_flash\FlashPlayerUpdateService.exe
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\new_flash\FlashUtil32_14_0_0_125_Plugin.exe
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\new_flash\install.log
c:\Documents and Settings\virtual\My Documents\Downloads\flash\new_flash\mms.cfg
```

```
c:\Documents and Settings\virtual\My
Documents\Downloads\flash\new_flash\NPSWF32_14_0_0_125.dll
c:\Documents and Settings\virtual\My Documents\Downloads\flash\new_flash\plugin.vch
There is new pair:
Old : FlashPlayerUpdateService.exe
New : FlashPlayerUpdateService.exe
There is new pair:
Old : FlashUtil32_14_0_0_145_Plugin.exe
New : FlashUtil32_14_0_0_125_Plugin.exe
There is new pair:
Old : NPSWF32_14_0_0_145.dll
New : NPSWF32_14_0_0_125.dll
```