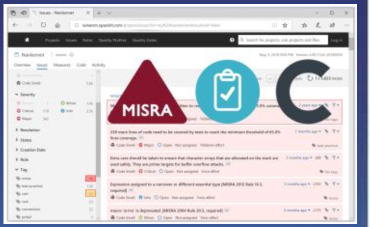
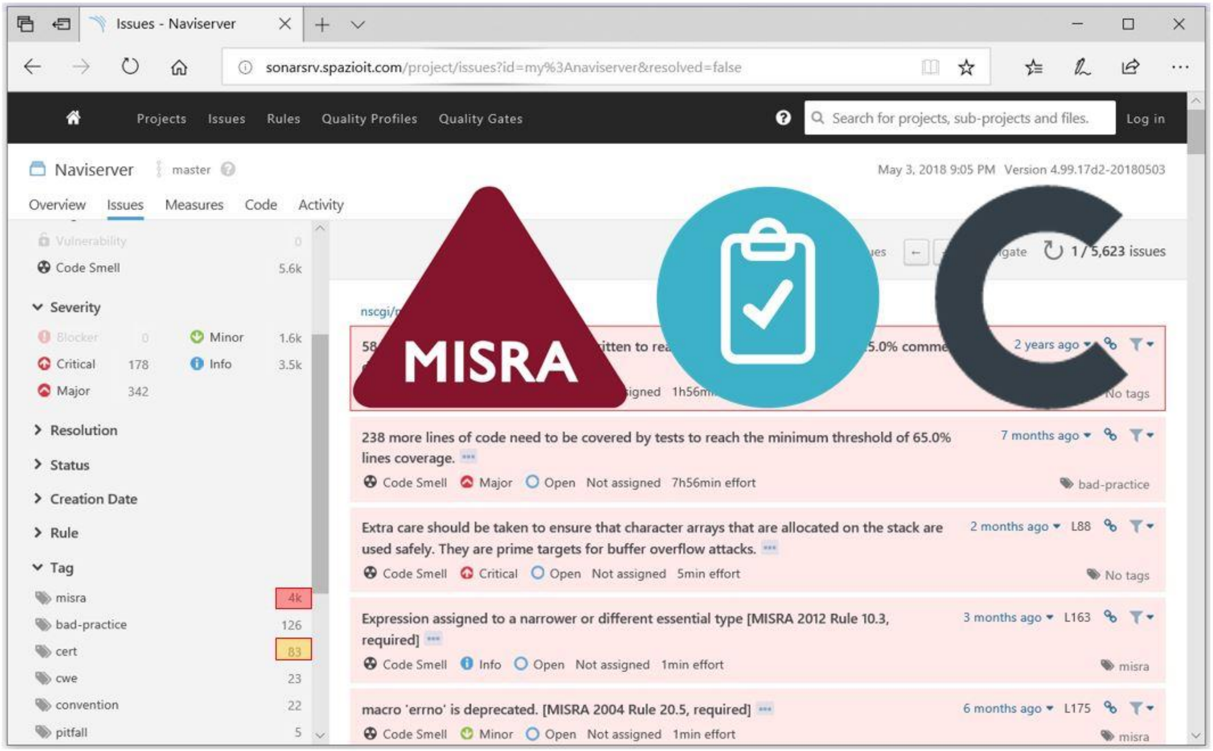


C Guidelines Compliance and Deviations (MISRA and CERT Cases)

Ada-Europe 2018 Lisbon



Issues - Naviserver

sonarsrv.spazioit.com/project/issues?id=my%3Anaviserver&resolved=false

Projects Issues Rules Quality Profiles Quality Gates

Search for projects, sub-projects and files. Log in

Naviserver master

May 3, 2018 9:05 PM Version 4.99.17d2-20180503

Overview Issues Measures Code Activity

Vulnerability	0
Code Smell	5.6k

Severity

Blocker	0	Minor	1.6k
Critical	178	Info	3.5k
Major	342		

Resolution

Status

Creation Date

Rule

Tag

misra	4k
bad-practice	126
cert	83
cwe	23
convention	22
pitfall	5

1 / 5,623 issues

nscgi/

58 … itten to rea … 5.0% comme … 2 years ago

…igned 1h56m … No tags

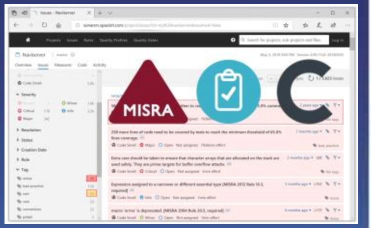
238 more lines of code need to be covered by tests to reach the minimum threshold of 65.0% lines coverage.
Code Smell Major Open Not assigned 7h56min effort — 7 months ago — bad-practice

Extra care should be taken to ensure that character arrays that are allocated on the stack are used safely. They are prime targets for buffer overflow attacks.
Code Smell Critical Open Not assigned 5min effort — 2 months ago L88 — No tags

Expression assigned to a narrower or different essential type [MISRA 2012 Rule 10.3, required]
Code Smell Info Open Not assigned 1min effort — 3 months ago L163 — misra

macro 'errno' is deprecated. [MISRA 2004 Rule 20.5, required]
Code Smell Minor Open Not assigned 1min effort — 6 months ago L175 — misra

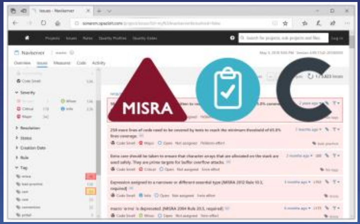
Maurizio Martignano
Spazio IT – Soluzioni Informatiche s.a.s
Via Manzoni 40
46030 San Giorgio di Mantova, Mantova
http://www.spazioit.com

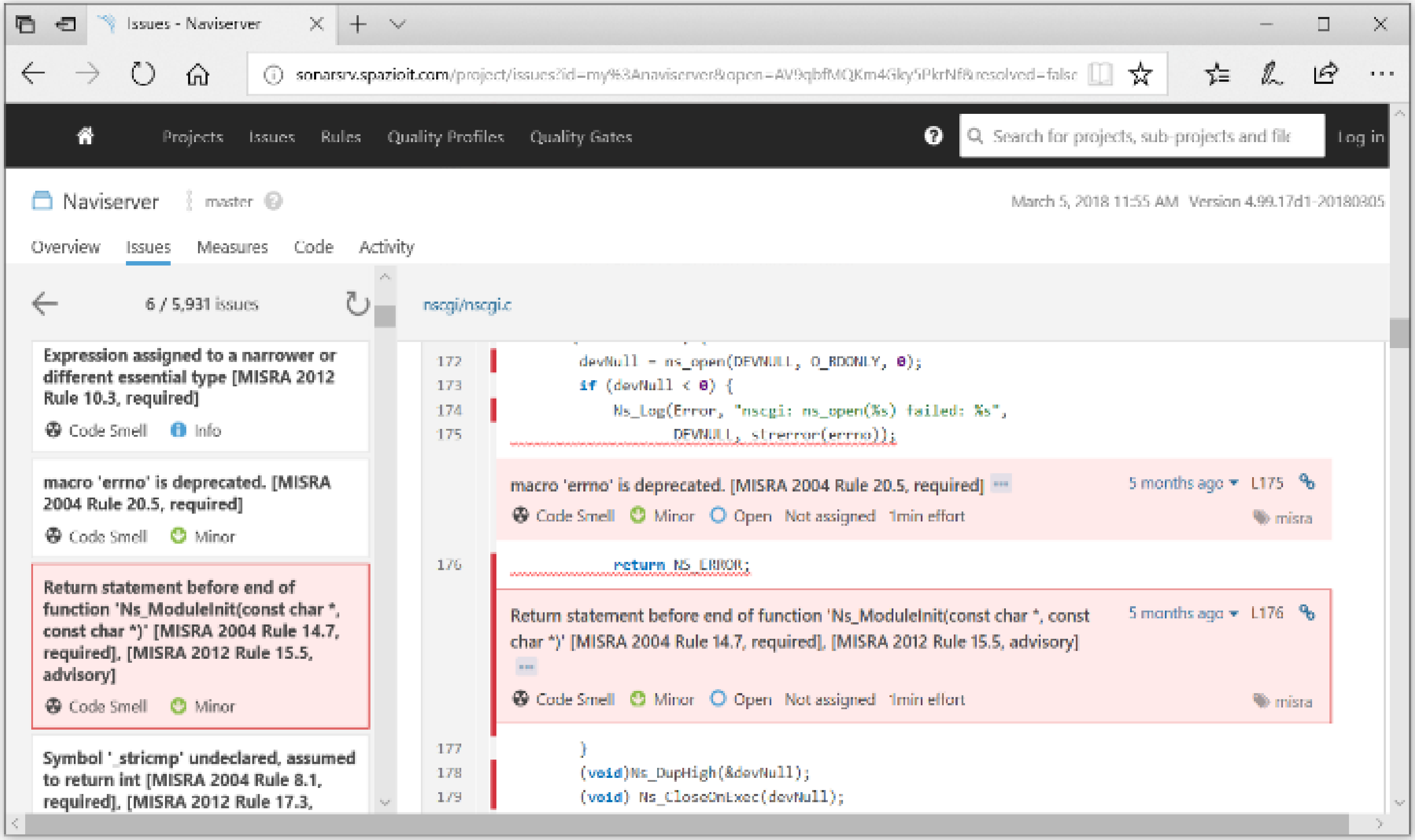


Agenda

- C compliance guidelines and deviations (intro)
- Customization examples
- MISRA and CERT compliance standardization efforts
- Analysis tools
- A viable approach to manage compliance

C compliance guidelines and deviations (intro)





Issues - Naviserver

sonarsrv.spazioit.com/project/issues?id=my%3Anaviserver&open=AV9qbfMQKm4Gky5PkrNf&resolved=false

Naviserver master

March 5, 2018 11:55 AM Version 4.99.17d1-20180305

6 / 5,931 issues

Expression assigned to a narrower or different essential type [MISRA 2012 Rule 10.3, required]
Code Smell · Info

macro 'errno' is deprecated. [MISRA 2004 Rule 20.5, required]
Code Smell · Minor

Return statement before end of function 'Ns_ModuleInit(const char *, const char *)' [MISRA 2004 Rule 14.7, required], [MISRA 2012 Rule 15.5, advisory]
Code Smell · Minor

Symbol '_stricmp' undeclared, assumed to return int [MISRA 2004 Rule 8.1, required], [MISRA 2012 Rule 17.3,

nsgi/nscgi.c

```
172         devNull = ns_open(DEVNULL, O_RDONLY, 0);
173         if (devNull < 0) {
174             Ns_Log(Error, "nscgi: ns_open(%s) failed: %s",
175                    DEVNULL, strerror(errno));
```

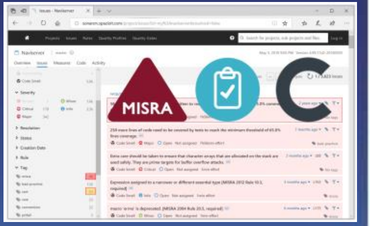
macro 'errno' is deprecated. [MISRA 2004 Rule 20.5, required] — 5 months ago · L175
Code Smell · Minor · Open · Not assigned · 1min effort · misra

```
176             return NS_ERROR;
```

Return statement before end of function 'Ns_ModuleInit(const char *, const char *)' [MISRA 2004 Rule 14.7, required], [MISRA 2012 Rule 15.5, advisory] — 5 months ago · L176
Code Smell · Minor · Open · Not assigned · 1min effort · misra

```
177         }
178         (void)Ns_DupHigh(&devNull);
179         (void) Ns_CloseOnExec(devNull);
```

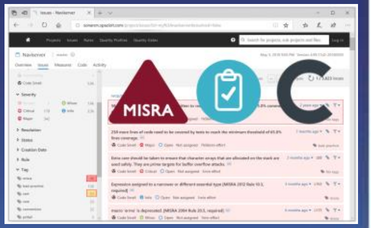
C compliance guidelines and deviations (intro)



- MISRA C:2012 and SEI CERT C provide a set of guidelines (rules and directives – MISRA or rules and recommendations - CERT) designed to help developers in writing quality code, i.e. code that is safer, more secure, understandable and maintainable.

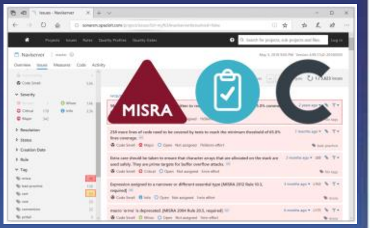
- It is not always possible to adhere to all these guidelines and this why in several software development projects deviations and compliance levels are established in the context of so called customization or tailoring activities.

C compliance guidelines and deviations (intro)



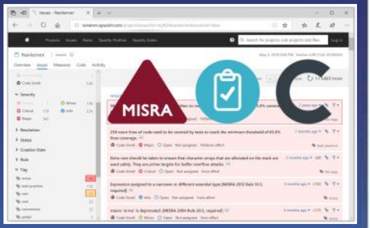
- The customization activity can be performed very formally, adhering once again to specific and additional standards, or rather informally on a project by project base, according to the specific needs and actual limitations of the project itself.

- The customization strategies adopted by actual projects range from not allowing any deviation to accepting every documented deviation.



Customization examples

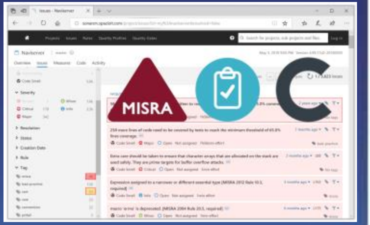
Integral types size and sign

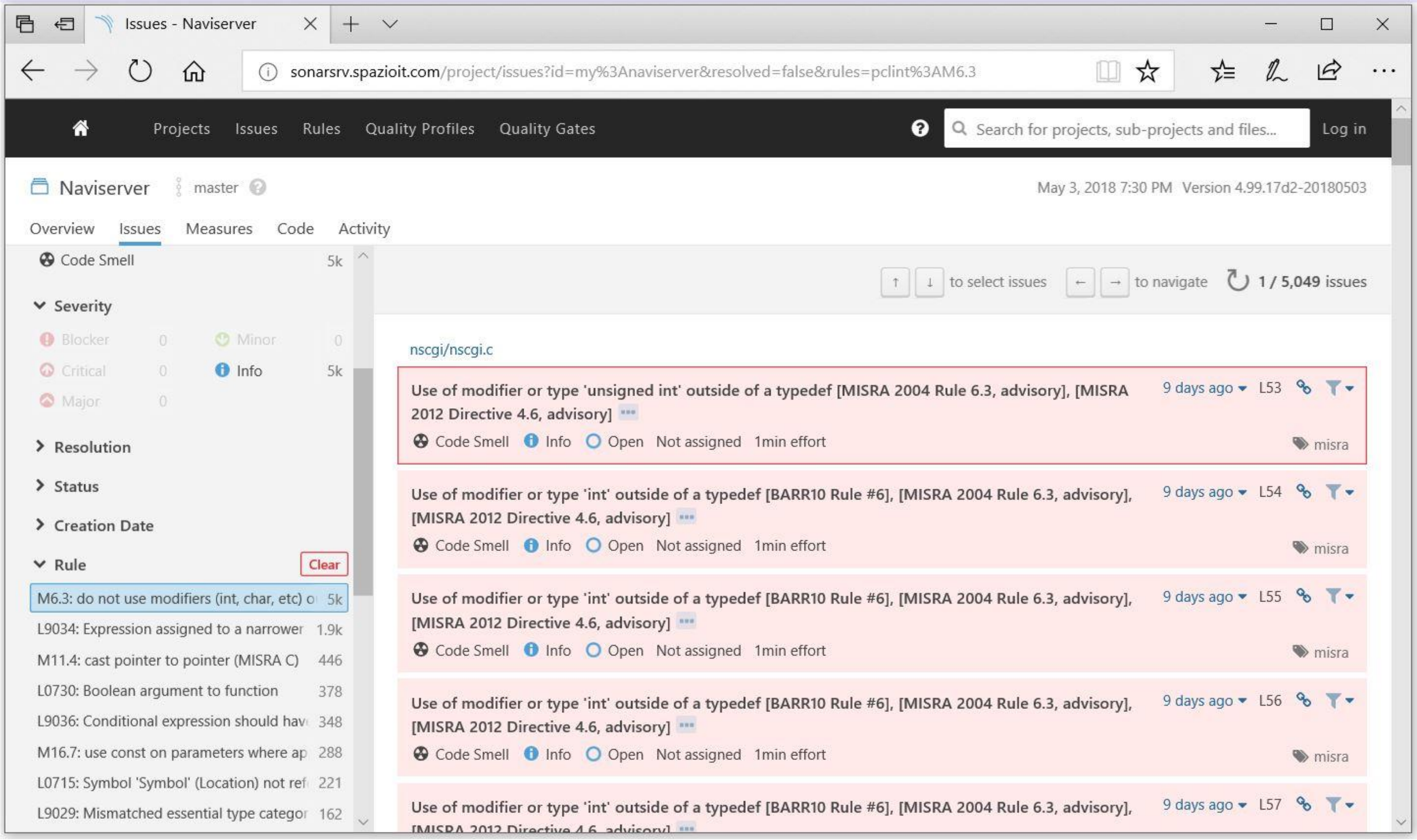


- MISRA C 2012 – Directive 4.6 (Advisory) – “typedefs indicating size and sign should be used instead of the basic integral types”.

- This MISRA directive recommends to use the “new” C99 integral types contained in “stdint.h” instead of the “old” integral numerical types, e.g. int8_t instead of char, uint8_t instead of unsigned char, int32_t instead of int (or long depending on the hardware architecture), etc…

Integral types size and sign





Issues - Naviserver
sonarsrv.spazioit.com/project/issues?id=my%3Anaviserver&resolved=false&rules=pclint%3AM6.3
Projects Issues Rules Quality Profiles Quality Gates
Search for projects, sub-projects and files...
Log in

Naviserver master

May 3, 2018 7:30 PM Version 4.99.17d2-20180503

Overview Issues Measures Code Activity

Code Smell	5k

Severity

Blocker	0	Minor	0
Critical	0	Info	5k
Major	0		

Resolution

Status

Creation Date

Rule Clear

M6.3: do not use modifiers (int, char, etc) o	5k
L9034: Expression assigned to a narrower	1.9k
M11.4: cast pointer to pointer (MISRA C)	446
L0730: Boolean argument to function	378
L9036: Conditional expression should hav	348
M16.7: use const on parameters where ap	288
L0715: Symbol 'Symbol' (Location) not ref	221
L9029: Mismatched essential type categor	162

↑ ↓ to select issues ← → to navigate 1 / 5,049 issues

nscgi/nscgi.c

Use of modifier or type 'unsigned int' outside of a typedef [MISRA 2004 Rule 6.3, advisory], [MISRA 2012 Directive 4.6, advisory]
Code Smell Info Open Not assigned 1min effort — 9 days ago L53 misra

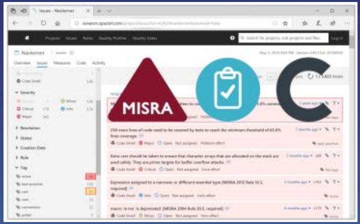
Use of modifier or type 'int' outside of a typedef [BARR10 Rule #6], [MISRA 2004 Rule 6.3, advisory], [MISRA 2012 Directive 4.6, advisory]
Code Smell Info Open Not assigned 1min effort — 9 days ago L54 misra

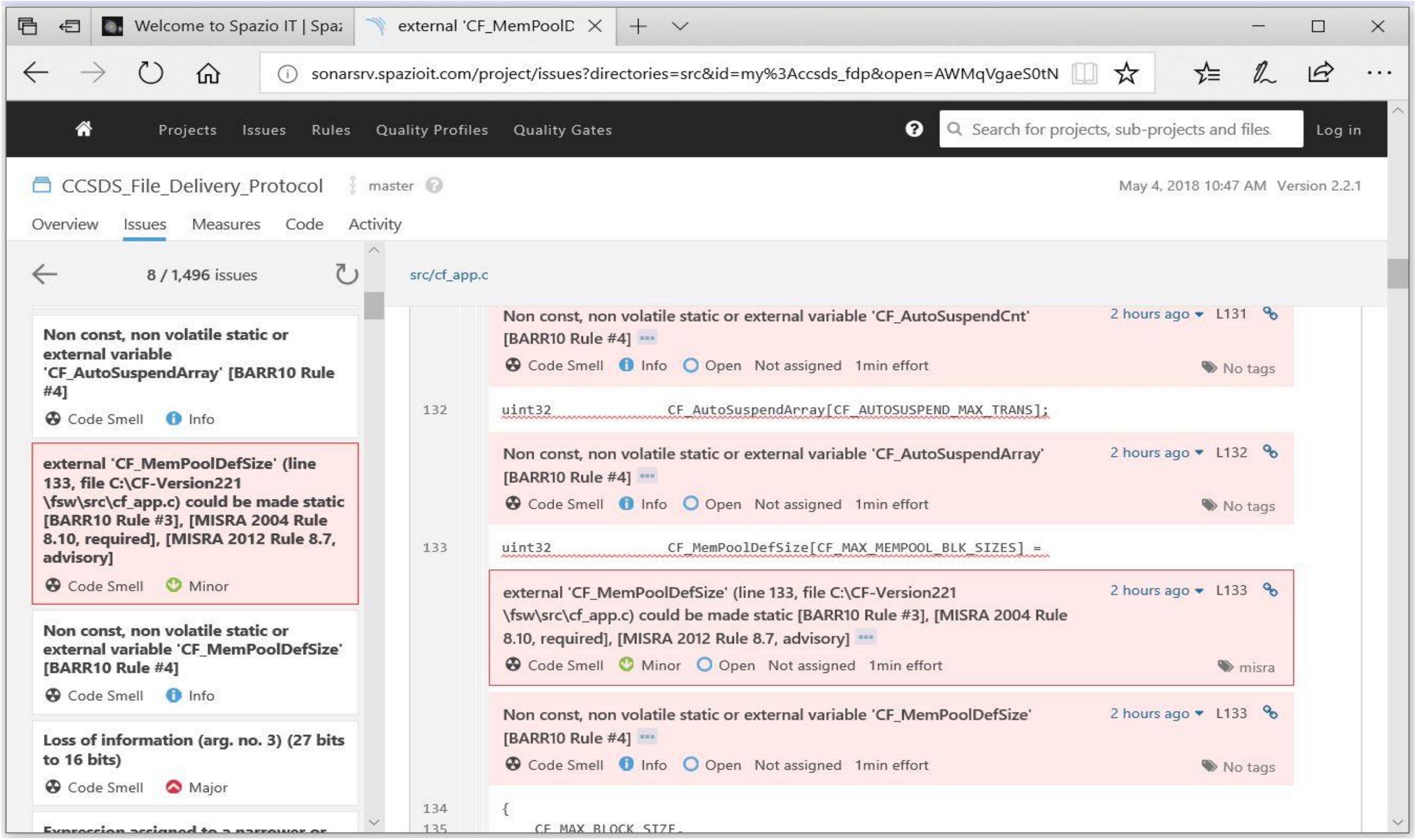
Use of modifier or type 'int' outside of a typedef [BARR10 Rule #6], [MISRA 2004 Rule 6.3, advisory], [MISRA 2012 Directive 4.6, advisory]
Code Smell Info Open Not assigned 1min effort — 9 days ago L55 misra

Use of modifier or type 'int' outside of a typedef [BARR10 Rule #6], [MISRA 2004 Rule 6.3, advisory], [MISRA 2012 Directive 4.6, advisory]
Code Smell Info Open Not assigned 1min effort — 9 days ago L56 misra

Use of modifier or type 'int' outside of a typedef [BARR10 Rule #6], [MISRA 2004 Rule 6.3, advisory], [MISRA 2012 Directive 4.6, advisory] — 9 days ago L57

Integral types size and sign




Welcome to Spazio IT | Spaz
external 'CF_MemPoolD
sonarsrv.spazioit.com/project/issues?directories=src&id=my%3Accsds_fdp&open=AWMqVgaeS0tN
Projects
Issues
Rules
Quality Profiles
Quality Gates
Search for projects, sub-projects and files.
Log in
CCSDS_File_Delivery_Protocol
master
May 4, 2018 10:47 AM Version 2.2.1
Overview
Issues
Measures
Code
Activity
8 / 1,496 issues
src/cf_app.c

Non const, non volatile static or external variable 'CF_AutoSuspendArray' [BARR10 Rule #4]
Code Smell Info

external 'CF_MemPoolDefSize' (line 133, file C:\CF-Version221\fsw\src\cf_app.c) could be made static [BARR10 Rule #3], [MISRA 2004 Rule 8.10, required], [MISRA 2012 Rule 8.7, advisory]
Code Smell Minor

Non const, non volatile static or external variable 'CF_MemPoolDefSize' [BARR10 Rule #4]
Code Smell Info

Loss of information (arg. no. 3) (27 bits to 16 bits)
Code Smell Major

Non const, non volatile static or external variable 'CF_AutoSuspendCnt' [BARR10 Rule #4] — 2 hours ago L131
Code Smell Info Open Not assigned 1min effort No tags

```
uint32            CF_AutoSuspendArray[CF_AUTOSUSPEND_MAX_TRANS];
```

Non const, non volatile static or external variable 'CF_AutoSuspendArray' [BARR10 Rule #4] — 2 hours ago L132
Code Smell Info Open Not assigned 1min effort No tags

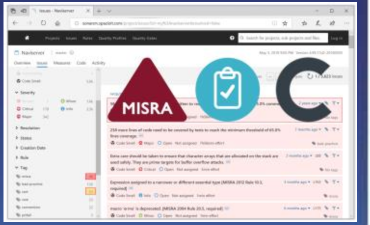
```
uint32            CF_MemPoolDefSize[CF_MAX_MEMPOOL_BLK_SIZES] =
```

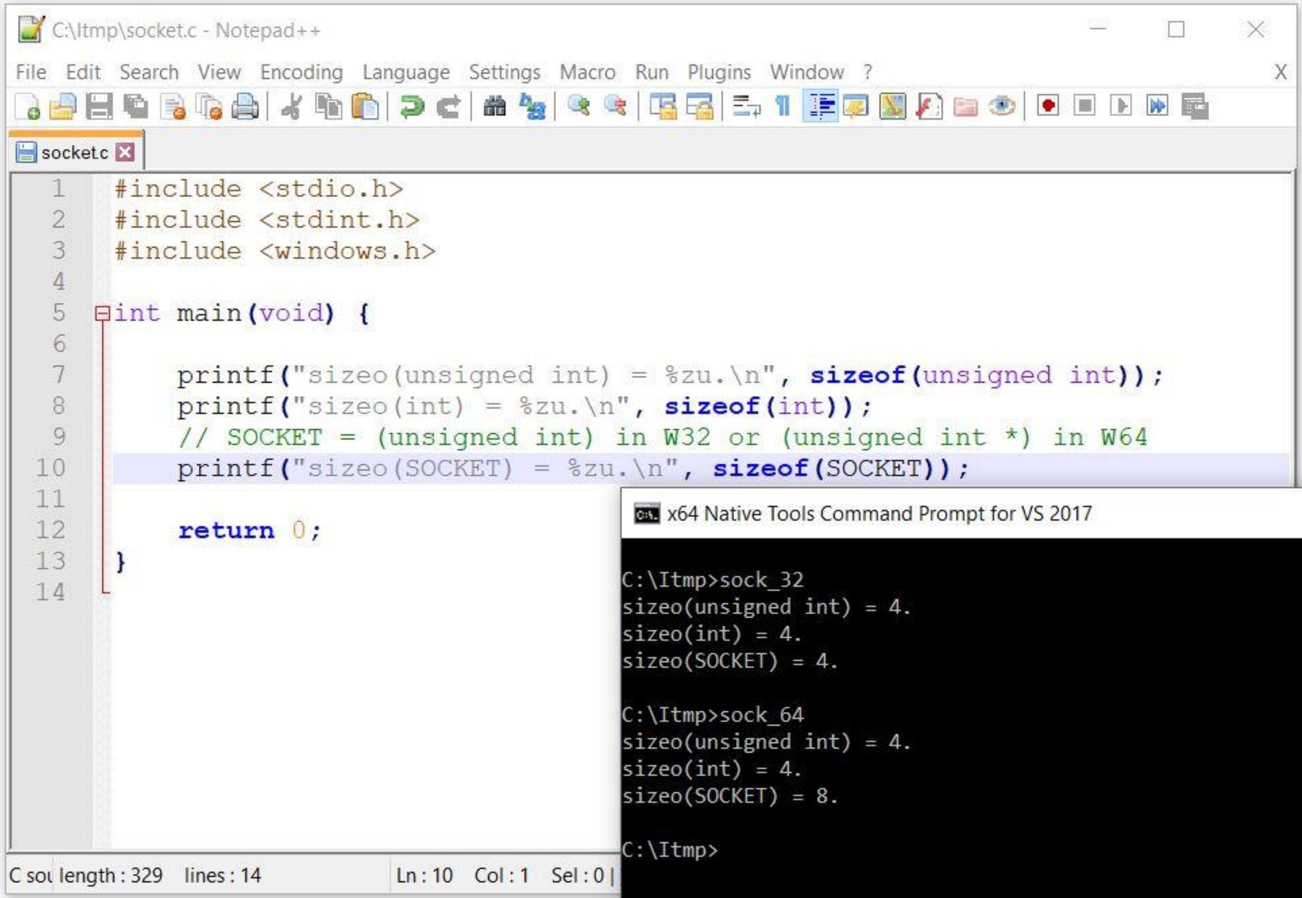
external 'CF_MemPoolDefSize' (line 133, file C:\CF-Version221\fsw\src\cf_app.c) could be made static [BARR10 Rule #3], [MISRA 2004 Rule 8.10, required], [MISRA 2012 Rule 8.7, advisory] — 2 hours ago L133
Code Smell Minor Open Not assigned 1min effort misra

Non const, non volatile static or external variable 'CF_MemPoolDefSize' [BARR10 Rule #4] — 2 hours ago L133
Code Smell Info Open Not assigned 1min effort No tags

```
{
    CF_MAX_BLOCK_SIZE,
```

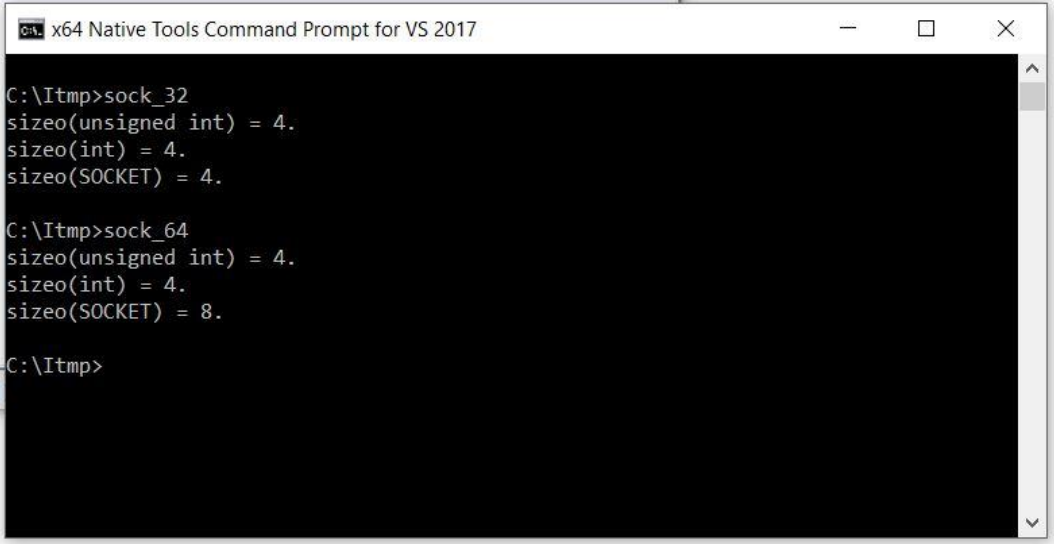
(Integral) types size and sign





C:\Itmp\socket.c - Notepad++

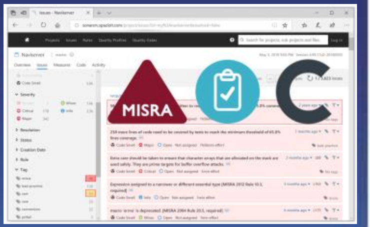
```c
#include <stdio.h>
#include <stdint.h>
#include <windows.h>

int main(void) {

    printf("sizeo(unsigned int) = %zu.\n", sizeof(unsigned int));
    printf("sizeo(int) = %zu.\n", sizeof(int));
    // SOCKET = (unsigned int) in W32 or (unsigned int *) in W64
    printf("sizeo(SOCKET) = %zu.\n", sizeof(SOCKET));

    return 0;
}
```



x64 Native Tools Command Prompt for VS 2017

```
C:\Itmp>sock_32
sizeo(unsigned int) = 4.
sizeo(int) = 4.
sizeo(SOCKET) = 4.

C:\Itmp>sock_64
sizeo(unsigned int) = 4.
sizeo(int) = 4.
sizeo(SOCKET) = 8.

C:\Itmp>
```

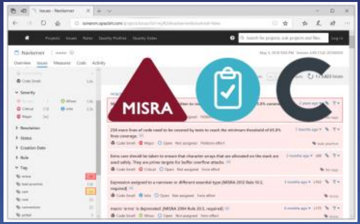
(Not NULL) Pointer Function Parameters

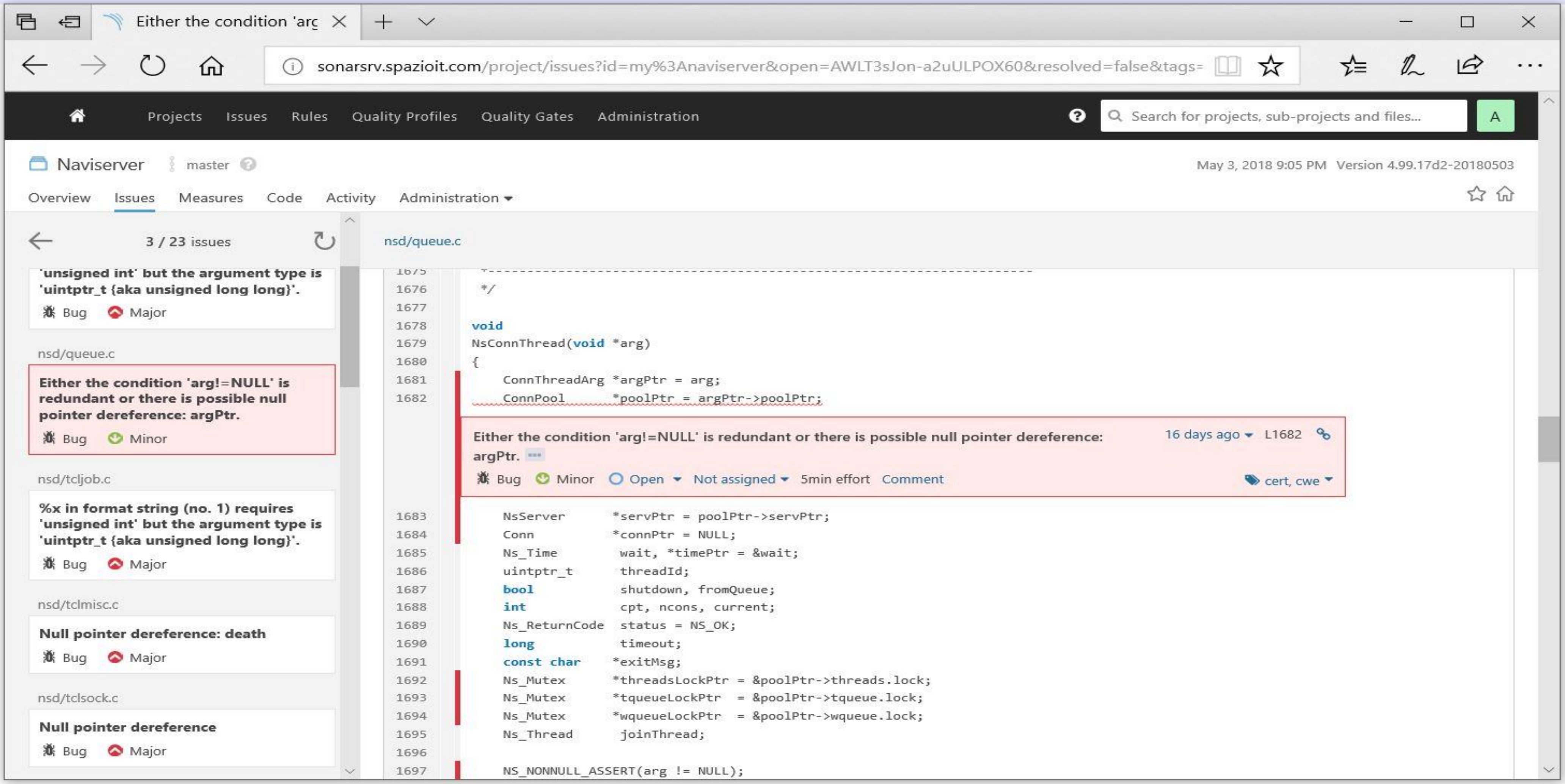


- Involved rules:
  - MISRA C 2012 – Directive 4.11 (Required) – “The validity of parameters passed to library functions shall be checked”.
  - CERT – EXP34-C – “Do not dereference null pointers”
  - CERT – MEM10-C – “Define and use a pointer validation function”

- The “common sense” rule is that function parameters should be checked for their validity. In the case of pointer function parameters, a first and simple check consists in verifying they are not NULL.

(Not NULL) Pointer Function Parameters




Either the condition 'arc
sonarsrv.spazioit.com/project/issues?id=my%3Anaviserver&open=AWLT3sJon-a2uULPOX60&resolved=false&tags=
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects, sub-projects and files...
A
Naviserver
master
May 3, 2018 9:05 PM Version 4.99.17d2-20180503
Overview
Issues
Measures
Code
Activity
Administration
3 / 23 issues
nsd/queue.c

'unsigned int' but the argument type is 'uintptr_t {aka unsigned long long}'.
Bug Major

nsd/queue.c

Either the condition 'arg!=NULL' is redundant or there is possible null pointer dereference: argPtr.
Bug Minor

nsd/tcljob.c

%x in format string (no. 1) requires 'unsigned int' but the argument type is 'uintptr_t {aka unsigned long long}'.
Bug Major

nsd/tclmisc.c

Null pointer dereference: death
Bug Major

nsd/tclsock.c

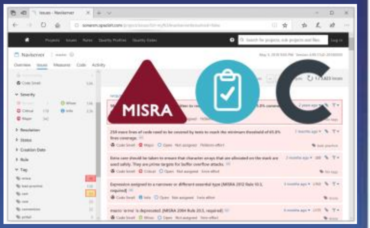
Null pointer dereference
Bug Major

```
1675   *----------------------------------------------------------------------
1676   */
1677
1678  void
1679  NsConnThread(void *arg)
1680  {
1681      ConnThreadArg *argPtr = arg;
1682      ConnPool      *poolPtr = argPtr->poolPtr;
```

Either the condition 'arg!=NULL' is redundant or there is possible null pointer dereference: argPtr.
Bug Minor Open Not assigned 5min effort Comment
16 days ago L1682
cert, cwe

```
1683      NsServer      *servPtr = poolPtr->servPtr;
1684      Conn          *connPtr = NULL;
1685      Ns_Time        wait, *timePtr = &wait;
1686      uintptr_t      threadId;
1687      bool           shutdown, fromQueue;
1688      int            cpt, ncons, current;
1689      Ns_ReturnCode  status = NS_OK;
1690      long           timeout;
1691      const char    *exitMsg;
1692      Ns_Mutex      *threadsLockPtr = &poolPtr->threads.lock;
1693      Ns_Mutex      *tqueueLockPtr  = &poolPtr->tqueue.lock;
1694      Ns_Mutex      *wqueueLockPtr  = &poolPtr->wqueue.lock;
1695      Ns_Thread      joinThread;
1696
1697      NS_NONNULL_ASSERT(arg != NULL);
```

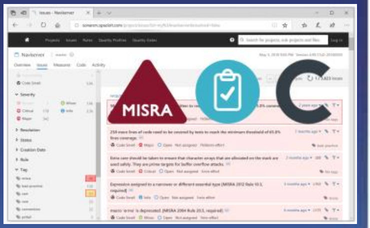
(Not NULL) Pointer Function Parameters



- Design Time
  - ASSERT type check

- Run Time
  - IF type check

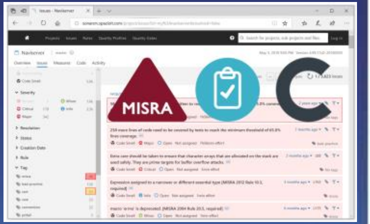
Forbidden Conversions (and then again)

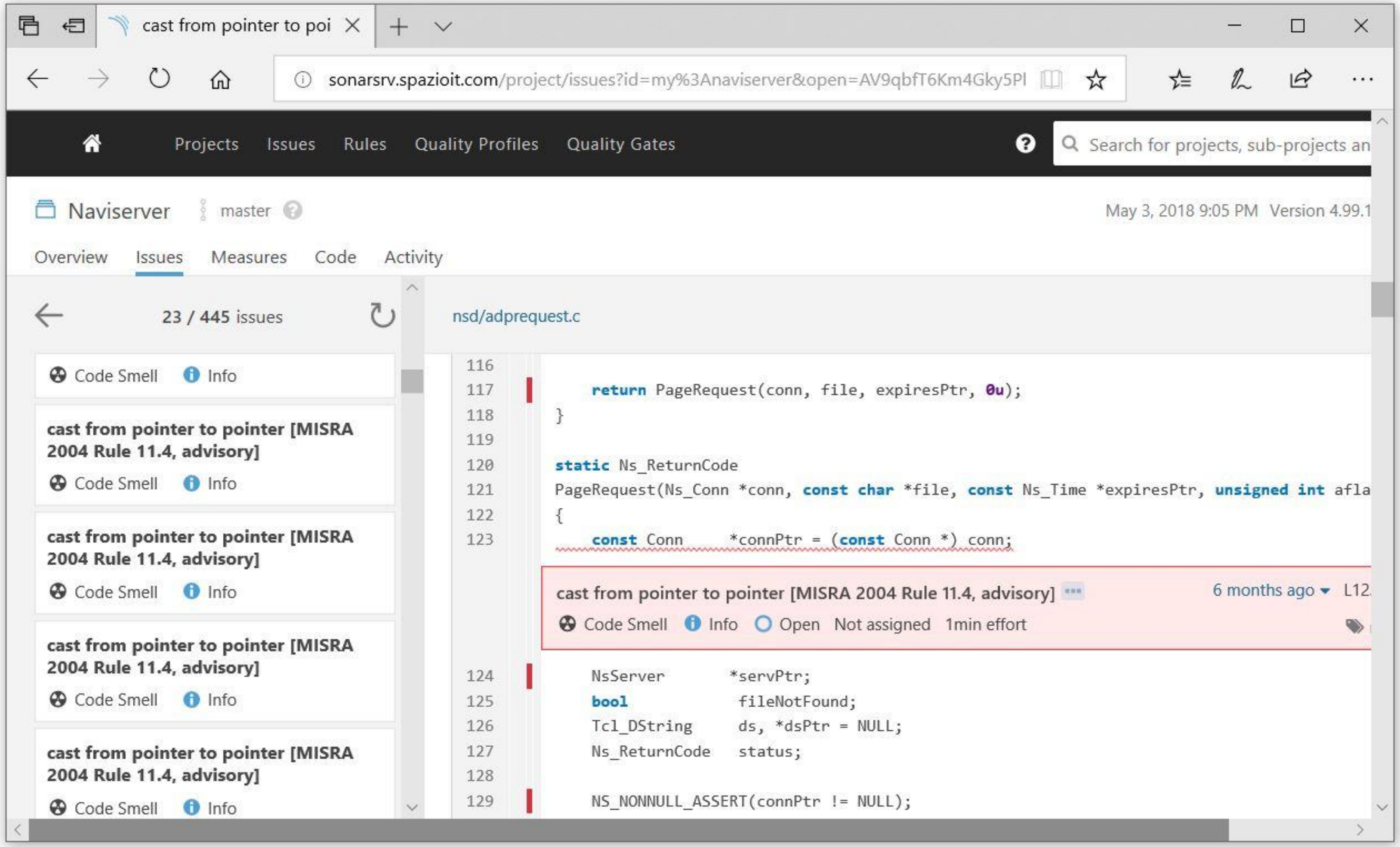


- Involved rules:
  - MISRA C 2012 – Rules 10.x – “Essential type model”
  - MISRA C 2012 – Rules 11.x – “Pointer type conversions”

- All these rules about conversions between pointer types and/or between essential types belonging to different categories or with different sizes are sound and make sense in generic, application level pieces of code. Wherever a violation is found something suspicious is taking place and most of the times it is an error.

Forbidden Conversions (and then again)





cast from pointer to poi

sonarsrv.spazioit.com/project/issues?id=my%3Anaviserver&open=AV9qbfT6Km4Gky5Pl

Naviserver master

May 3, 2018 9:05 PM Version 4.99.1

23 / 445 issues

Code Smell Info

cast from pointer to pointer [MISRA 2004 Rule 11.4, advisory]
Code Smell Info

cast from pointer to pointer [MISRA 2004 Rule 11.4, advisory]
Code Smell Info

cast from pointer to pointer [MISRA 2004 Rule 11.4, advisory]
Code Smell Info

cast from pointer to pointer [MISRA 2004 Rule 11.4, advisory]
Code Smell Info

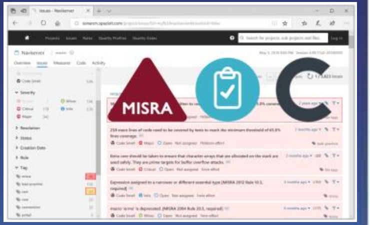
nsd/adprequest.c

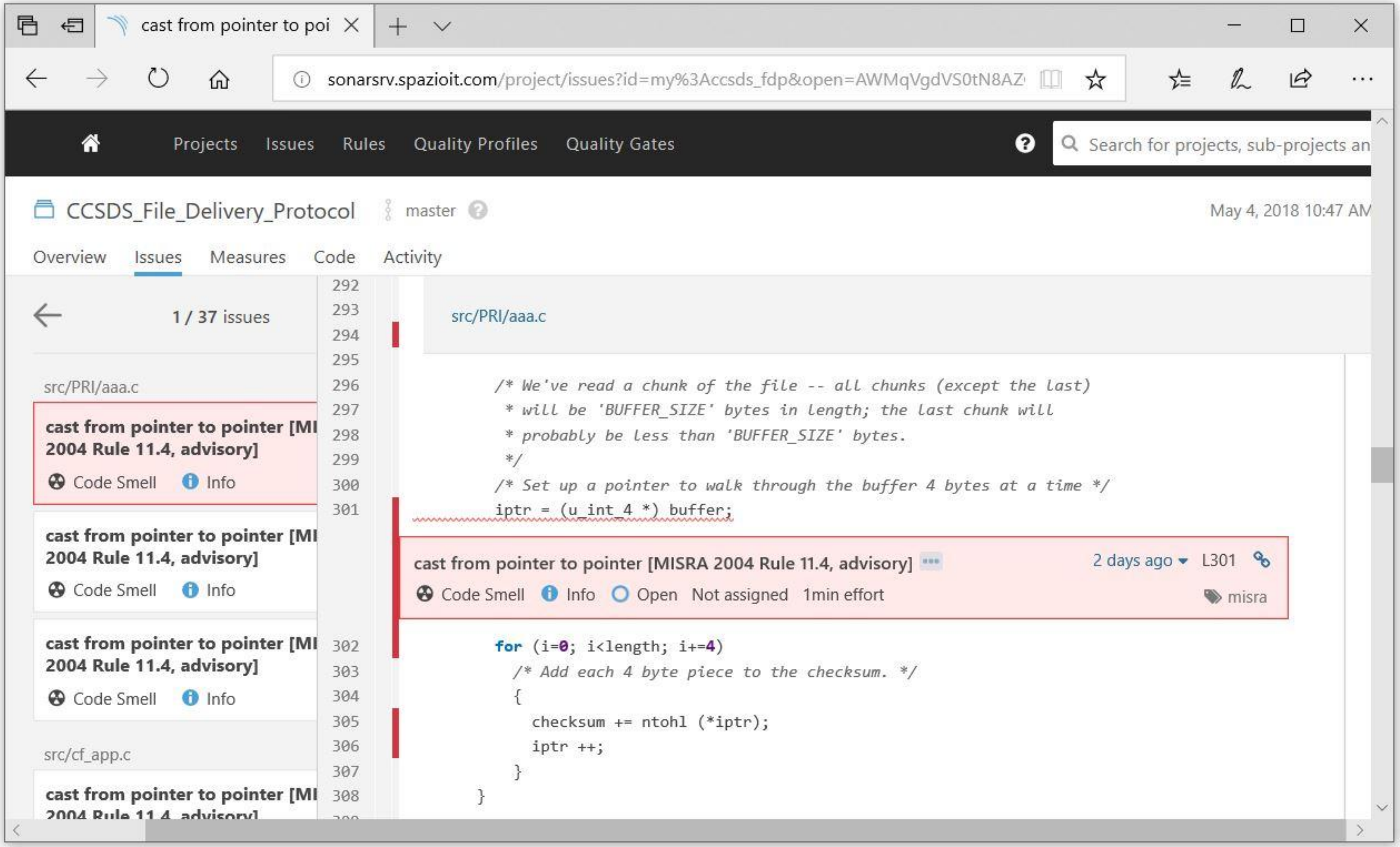
```
    return PageRequest(conn, file, expiresPtr, 0u);
}

static Ns_ReturnCode
PageRequest(Ns_Conn *conn, const char *file, const Ns_Time *expiresPtr, unsigned int afla
{
    const Conn      *connPtr = (const Conn *) conn;
```

cast from pointer to pointer [MISRA 2004 Rule 11.4, advisory] 6 months ago L12
Code Smell Info Open Not assigned 1min effort

```
    NsServer       *servPtr;
    bool            fileNotFound;
    Tcl_DString     ds, *dsPtr = NULL;
    Ns_ReturnCode   status;

    NS_NONNULL_ASSERT(connPtr != NULL);
```

Forbidden Conversions (and then again)





cast from pointer to pointer [MI
2004 Rule 11.4, advisory]
Code Smell Info

cast from pointer to pointer [MI
2004 Rule 11.4, advisory]
Code Smell Info

cast from pointer to pointer [MI
2004 Rule 11.4, advisory]
Code Smell Info

cast from pointer to pointer [MI
2004 Rule 11.4, advisory]

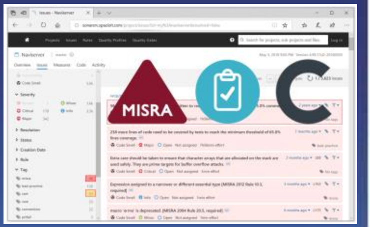
src/PRI/aaa.c

```
     /* We've read a chunk of the file -- all chunks (except the last)
      * will be 'BUFFER_SIZE' bytes in length; the last chunk will
      * probably be less than 'BUFFER_SIZE' bytes.
      */
     /* Set up a pointer to walk through the buffer 4 bytes at a time */
     iptr = (u_int_4 *) buffer;
```

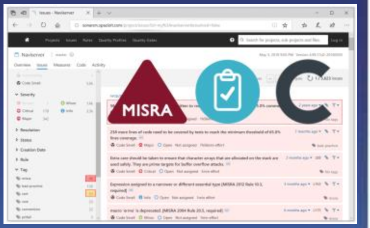
cast from pointer to pointer [MISRA 2004 Rule 11.4, advisory] — 2 days ago — L301
Code Smell · Info · Open · Not assigned · 1min effort · misra

```
     for (i=0; i<length; i+=4)
       /* Add each 4 byte piece to the checksum. */
       {
         checksum += ntohl (*iptr);
         iptr ++;
       }
   }
```

MISRA and CERT compliance standardization efforts



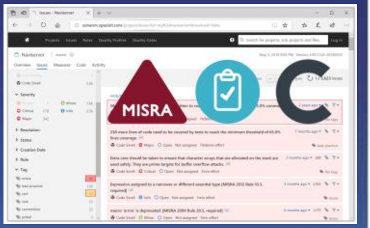
MISRA compliance standardization efforts



- MISRA guidelines are divided in:
  - directives: guidelines which are not defined with reference to the source code alone, but which also refer to, or impose requirements on processes and documentation;
  - rules: guidelines which impose requirements on the source code and the source code only.

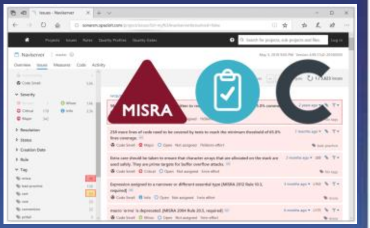
- Rules are then divided into:
  - decidable: rules that can always be assessed, verified by a (properly configured) analysis tool;
  - undecidable: rules that cannot be assessed, verified by an analysis tool in every situation.

MISRA compliance standardization efforts



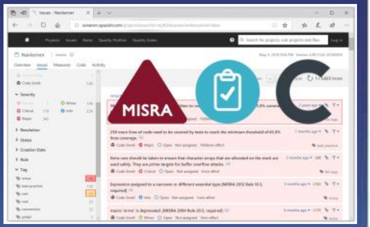
- Guidelines are categorized into:
  - mandatory: guidelines for which violation is never permitted;
  - required: guidelines for which violations are permitted if justified by documented deviations
  - advisory: guidelines that can be violated without the need of a corresponding deviation.
- Resulting conclusions:
  - Mandatory guidelines cannot have deviations.
  - Among the required guidelines decidable rules are the less expensive to verify while undecidable rules and directives are more expensive.
  - Advisory guidelines can be followed till it make sense, till it is reasonably practical to do so.

CERT compliance standardization efforts



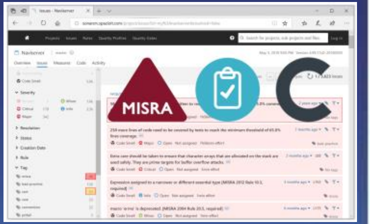
- CERT guidelines are divided into:
  - recommendations: guidelines that are likely to improve the quality of the system;
  - rules: guidelines whose violations are likely to introduce defects which may adversely affect the safety, reliability, or security of the system.
- Each guideline contains a risk assessment based on its severity, likelihood and remediation cost; all these attributes are expressed as a number ranging from 1 to 3.
  - severity: 1 – low, 2 – medium, 3 – high
  - likelihood: 1 – unlikely, 2 – probable, 3 – likely
  - remediation cost: 1 – high, 2 – medium, 3 – low

CERT compliance standardization efforts



- The risk associated to a guideline is the product of these three attributes, which is called priority.
  - Though the product ranges from 1 to 27, only the following 10 distinct values are present in the document: 1, 2, 3, 4, 6, 8, 9, 12, 18, and 27.
- This scheme allows the definition of levels:
  - L1: guidelines with priority from 12 to 27
    high severity, likely and inexpensive to fix
  - L2. guidelines with priority from 6 to 9
    medium severity, probable and medium cost to fix;
  - L3: guidelines with priority from 1 to 4
    low severity, unlikely and expensive to fix

Analysis Tools



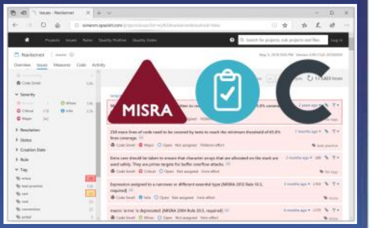
pc-lint

frama C

Software Analyzers

POLYSPACE

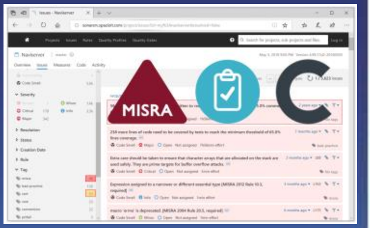
sonarqube



Analysis tools

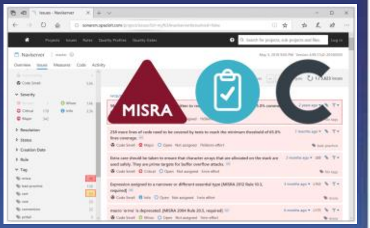
- Analysis tools can be divided into three broad categories:
  - "Shallow Analysers": based on patterns matching, e.g. "PC-Lint", "splint", "Understand", "Cppcheck", …
  - "Deep Analysers": based on techniques like bounded model checking, semantic analysis and abstract execution, e.g. "CBMC", "Frama-C", "Polyspace", …
  - "Compiler Based Analysers": based on the analyses performed by the compilers themselves, e.g. "Clang Static Analyzer", "Facebook Infer", …
- *"Shallow Analyser"* are usually able to verify/assess the compliance of the majority of MISRA and CERT guidelines. In some cases, for some specific guidelines, "*Deep Analysers*" or manual intervention may be required.

Analysis tools

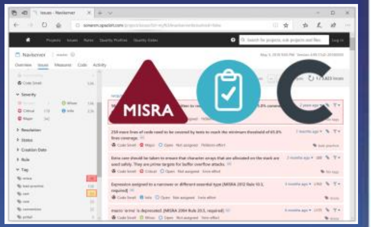


- The problem with “*Deep Analysers*” is that they require a lot of computing resources to perform their analyses and this might compromise their applicability to large codebases.
- A recent, very interesting trend in static analysis is the adoption of “*Compiler Based Analysers*”; they offer several advantages, among which the most important are:
  - they are the “real thing”, the very same tools used to build the software under analysis;
  - they are fast and can easily analyse large codebases;
  - they are easy to use, especially by developers and testers who are already accustomed to the compilation toolsets.

A practical approach



Guidelines Selection



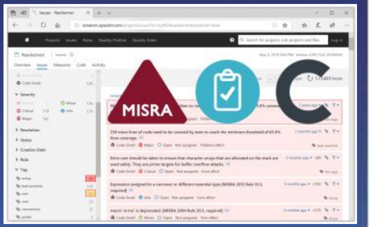
MISRA

- Decidable Rules
- Undecidable Rules
- Directives

CERT

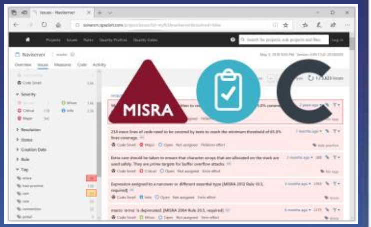
- L1 Guidelines
- L2 Guidelines
- L3 Guidelines

Steps



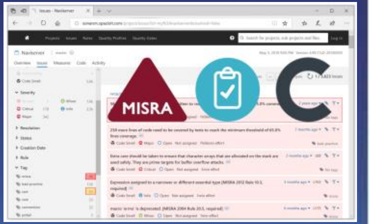
- System partitioning (based on criticality, but also on the “distance from the hardware”).
- Tools configuration (reducing “false-negatives” and “false positives”)
- (continuous) Analysis Execution

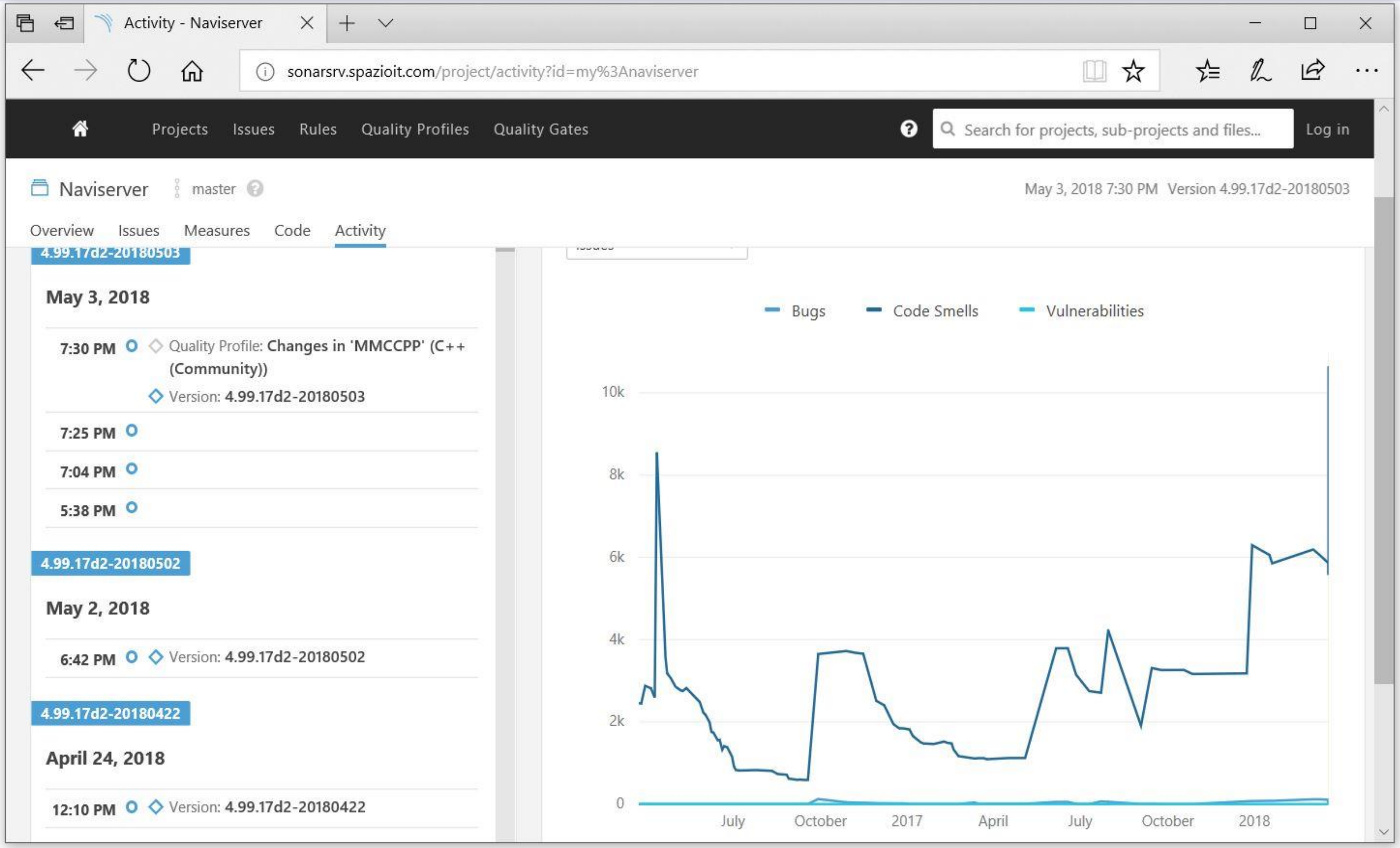
Results Exploitation



- No “Quality-Stamps” but “Hot-Spots” in the codebase
- Training (transfer of knowledge and experience)

Results Exploitation





Activity - Naviserver

sonarsrv.spazioit.com/project/activity?id=my%3Anaviserver

Projects Issues Rules Quality Profiles Quality Gates
Search for projects, sub-projects and files...
Log in
Naviserver
master
May 3, 2018 7:30 PM Version 4.99.17d2-20180503
Overview Issues Measures Code Activity

4.99.17d2-20180503

May 3, 2018

7:30 PM Quality Profile: Changes in 'MMCCPP' (C++ (Community))
Version: 4.99.17d2-20180503

7:25 PM

7:04 PM

5:38 PM

4.99.17d2-20180502

May 2, 2018

6:42 PM Version: 4.99.17d2-20180502

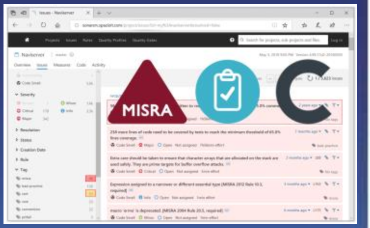
4.99.17d2-20180422

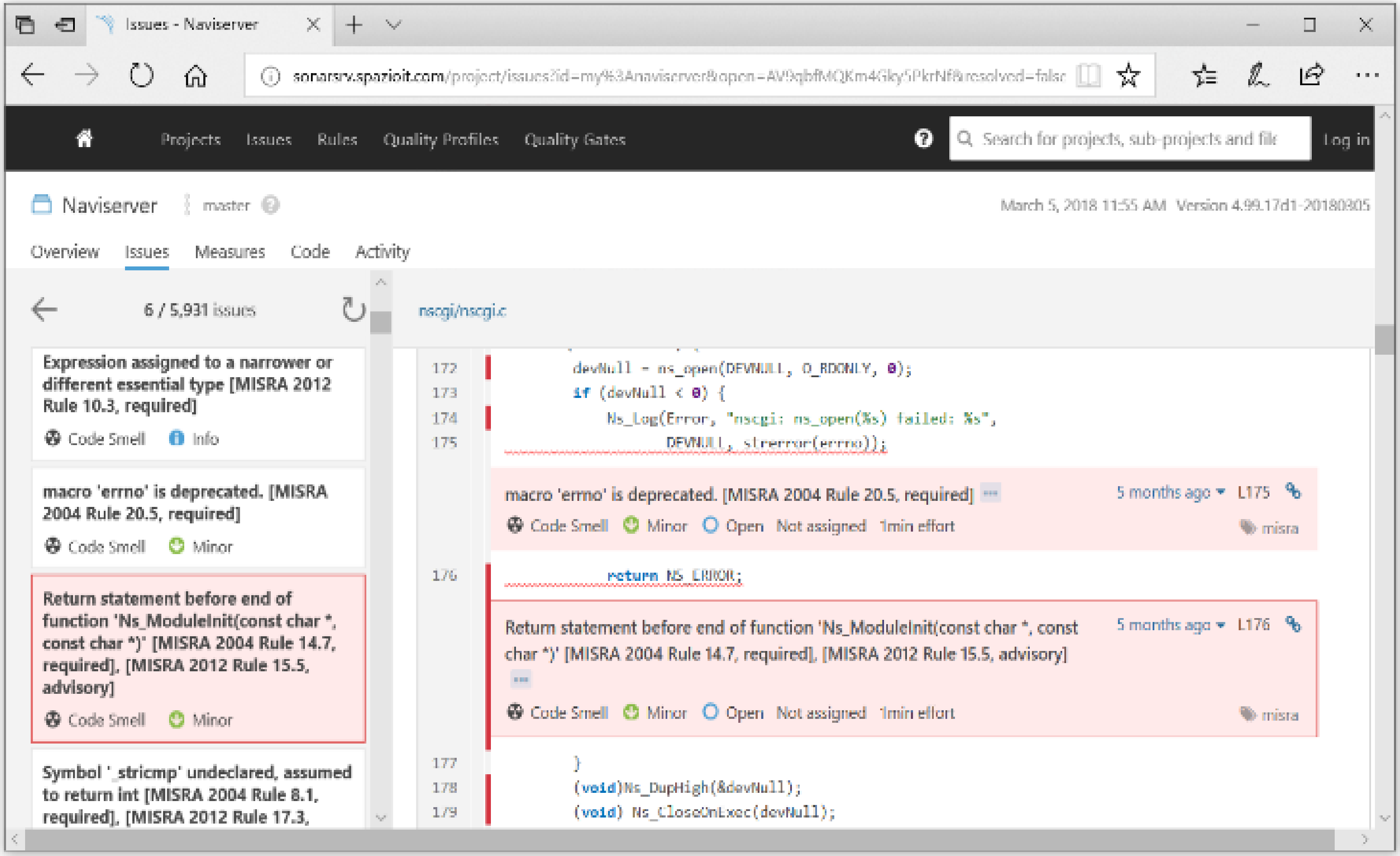
April 24, 2018

12:10 PM Version: 4.99.17d2-20180422

Bugs
Code Smells
Vulnerabilities
10k
8k
6k
4k
2k
0
July
October
2017
April
July
October
2018

Results Exploitation





Issues - Naviserver

sonarsrv.spazioit.com/project/issues?id=my%3Anaviserver&open=AV9qbfMQKm4GkySPkrNf&resolved=false

Naviserver master

March 5, 2018 11:55 AM Version 4.99.17d1-20180305

6 / 5,931 issues

Expression assigned to a narrower or different essential type [MISRA 2012 Rule 10.3, required]
Code Smell Info

macro 'errno' is deprecated. [MISRA 2004 Rule 20.5, required]
Code Smell Minor

Return statement before end of function 'Ns_ModuleInit(const char *, const char *)' [MISRA 2004 Rule 14.7, required], [MISRA 2012 Rule 15.5, advisory]
Code Smell Minor

Symbol '_stricmp' undeclared, assumed to return int [MISRA 2004 Rule 8.1, required], [MISRA 2012 Rule 17.3,

nscgi/nscgi.c

```
172        devNull = ns_open(DEVNULL, O_RDONLY, 0);
173        if (devNull < 0) {
174            Ns_Log(Error, "nscgi: ns_open(%s) failed: %s",
175                   DEVNULL, strerror(errno));
```

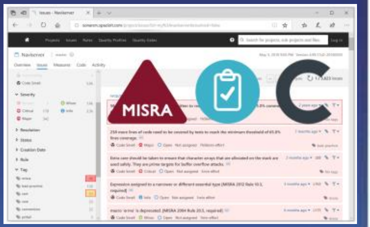
macro 'errno' is deprecated. [MISRA 2004 Rule 20.5, required] 5 months ago L175
Code Smell Minor Open Not assigned 1min effort misra

```
176            return NS_ERROR;
```

Return statement before end of function 'Ns_ModuleInit(const char *, const char *)' [MISRA 2004 Rule 14.7, required], [MISRA 2012 Rule 15.5, advisory] 5 months ago L176
Code Smell Minor Open Not assigned 1min effort misra

```
177        }
178        (void)Ns_DupHigh(&devNull);
179        (void) Ns_CloseOnExec(devNull);
```

References



- MISRA C:2012, “Guidelines for The Use of The C Language in Vehicle Based Software”, March 2013, The Motor Industry Software Reliability Association, ISBN 978-1-906400-11-8
- SEI CERT C, “SEI CERT C Coding Standard - Rules for Developing Safe, Reliable, and Secure Systems”, 2016 Edition, Software Engineering Institute, Carnegie Mellon University
- MISRA COMPLIANCE:2016, “Achieving compliance with MISRA Coding Guidelines”, April 2016, The Motor Industry Software Reliability Association, ISBN 978-1-906400-13-2
- [SEI CERT C, “SEI CERT C Coding Standard” – Web Version: https://wiki.sei.cmu.edu/confluence/display/c/SEI+CERT+C+Coding+Standard
- http://www.sonarqube.org
- http://sonarsrv.spazioit.com/projects

Thank you for your attention!