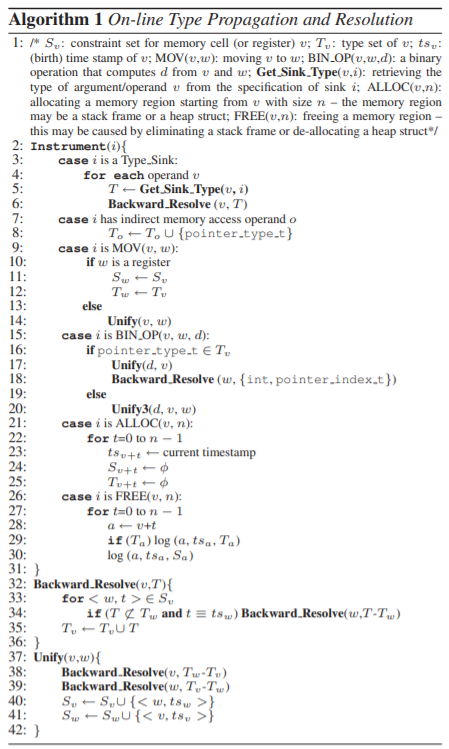
**REWARDS :- getting the information from the binary file about the data-typs and other data structures**

**Its designed according to follows :-**

1. Type Sinks :- information from where all the types & schematics are executed ( like syscalls , std lib , type revealing inst). Monitoring the system call invocations ( from the Eax register as mentioned in the intel arch , check the base ptrs and stack information mentioned in the accumulator instruction for execution based on the logic. Thus watching the following pattern of calls with the conditional calls gives a fairey infrastructure of the actual implementation. Std lib calls and structures , ADT and defined types are example.
2. Type revealing instructions :- string moving instruction or type deref are the example of code fetching the value . they are the main functions for propagation of the instructions and thus explains the functioning of the given structure.

Online type proposition and resolution algorithm with the given algorithm :-



Here we will execute the binary , for all the type sinks and standard types , we will be timestamping them and then based on the type of the function , we will accordingly return the corresponding function in the conversion mapped according to IA. If there is indirect call of the code to the memory register , we will be then propagating the result or else we will get the information. For finding recursively the type of the given function , variable which has be referenced multiple times , we will be using the Back.

**Then comes the main part :- tracking the Mapping address for getting the information of overall program binary implementation and also the detacting of the pointer types..**

Thus the use case include :- getting the ip addresses of the server and client IP address from the code analysis .

Trying to ascertain the dead code for getting the

MALWARE REINGEERING ASSIGNMENTS FROM REV ENGG 101 .

PORTABLE EXECUTION HEADER :- WINDOWS has the portable execution header . its of following parts :- dos header , stub , memory address in stack , program image , DLL’s and the thread environment and the most improtatn :- process env block :- information of the loaded mods and process.

X86 commands , registers and flags :- getting the obfuscation from the encryption , packing and other regex and substitution techniques . also the compression is done by the help of the compression software modules as follows :-

Compression

Combining the compressed data with decompression code into a single executable

* Runtime packers
* Self extractive archives

List of packers

|  |  |
| --- | --- |
| * [Themida](http://www.oreans.com/themida.php) * [Armadillo](http://www.siliconrealms.com/armadillo.php) * [ASPack](http://www.aspack.com/aspack.html) * [ASPR (ASProtect)](http://www.aspack.com/asprotect32.html) * [BoxedApp Packer](http://www.boxedapp.com/boxedapppacker) * [CExe](http://www.scottlu.com/Content/CExe.html) * [dotBundle](http://www.dotbundle.com/) * [Enigma Protector](http://www.enigmaprotector.com/) * [EXE Bundle](http://www.webtoolmaster.com/exebundle.htm) * [EXE Stealth](http://www.webtoolmaster.com/exestealth.htm) * [eXPressor](http://www.cgsoftlabs.ro/express.html) * [FSG](http://xtreeme.prv.pl/) | * [kkrunchy](http://www.farbrausch.de/~fg/kkrunchy/) * [MEW](https://web.archive.org/web/20070831063728/http:/northfox.uw.hu/index.php?lang=eng&id=dev) * [MPRESS](http://www.matcode.com/mpress.htm) * [Obsidium](http://www.obsidium.de/) * [PESpin](http://pespin.w.interia.pl/) * [Petite](http://www.un4seen.com/petite) * [RLPack Basic](http://www.softpedia.com/get/Programming/Packers-Crypters-Protectors/RLPack-Basic-Edition.shtml) * [Smart Packer Pro](http://www.smartpacker.nl/) * [Themida](http://www.oreans.com/themida.php) * [UPX](https://upx.github.io/) * [VMProtect](http://vmpsoft.com/products/vmprotect) * [XComp/XPack](http://soft-lab.de/JoKo) |