Cppcheck Defects Detection Automation for the Athena Full Scan

SHARMAZANASHVILI Alexander
Georgian Technical University

TODUA Luka
Georgian Technical University

▪ Cppcheck Scan

- Cppcheck automation is Finished and Source codes are placed at Coverity Server aibuild002.cern.ch under /build/cppcheck/ directory.
- To access the automation tool user must have access on Coverity server
- The access on Coverity server is controlled via an egroup:

  https://e-groups.cern.ch/e-groups/Egroup.do?egroupId=220386
- To Startup the full automation process, user needs valid Kerberos ticket for Jira ticket creation and Developers permission at ATLASSQ Jira group. only thing left is to run one bash shell file (auto.sh).

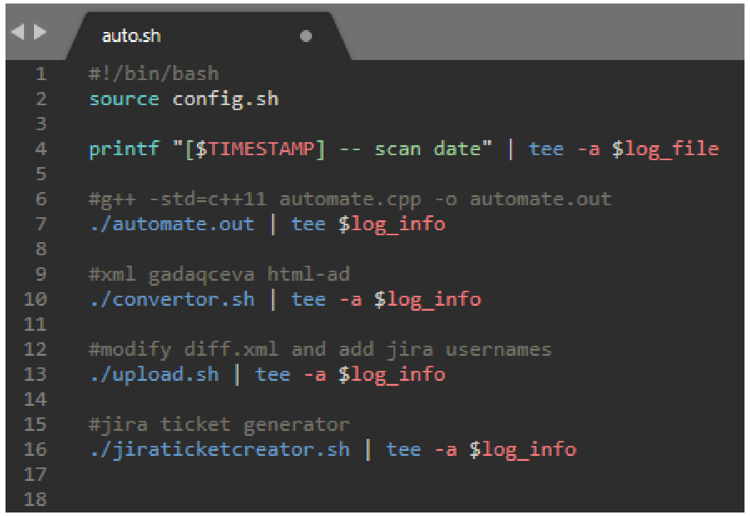
▪ Cppcheck Automation

Running auto.sh is capable of:

1. Cloning/Pulling Athena repository
2. Scanning Athena repository with Cppcheck. Generating Defects list in XML format
3. Finding new Defects. Comparison of past week Defects list and current Defects list
4. Searching Authors, their Emails and MR date for each Defect with Git command
5. Finding Defect Authors username in Cern phonebook for Jira Tickets with Ldapsearch command
6. Creating Jira Tickets and assigning to Defect Authors automatically
7. Generating Statistical data such as Overall Defects, Fixed Defects, Overall Fixed Defects in XML format
8. Converting XML files to HTML tables
9. Uploading HTML tables to Cppcheck Web page:
   http://cppcheck-list.web.cern.ch/cppcheck-list/

▪ Automation Files

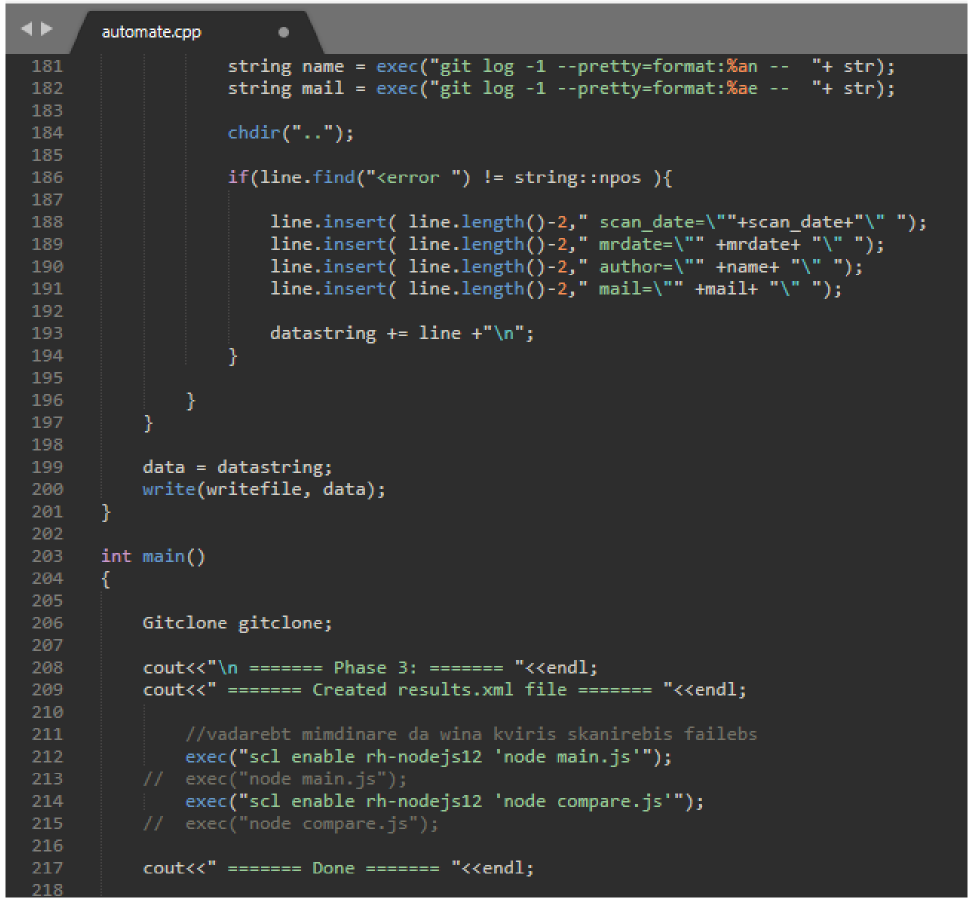
auto.sh shell file is the main file which runs automation
Steps, such as generating defects xml file, converting it into
.html table, getting defect author usernames for jira and
Creating Jira tickets for each defect.



auto.sh

```bash
#!/bin/bash
source config.sh

printf "[$TIMESTAMP] -- scan date" | tee -a $log_file

#g++ -std=c++11 automate.cpp -o automate.out
./automate.out | tee $log_info

#xml gadaqceva html-ad
./convertor.sh | tee -a $log_info

#modify diff.xml and add jira usernames
./upload.sh | tee -a $log_info

#jira ticket generator
./jiraticketcreator.sh | tee -a $log_info

```

This is automate.cpp file. Here we clone or pull Athena repository
From Gitlab, scan it with Cppcheck check all definitions enabled,
Then run nodejs based defects filter function for getting new defects and
with git log we get authors name, email, MR date for each defect

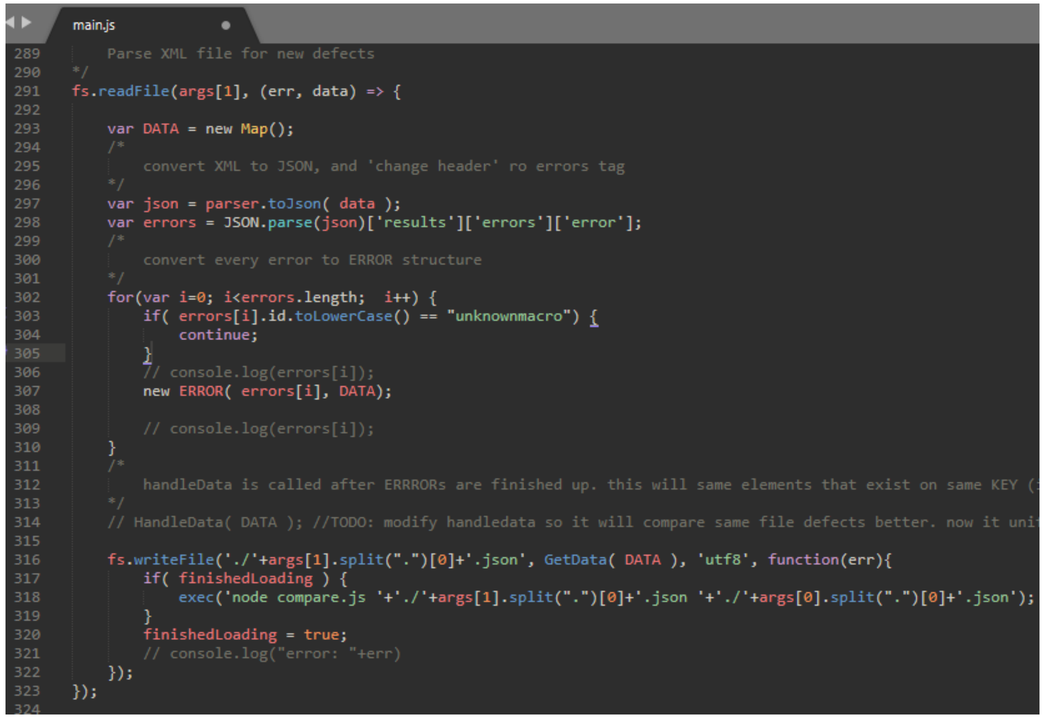


automate.cpp

```cpp
                string name = exec("git log -1 --pretty=format:%an --  "+ str);
                string mail = exec("git log -1 --pretty=format:%ae --  "+ str);

                chdir("..");

                if(line.find("<error ") != string::npos ){

                    line.insert( line.length()-2," scan_date=\""+scan_date+"\" ");
                    line.insert( line.length()-2," mrdate=\"" +mrdate+ "\" ");
                    line.insert( line.length()-2," author=\"" +name+ "\" ");
                    line.insert( line.length()-2," mail=\"" +mail+ "\" ");

                    datastring += line +"\n";
                }

            }
        }

        data = datastring;
        write(writefile, data);
    }

    int main()
    {

        Gitclone gitclone;

        cout<<"\n ======= Phase 3: ======= "<<endl;
        cout<<" ======= Created results.xml file ======= "<<endl;

            //vadarebt mimdinare da wina kviris skanirebis failebs
            exec("scl enable rh-nodejs12 'node main.js'");
        //  exec("node main.js");
            exec("scl enable rh-nodejs12 'node compare.js'");
        //  exec("node compare.js");

        cout<<" ======= Done ======= "<<endl;

```

Automation Files

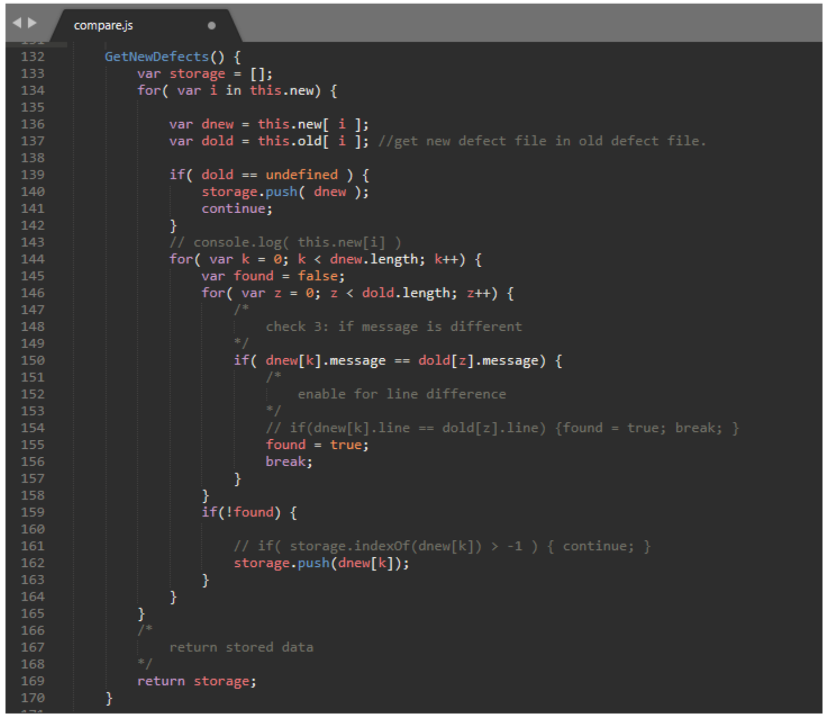
This is main.js file, part of the defects filter function, from Cppcheck generated xml defects we remove cppcheck “UnknownMacro” defects, Because they are cppcheck scanning errors. Cppcheck isn’t able to find All definition of the macro. Then from the rest of the defects we get file Path, defect message, line and modify it into MultiMap Data Structure and write in Json file



main.js

```
    Parse XML file for new defects
*/
fs.readFile(args[1], (err, data) => {

    var DATA = new Map();
    /*
        convert XML to JSON, and 'change header' ro errors tag
    */
    var json = parser.toJson( data );
    var errors = JSON.parse(json)['results']['errors']['error'];
    /*
        convert every error to ERROR structure
    */
    for(var i=0; i<errors.length;  i++) {
        if( errors[i].id.toLowerCase() == "unknownmacro") {
            continue;
        }
        // console.log(errors[i]);
        new ERROR( errors[i], DATA);

        // console.log(errors[i]);
    }
    /*
        handleData is called after ERRRORs are finished up. this will same elements that exist on same KEY (
    */
    // HandleData( DATA ); //TODO: modify handledata so it will compare same file defects better. now it uni

    fs.writeFile('./'+args[1].split(".")[0]+'.json', GetData( DATA ), 'utf8', function(err){
        if( finishedLoading ) {
            exec('node compare.js '+'./'+args[1].split(".")[0]+'.json '+'./'+args[0].split(".")[0]+'.json');
        }
        finishedLoading = true;
        // console.log("error: "+err)
    });
});
```

This is compare.js file, second part of the defects filter function, here we have Map to Map defect comparison. where Map key value is File path. If file paths are matched they compare Map values of each other. Map values are defect message and line. If defect message is different this means that we found a new defect and we save it into new json File. This happens for every defect in json files generated from main.js. The same way we also generate cppcheck statistical data such as overall defects, fixed defects and overall fixed defects.

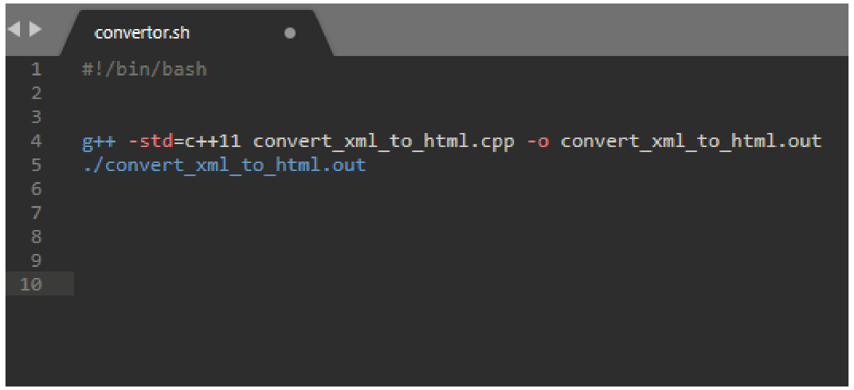


compare.js

```
    GetNewDefects() {
        var storage = [];
        for( var i in this.new) {

            var dnew = this.new[ i ];
            var dold = this.old[ i ]; //get new defect file in old defect file.

            if( dold == undefined ) {
                storage.push( dnew );
                continue;
            }
            // console.log( this.new[i] )
            for( var k = 0; k < dnew.length; k++) {
                var found = false;
                for( var z = 0; z < dold.length; z++) {
                    /*
                        check 3: if message is different
                    */
                    if( dnew[k].message == dold[z].message) {
                        /*
                            enable for line difference
                        */
                        // if(dnew[k].line == dold[z].line) {found = true; break; }
                        found = true;
                        break;
                    }
                }
                if(!found) {

                    // if( storage.indexOf(dnew[k]) > -1 ) { continue; }
                    storage.push(dnew[k]);
                }
            }
        }
        /*
            return stored data
        */
        return storage;
    }
```

Automation Files

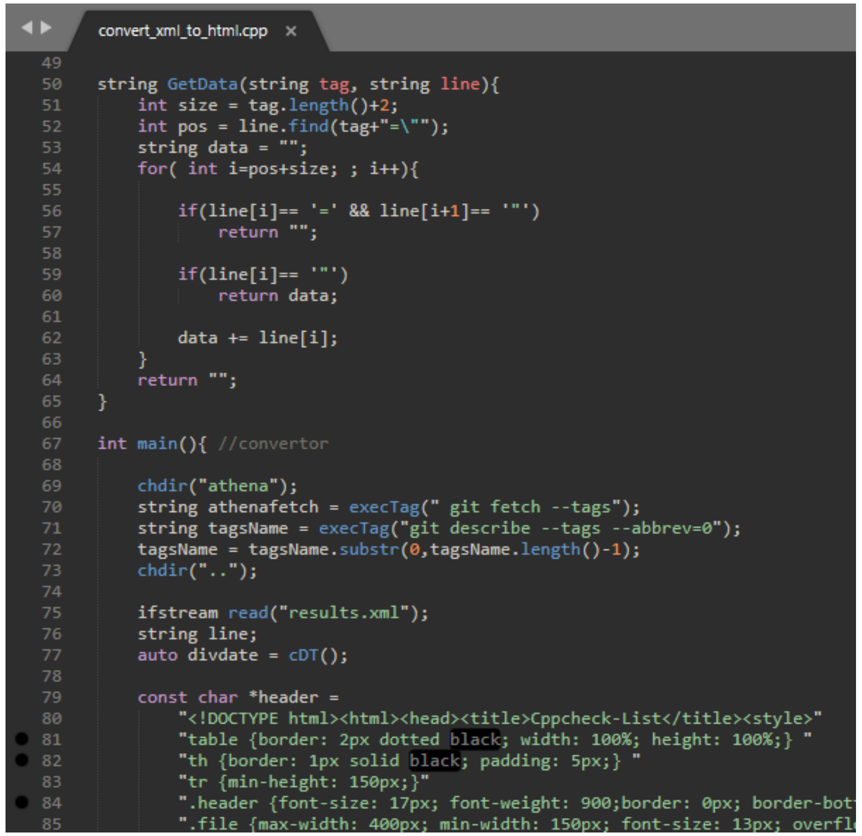
This is convertor.sh shell file. Which manages converting xml file to html tables



convertor.sh

```bash
#!/bin/bash


g++ -std=c++11 convert_xml_to_html.cpp -o convert_xml_to_html.out
./convert_xml_to_html.out




```

This is convert_xml_to_html.cpp file,
With fstream we are reading new defects xml file,
Getting data such as file path, defect message,
authors, emails And creating html tables for
new defects, overall defects, fixed defects, overall fixed
defects

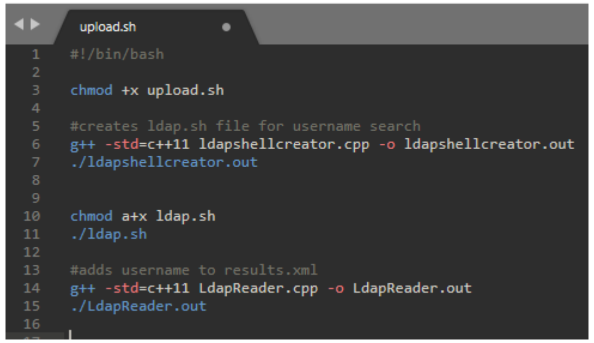


convert_xml_to_html.cpp

```cpp

string GetData(string tag, string line){
    int size = tag.length()+2;
    int pos = line.find(tag+"=\"");
    string data = "";
    for( int i=pos+size; ; i++){

        if(line[i]== '=' && line[i+1]== '"')
            return "";

        if(line[i]== '"')
            return data;

        data += line[i];
    }
    return "";
}

int main(){ //convertor

    chdir("athena");
    string athenafetch = execTag(" git fetch --tags");
    string tagsName = execTag("git describe --tags --abbrev=0");
    tagsName = tagsName.substr(0,tagsName.length()-1);
    chdir("..");

    ifstream read("results.xml");
    string line;
    auto divdate = cDT();

    const char *header =
        "<!DOCTYPE html><html><head><title>Cppcheck-List</title><style>"
        "table {border: 2px dotted black; width: 100%; height: 100%;} "
        "th {border: 1px solid black; padding: 5px;} "
        "tr {min-height: 150px;}"
        ".header {font-size: 17px; font-weight: 900;border: 0px; border-bot
        ".file {max-width: 400px; min-width: 150px; font-size: 13px; overfl
```

Automation Files

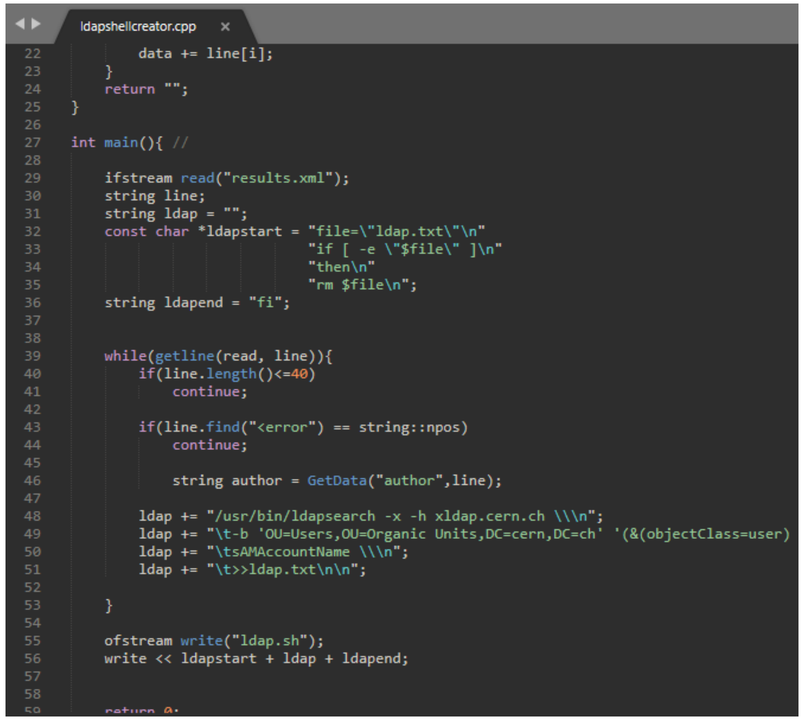
Upload.sh shell file, runs ldapshellcreator.cpp, Ldap.sh and LdapReader.cpp file. Processing all this files are Needed for automatic assigning of Jira tickets



upload.sh

```
#!/bin/bash

chmod +x upload.sh

#creates ldap.sh file for username search
g++ -std=c++11 ldapshellcreator.cpp -o ldapshellcreator.out
./ldapshellcreator.out


chmod a+x ldap.sh
./ldap.sh

#adds username to results.xml
g++ -std=c++11 LdapReader.cpp -o LdapReader.out
./LdapReader.out

```

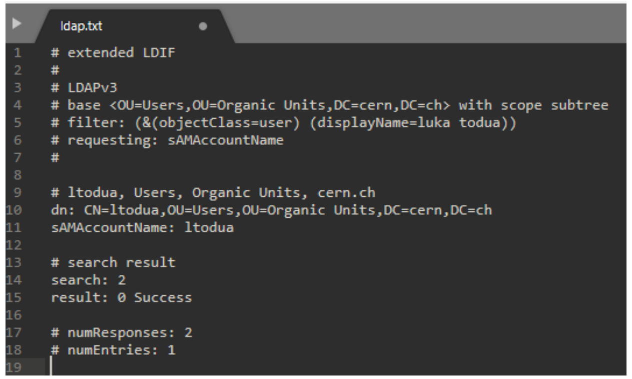
ldapshellcreator.cpp creates ldap.sh shell file. Full of ldapsearch command For each defect author. With help of Ldapsearch command we search for authors usernames by their name in CERN phonebook.



ldapshellcreator.cpp

```
            data += line[i];
        }
        return "";
}

int main(){ //

        ifstream read("results.xml");
        string line;
        string ldap = "";
        const char *ldapstart = "file=\"ldap.txt\"\n"
                                "if [ -e \"$file\" ]\n"
                                "then\n"
                                "rm $file\n";
        string ldapend = "fi";


        while(getline(read, line)){
            if(line.length()<=40)
                continue;

            if(line.find("<error") == string::npos)
                continue;

                string author = GetData("author",line);

            ldap += "/usr/bin/ldapsearch -x -h xldap.cern.ch \\\n";
            ldap += "\t-b 'OU=Users,OU=Organic Units,DC=cern,DC=ch' '(&(objectClass=user)
            ldap += "\tsAMAccountName \\\n";
            ldap += "\t>>ldap.txt\n\n";

        }

        ofstream write("ldap.sh");
        write << ldapstart + ldap + ldapend;


        return 0;
```

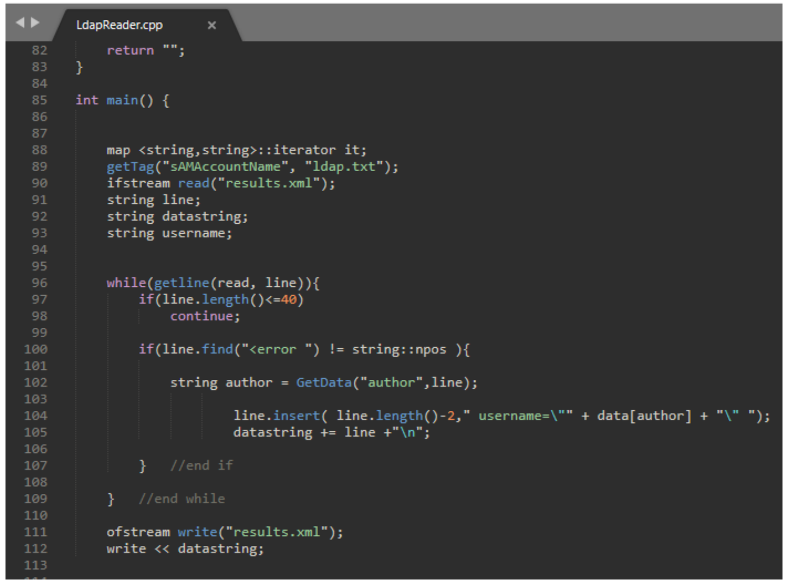
This is ldap.txt file. output of ldap.sh shell file



ldap.txt

```
# extended LDIF
#
# LDAPv3
# base <OU=Users,OU=Organic Units,DC=cern,DC=ch> with scope subtree
# filter: (&(objectClass=user) (displayName=luka todua))
# requesting: sAMAccountName
#

# ltodua, Users, Organic Units, cern.ch
dn: CN=ltodua,OU=Users,OU=Organic Units,DC=cern,DC=ch
sAMAccountName: ltodua

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1

```

With LdapReader.cpp file, we read output of ldap.sh, ldap.txt file, from where we find authors usernames and insert it into new defects Xml file

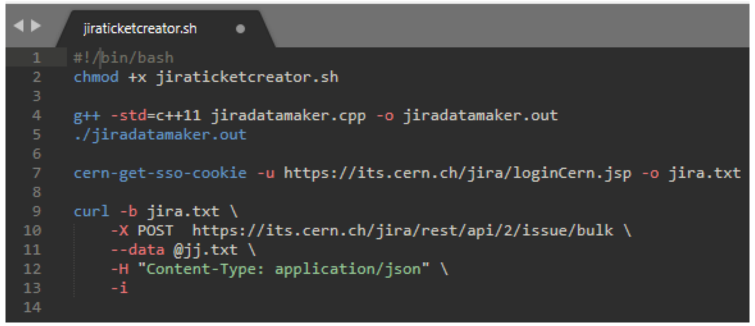


LdapReader.cpp

```
        return "";
}

int main() {


        map <string,string>::iterator it;
        getTag("sAMAccountName", "ldap.txt");
        ifstream read("results.xml");
        string line;
        string datastring;
        string username;


        while(getline(read, line)){
            if(line.length()<=40)
                continue;

            if(line.find("<error ") != string::npos ){

                string author = GetData("author",line);

                        line.insert( line.length()-2," username=\"" + data[author] + "\" ");
                        datastring += line +"\n";

            }   //end if

        }   //end while

        ofstream write("results.xml");
        write << datastring;

```

Automation Files

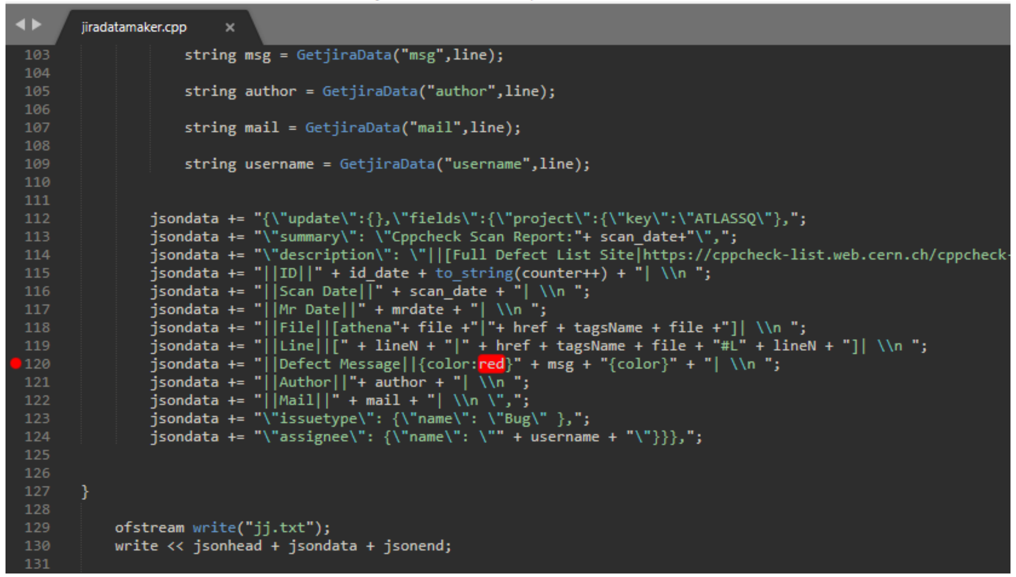
jiraticketcreator.sh file generates json file for Jira REST API from new defects xml file, with cern-get-sso-cookie command we get authentication cookie file from Kerberos ticket and with CURL POST request we send Jira REST API generated file to Jira web page for ticket creation.



jiraticketcreator.sh

```
#!/bin/bash
chmod +x jiraticketcreator.sh

g++ -std=c++11 jiradatamaker.cpp -o jiradatamaker.out
./jiradatamaker.out

cern-get-sso-cookie -u https://its.cern.ch/jira/loginCern.jsp -o jira.txt

curl -b jira.txt \
    -X POST  https://its.cern.ch/jira/rest/api/2/issue/bulk \
    --data @jj.txt \
    -H "Content-Type: application/json" \
    -i

```

This is jiradatamaker.cpp file, here we process new defects xml to create json format data file for Jira REST API

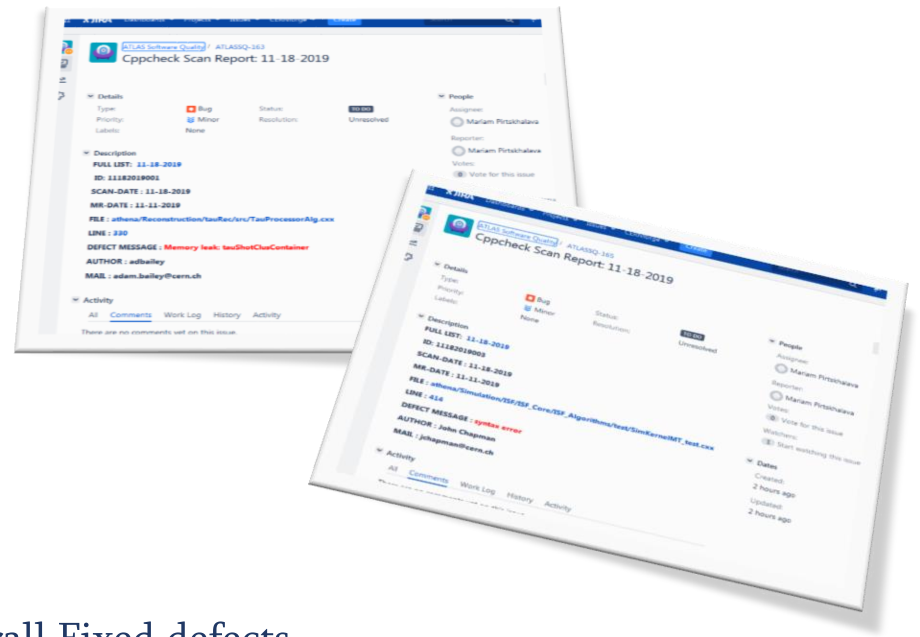


jiradatamaker.cpp

```
                string msg = GetjiraData("msg",line);

                string author = GetjiraData("author",line);

                string mail = GetjiraData("mail",line);

                string username = GetjiraData("username",line);


            jsondata += "{\"update\":{},\"fields\":{\"project\":{\"key\":\"ATLASSQ\"},";
            jsondata += "\"summary\": \"Cppcheck Scan Report:"+ scan_date+"\",";
            jsondata += "\"description\": \"||[Full Defect List Site|https://cppcheck-list.web.cern.ch/cppcheck
            jsondata += "||ID||" + id_date + to_string(counter++) + "| \\n ";
            jsondata += "||Scan Date||" + scan_date + "| \\n ";
            jsondata += "||Mr Date||" + mrdate + "| \\n ";
            jsondata += "||File||[athena"+ file +"|"+ href + tagsName + file +"]| \\n ";
            jsondata += "||Line||[" + lineN + "|" + href + tagsName + file + "#L" + lineN + "]| \\n ";
            jsondata += "||Defect Message||{color:red}" + msg + "{color}" + "| \\n ";
            jsondata += "||Author||"+ author + "| \\n ";
            jsondata += "||Mail||" + mail + "| \\n \",";
            jsondata += "\"issuetype\": {\"name\": \"Bug\" },";
            jsondata += "\"assignee\": {\"name\": \"" + username + "\"}}},";


    }

        ofstream write("jj.txt");
        write << jsonhead + jsondata + jsonend;

```

- Results of automation:

2. Jira Tickets



ATLAS Software Quality / ATLASSQ-163

Cppcheck Scan Report: 11-18-2019

Details

Type:	Bug	Status:	TO DO
Priority:	Minor	Resolution:	Unresolved
Labels:	None		

Description

FULL LIST: 11-18-2019
ID: 11182019001
SCAN-DATE : 11-18-2019
MR-DATE : 11-11-2019
FILE : athena/Reconstruction/tauRec/src/TauProcessorAlg.cxx
LINE : 330
DEFECT MESSAGE : Memory leak: tauShotClusContainer
AUTHOR : adbailey
MAIL : adam.bailey@cern.ch

People
Assignee: Mariam Pirtskhalava
Reporter: Mariam Pirtskhalava
Votes: 0 Vote for this issue

Activity
All Comments Work Log History Activity
There are no comments yet on this issue.

ATLAS Software Quality / ATLASSQ-165

Cppcheck Scan Report: 11-18-2019

Details

Type:	Bug	Status:	TO DO
Priority:	Minor	Resolution:	Unresolved
Labels:	None		

Description

FULL LIST: 11-18-2019
ID: 11182019003
SCAN-DATE : 11-18-2019
MR-DATE : 11-11-2019
FILE : athena/Simulation/ISF/ISF_Core/ISF_Algorithms/test/SimKernelMT_test.cxx
LINE : 414
DEFECT MESSAGE : syntax error
AUTHOR : John Chapman
MAIL : jchapman@cern.ch

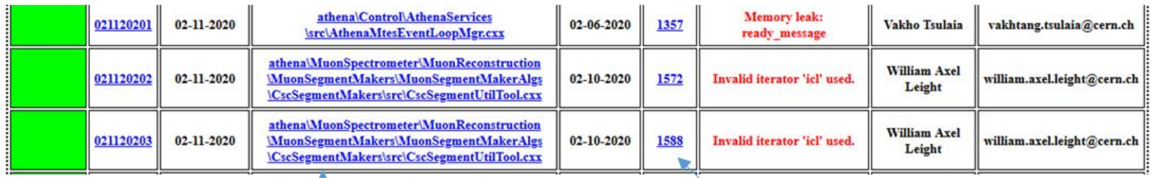
People
Assignee: Mariam Pirtskhalava
Reporter: Mariam Pirtskhalava
Votes: 0 Vote for this issue
Watchers: 1 Start watching this issue

Dates
Created: 2 hours ago
Updated: 2 hours ago

Activity
All Comments Work Log History Activity

1. *.html* table of the new defects

https://cppcheck-list.web.cern.ch/cppcheck-list/



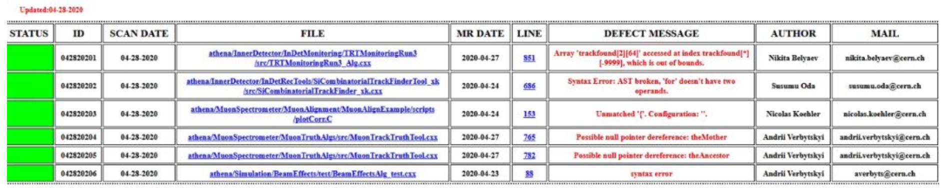
🟩	021120201	02-11-2020	athena\Control\AthenaServices \src\AthenaMtesEventLoopMgr.cxx	02-06-2020	1357	Memory leak: ready_message	Vakho Tsulaia	vakhtang.tsulaia@cern.ch
🟩	021120202	02-11-2020	athena\MuonSpectrometer\MuonReconstruction \MuonSegmentMakers\MuonSegmentMakerAlgs \CscSegmentMakers\src\CscSegmentUtilTool.cxx	02-10-2020	1572	Invalid iterator 'icl' used.	William Axel Leight	william.axel.leight@cern.ch
🟩	021120203	02-11-2020	athena\MuonSpectrometer\MuonReconstruction \MuonSegmentMakers\MuonSegmentMakerAlgs \CscSegmentMakers\src\CscSegmentUtilTool.cxx	02-10-2020	1588	Invalid iterator 'icl' used.	William Axel Leight	william.axel.leight@cern.ch

Opening Code on the Gitlab

Opening Code on the defected string

3. Statistical data

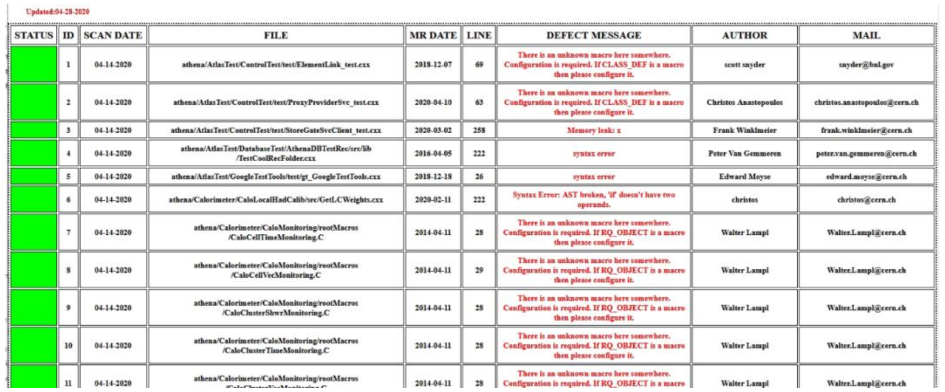
List of Overall Fixed defects



Updated:04-28-2020

STATUS	ID	SCAN DATE	FILE	MR DATE	LINE	DEFECT MESSAGE	AUTHOR	MAIL
🟩	042820201	04-28-2020	athena/InnerDetector/InDetMonitoring/TRTMonitoringRun3 /src/TRTMonitoringRun3_Alg.cxx	2020-04-27	851	Array 'trackfound[2][64]' accessed at index trackfound[*][-9999], which is out of bounds.	Nikita Belyaev	nikita.belyaev@cern.ch
🟩	042820202	04-28-2020	athena/InnerDetector/InDetRecTools/SiCombinatorialTrackFinderTool_xk /src/SiCombinatorialTrackFinder_xk.cxx	2020-04-24	686	Syntax Error: AST broken, 'for' doesn't have two operands.	Susumu Oda	susumu.oda@cern.ch
🟩	042820203	04-28-2020	athena/MuonSpectrometer/MuonAlignment/MuonAlignExample/scripts /plotCorr.C	2020-04-24	153	Unmatched '{'. Configuration: ''.	Nicolas Koehler	nicolas.koehler@cern.ch
🟩	042820204	04-28-2020	athena/MuonSpectrometer/MuonTruthAlgs/src/MuonTrackTruthTool.cxx	2020-04-27	765	Possible null pointer dereference: theMother	Andrii Verbytskyi	andrii.verbytskyi@cern.ch
🟩	042820205	04-28-2020	athena/MuonSpectrometer/MuonTruthAlgs/src/MuonTrackTruthTool.cxx	2020-04-27	782	Possible null pointer dereference: theAncestor	Andrii Verbytskyi	andrii.verbytskyi@cern.ch
🟩	042820206	04-28-2020	athena/Simulation/BeamEffects/test/BeamEffectsAlg_test.cxx	2020-04-23	88	syntax error	Andrii Verbytskyi	averbyts@cern.ch

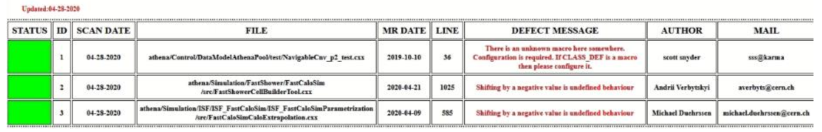
List of Overall defects



Updated:04-28-2020

STATUS	ID	SCAN DATE	FILE	MR DATE	LINE	DEFECT MESSAGE	AUTHOR	MAIL
🟩	1	04-14-2020	athena/AtlasTest/ControlTest/test/ElementLink_test.cxx	2018-12-07	69	There is an unknown macro here somewhere. Configuration is required. If CLASS_DEF is a macro then please configure it.	scott snyder	snyder@bnl.gov
🟩	2	04-14-2020	athena/AtlasTest/ControlTest/test/ProxyProviderSvc_test.cxx	2020-04-10	63	There is an unknown macro here somewhere. Configuration is required. If CLASS_DEF is a macro then please configure it.	Christos Anastopoulos	christos.anastopoulos@cern.ch
🟩	3	04-14-2020	athena/AtlasTest/ControlTest/test/StoreGateSvcClient_test.cxx	2020-03-02	258	Memory leak: x	Frank Winklmeier	frank.winklmeier@cern.ch
🟩	4	04-14-2020	athena/AtlasTest/DatabaseTest/AthenaDBTestRec/src/lib /TestCoolRecFolder.cxx	2016-04-05	222	syntax error	Peter Van Gemmeren	peter.van.gemmeren@cern.ch
🟩	5	04-14-2020	athena/AtlasTest/GoogleTestTools/test/gt_GoogleTestTools.cxx	2018-12-18	26	syntax error	Edward Moyse	edward.moyse@cern.ch
🟩	6	04-14-2020	athena/Calorimeter/CaloLocalHadCalib/src/GetLCWeights.cxx	2020-02-11	222	Syntax Error: AST broken, 'if' doesn't have two operands.	christos	christos@cern.ch
🟩	7	04-14-2020	athena/Calorimeter/CaloMonitoring/rootMacros /CaloCellTimeMonitoring.C	2014-04-11	28	There is an unknown macro here somewhere. Configuration is required. If RQ_OBJECT is a macro then please configure it.	Walter Lampl	Walter.Lampl@cern.ch
🟩	8	04-14-2020	athena/Calorimeter/CaloMonitoring/rootMacros /CaloCellVecMonitoring.C	2014-04-11	29	There is an unknown macro here somewhere. Configuration is required. If RQ_OBJECT is a macro then please configure it.	Walter Lampl	Walter.Lampl@cern.ch
🟩	9	04-14-2020	athena/Calorimeter/CaloMonitoring/rootMacros /CaloClusterShwrMonitoring.C	2014-04-11	28	There is an unknown macro here somewhere. Configuration is required. If RQ_OBJECT is a macro then please configure it.	Walter Lampl	Walter.Lampl@cern.ch
🟩	10	04-14-2020	athena/Calorimeter/CaloMonitoring/rootMacros /CaloClusterTimeMonitoring.C	2014-04-11	28	There is an unknown macro here somewhere. Configuration is required. If RQ_OBJECT is a macro then please configure it.	Walter Lampl	Walter.Lampl@cern.ch
🟩	11	04-14-2020	athena/Calorimeter/CaloMonitoring/rootMacros	2014-04-11	28	There is an unknown macro here somewhere. Configuration is required. If RQ_OBJECT is a macro	Walter Lampl	Walter.Lampl@cern.ch

List of Fixed defects since the last scan



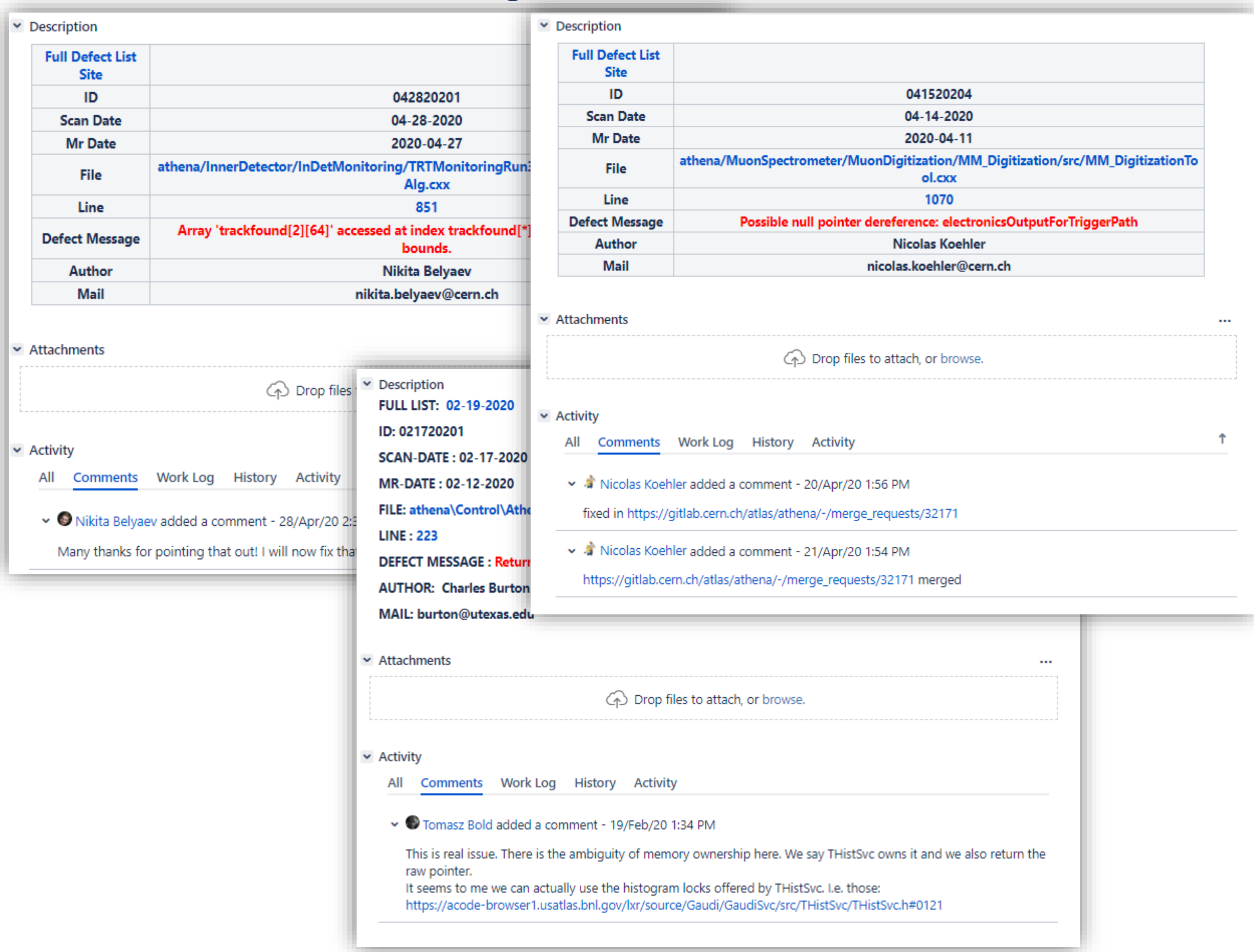
Updated:04-28-2020

STATUS	ID	SCAN DATE	FILE	MR DATE	LINE	DEFECT MESSAGE	AUTHOR	MAIL
🟩	1	04-28-2020	athena/Control/DataModelAthenaPool/test/NavigableCnv_p2_test.cxx	2019-10-10	36	There is an unknown macro here somewhere. Configuration is required. If CLASS_DEF is a macro then please configure it.	scott snyder	sss@karma
🟩	2	04-28-2020	athena/Simulation/FastShower/FastCaloSim /src/FastShowerCellBuilderTool.cxx	2020-04-21	1025	Shifting by a negative value is undefined behaviour	Andrii Verbytskyi	averbyts@cern.ch
🟩	3	04-28-2020	athena/Simulation/ISF/ISF_FastCaloSim/ISF_FastCaloSimParametrization /src/FastCaloSimCaloExtrapolation.cxx	2020-04-09	585	Shifting by a negative value is undefined behaviour	Michael Duehrssen	michael.duehrssen@cern.ch

- In our SQ group we have 31 authors

1. Adam Baley
2. Adam Edward Barton
3. Ahmed Hasib
4. Andrei Sukharev
5. Apostolos Tsirigotis
6. Benedict Tobias Winter
7. Ban Nachman
8. Charles Barton
9. Chris Lee
10. Christos Anastopoulos
11. Dario Barberis
12. Edward Moyse
13. Goetz Gaycken
14. Hao Xu
15. John Derek Chapman
16. Matous Vozak
17. Nicolas Koehler
18. Nikita Belyaev
19. Pascal Boeschoten
20. Peter Onysi
21. Rafal Bielski
22. Ruth POttgen
23. Scott Snyder
24. Shaun Roe
25. Soshi Tsuno
26. Susumo Oda
27. Tim Martin
28. Tomasz Bold
29. Walter Lampl
30. Vakhtang Tsulaia
31. William Axel Leight

- We have good feedbacks from authors:



Description

Full Defect List Site	
ID	042820201
Scan Date	04-28-2020
Mr Date	2020-04-27
File	athena/InnerDetector/InDetMonitoring/TRTMonitoringRun Alg.cxx
Line	851
Defect Message	Array 'trackfound[2][64]' accessed at index trackfound[*] bounds.
Author	Nikita Belyaev
Mail	nikita.belyaev@cern.ch

Attachments

Drop files

Activity

All Comments Work Log History Activity

Nikita Belyaev added a comment - 28/Apr/20 2:

Many thanks for pointing that out! I will now fix tha

Description

Full Defect List Site	
ID	041520204
Scan Date	04-14-2020
Mr Date	2020-04-11
File	athena/MuonSpectrometer/MuonDigitization/MM_Digitization/src/MM_DigitizationTool.cxx
Line	1070
Defect Message	Possible null pointer dereference: electronicsOutputForTriggerPath
Author	Nicolas Koehler
Mail	nicolas.koehler@cern.ch

Attachments

Drop files to attach, or browse.

Activity

All Comments Work Log History Activity

Nicolas Koehler added a comment - 20/Apr/20 1:56 PM

fixed in https://gitlab.cern.ch/atlas/athena/-/merge_requests/32171

Nicolas Koehler added a comment - 21/Apr/20 1:54 PM

https://gitlab.cern.ch/atlas/athena/-/merge_requests/32171 merged

Description

FULL LIST: 02-19-2020

ID: 021720201

SCAN-DATE : 02-17-2020

MR-DATE : 02-12-2020

FILE: athena\Control\Athe

LINE : 223

DEFECT MESSAGE : Retur

AUTHOR: Charles Burton

MAIL: burton@utexas.edu

Attachments

Drop files to attach, or browse.

Activity

All Comments Work Log History Activity

Tomasz Bold added a comment - 19/Feb/20 1:34 PM

This is real issue. There is the ambiguity of memory ownership here. We say THistSvc owns it and we also return the raw pointer.
It seems to me we can actually use the histogram locks offered by THistSvc. I.e. those:
https://acode-browser1.usatlas.bnl.gov/lxr/source/Gaudi/GaudiSvc/src/THistSvc/THistSvc.h#0121

Future plans:

- ➢Development of Cppcheck Scan for individual MR’s
- ➢To make Individual MR’s Scanning process automated
- ➢Startup Coverity Scan of the Athena repository

Development of Cppcheck Scan for individual MR’s

- Objective is to read latest MR’s from Gitlab, initiate individual Cppcheck scanning and provide defects report back on Gitlab
- We have Developed 6 steps for this process:
    1. Access latest MR’s from the Master branch
    2. Get MR’s file path
    3. Clone Athena repository
    4. Scan MR’s file path with Cppcheck and generate XML file
    5. Create Individual MR’s template and fill it with XML data
    6. Upload template on Gitlab into MR section
- We did 5 MR’s Scan for testing purpose

Startup Coverity Scan of the Athena repository

- Re-build Coverity scan of the Athena full repository
- For the moment we have issues with the Coverity build. We are trying to build ATLAS Externals from the nightly
- After Coverity setup, we will develop the Coverity scan process
- Then it will be possible to run the Coverity scans of Athena full repository
- Also,We will make this process automated

Thanks for Your Attention!