TINTORERA

Attack Surface Intelligence of Source Code

#HITB2014AMS
De Beurs van Berlage

The 5th Annual Hack In The Box Security Conference in The Netherlands
ME & VULNEX

Simon Roses Femerling

- Founder & CEO, VULNEX [www.vulnex.com](http://www.vulnex.com)
- @simonroses
- Former Microsoft, PwC, @Stake
- Black Hat, RSA, OWASP, SOURCE, AppSec, DeepSec, TECHNET

VULNEX

- CyberSecurity Startup
- @vulnexsl
- Services & Training
- Products: BinSecSweeper (Binary Analysis)
TALK OBJECTIVES

• GCC & Python, hand to hand

• Transformations: source code to useful data

• Practical code understanding
WORK IN PROGRESS
AGENDA

1. The need of Attack Surface Intelligence of Source Code
2. GCC Overview
3. GCC-Python-Plugin
4. Source Code Intelligence
5. Tintorera Overview
6. Tintorera Analysis Demos
7. Conclusions
8. Q&A
1. The need of Attack Surface Intelligence of Source Code
## 1. CODE IS GETTING COMPLEX!

<table>
<thead>
<tr>
<th>Software</th>
<th>SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>14 Million</td>
</tr>
<tr>
<td>Windows Server 2003</td>
<td>50 Million</td>
</tr>
<tr>
<td>Debian 7.0</td>
<td>419 Million</td>
</tr>
<tr>
<td>Mac OS X 10.4</td>
<td>86 Million</td>
</tr>
<tr>
<td>Linux Kernel 2.6.25</td>
<td>13.5 Million</td>
</tr>
<tr>
<td>Linux Kernel 3.6</td>
<td>15.9 Million</td>
</tr>
</tbody>
</table>
1. DOCUMENTATION

MISSING
IT Documentation

Our server crashed and we’re not super jazzed about it. We don’t know where our backups are, but are pretty sure they exist. Could really use some IT documentation right about now.

CAN YOU HELP?
Please call 778-555-6666
1. TYPICAL CODE REVIEW
1. WHERE TO START?

- File operations
- Networking
- Processes
- Crypto
- Authentication
- ??
1. TOOLS?

grep

rough-auditing-tool-for-security
Rough Auditing Tool for Security (RATS)

eclipse
2. GCC Overview
2. GCC

- Compiler system that supports various programming languages
- Popular UNIX variants
- Supports all major languages: C, C++, Java, Objective-C, etc.
- PLUGINS!!
- FREE
2. GCC INTERNALS

2. GCC TERMINOLOGY

- GENERIC is common representation shared by all front ends
  - Each parser must emit GENERIC

- GIMPLE is a simplified version of GENERIC
  - 3 address representation
  - Simplified control flow

- RTL (Register Transfer Language), assembler for an abstract machine
2. GCC PASSES

The lowering passes

*warn_unused_result
*diagnose_omp_blocks
mudflap1
omp_lower
lower
ehopt
eh
cfg
*warn_function_return
*build_cgraph_edges

The "small IPA" passes

*free_lang_data
visibility
early_local_cleanups
*free_cfg_annotations
*init_datastructures
ompexpg
*referenced_vars
ssa
veclower
*early_warn_uninitialized
*rebuild_cgraph_edges
inline_param
einline
early_optimizations
*remove_cgraph callees_edges
copyrename
ccp
forwprop
ealias
esra
copyprop
mergephi
cdce
eiga_sra
tailr
switchconv
ehclean-up
profile
local-pure-const
fnsplit
release ssa
*rebuild_cgraph_edges
inline_param
tree_profile_ipa
feedback_fnsplit

3. GCC-Python-Plugin
3. GCC-PYTHON-PLUGIN

• GCC plugin that embeds Python in GCC 😊

• Now your Python script can access GCC passes and perform analysis

• Developed by David Malcolm (Fedora)

# Copyright 2011 David Malcolm <dmalcolm@redhat.com>
# Copyright 2011 Red Hat, Inc.

# This is free software: you can redistribute it and/or modify it
# under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
# General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see
# <http://www.gnu.org/licenses/>.

# Sample python script, to be run by our gcc plugin
# Show all the passes that get executed

import gcc

def my_pass_execution_callback(*args, **kwargs):
    (optpass, fun) = args
    print(args)

gcc.register_callback(gcc.PLUGIN_PASS_EXECUTION,
                      my_pass_execution_callback)
3. GCC-PYTHON-PLUGIN DEMO
3. GCC-PYTHON-PLUGIN IDEAS

• Write scripts for:
  – malloc/free usage
  – Array boundary checks
  – Code visualizations
  – You name it!
4. Source Code Intelligence
4. CODE UNDERSTATING

- What API are being used?
- Number of functions?
- Inputs / Outputs of functions?
- Function relationship
- What comments said?
- Code complexity
4. CODE METRICS

• Controversial topic but needed

• Metrics:
  – Function complexity (Cyclomatic)
  – Number of:
    • Lines
    • Code
    • Blanks
    • Comments
  – Line Length
  – Number: Bugs per Line
  – You name it....
4. CODE COMPLEXITY

- Counts the number of linearly independent paths through the source code
- Basically we can have an idea of the complexity of functions
- Complexity is security enemy!
- Created by Thomas McCabe
### 4. CODE COMPLEXITY THRESHOLD

<table>
<thead>
<tr>
<th>Cyclomatic Complexity</th>
<th>Risk Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>a simple program, without much risk</td>
</tr>
<tr>
<td>11-20</td>
<td>more complex, moderate risk</td>
</tr>
<tr>
<td>21-50</td>
<td>complex, high risk program</td>
</tr>
<tr>
<td>greater than 50</td>
<td>untestable program (very high risk)</td>
</tr>
</tbody>
</table>

http://www.sei.cmu.edu/reports/97hb001.pdf
4. SOURCE CODE ANALYSIS FLOWGRAPH NOTATION

If .. then

If .. then .. else

If .. and .. then

If .. or .. then

Do .. While

While .. Do

Switch
4. SOURCE CODE VISUALS TOO

**Binary Code**

```
xan_1000_validate_struct:
    mov  eax, [esp+xan]
test  eax, eax
jz  short loc_90001015
```

**Source Code**

```c
BLOCK 0
entry

BLOCK 2
data = 0B;
69  data = NULL;
    data = 0B;

BLOCK 3
source:
D.4317 = operator new [] (400);
data = D.4317;
72  data = new long[100];
D.4317 = operator new [] (400);
data = D.4317;

BLOCK 4
sink:
    if (data != 0B)
sink:
        if (data != 0B)
true

BLOCK 5
operator delete [] (data);
77  delete [] data;
operator delete [] (data);
false

BLOCK 6
return;
78  return;

BLOCK 1
exit
```
5. Tintorerera Overview
5. TINTORERA – BLUE SHARK

• “Put source code into context”

• Objective: Get a feeling of the code while compiling!!

• Intelligence of source code:
  – Code visualizations
  – Comments analysis
  – API identification
  – Metrics
  – HTML Reports

• C code transformed to JSON files, now you can query and perform analysis on data
5. TINTORERA INTERNALS

- Two files:
  - analyzer.py: To be used while compiling a project
  - do_report_tintorera.py: Use after project has been compiled to generate report

- Composed of:
  - Python code
  - JSON data files
  - HTML / CSS / Javascript
5. TINTORERA STRUCTURE

- Python files
- Folders:
  - data/ : API JSON file
  - templates/ : HTML templates
  - js/ : Javascript code
  - images/
  - Tintorera_lib/ : python code
5. TINTORERA INSTALL & USAGE

1. GCC version 4.7 or later

2. Install gcc-python-plugin (See web doc)

3. Set path:
   1. Export LD_LIBRARY_PATH=/gcc-python-plugin/gcc-c-api

4. Add line to Makefile (CC= tag)
   1. gcc -fplugin=/gcc-python-plugin/python.so -fplugin-arg-python-script=./tintorera/analyzer.py

5. Run make

6. After compile use:
   1. Python do_report_tintorera.py –c tinan.cfg
5. TINTORERA CONFIG FILE

• Edit tinan.cfg to suit your needs

• Set parameters such as:
  – Folder to save analysis report
  – Enable / disable analysis
    • Basic blocks
    • Callgraphs
    • Comments
    • Gimples
    • Etc.
  – Cyclomatic Thresholds
5. TINTORERA DATA FILES

- Folder: /data
- File: tinto_api.json
- JSON file to define APIs
5. CODE TRANSFORMATION

SOURCE CODE → JSON FILES

HTML REPORT
5. TRANSFORMED JSON FILES

- 3 files:
  1. tintorera_bb_file.json: code basic blocks
  2. tintorera_meta_info.json: general information, file size and code & comments not inside functions
  3. tintorera_temp_file.json: functions information
5. TINTORERA_BB_FILE.JSON

```json
{
  "file_name": "loop_tester.c",
  "func_gimples": [
    {
      "bb_data": "gcc.BasicBlock(index=0)",
      "bb_0": 0,
      "desc": "entry"
    },
    {
      "bb_1": 1,
      "bb_data": "gcc.BasicBlock(index=1)",
      "desc": "exit"
    },
    {
      "bb_2": 2,
      "bb_data": [
        {
          "loc": "loop_tester.c:132",
          "code": " no_loop();",
          "repr(stmt)": "gcc.GimpleCall()",
          "exprcode": "<type 'gcc.CallExpr'>",
          "str(stmt)": "no_loop ()",
          "rhs": "[<gcc.AddrExpr object at 0xaf024e8>, None]",
          "lhs": "None",
          "exprtype": "<gcc.VoidType object at 0xaf02c98>"
        },
        {
          "loc": "loop_tester.c:133",
          "code": " if_stmt();",
          "repr(stmt)": "gcc.GimpleCall()",
          "exprcode": "<type 'gcc.CallExpr'>",
          "str(stmt)": "if_stmt ()",
          "rhs": "[<gcc.AddrExpr object at 0xaf02518>, None]",
          "lhs": "None",
          "exprtype": "<gcc.VoidType object at 0xaf02c98>"
        },
        {
          "loc": "loop_tester.c:134",
          "code": " if_stmt();",
          "repr(stmt)": "gcc.GimpleCall()",
          "exprcode": "<type 'gcc.CallExpr'>",
          "str(stmt)": "if_stmt ()",
          "rhs": "[<gcc.AddrExpr object at 0xaf02518>, None]",
          "lhs": "None",
          "exprtype": "<gcc.VoidType object at 0xaf02c98>"
        }
      ]
    }
  ]
}
```
```json
{
    "file_name": "loop_tester.c",
    "file_size": 1541,
    "file_comments": [],
    "file_comments_len": 0,
    "file_lines_len": [
        { "line": 0, "len": 19 },
        { "line": 1, "len": 1 },
        { "line": 2, "len": 19 }
    ],
    "file_loc": 3,
    "file_len": 2,
    "file_blank": 1,
    "file_metrics": {
        "total_code": 127,
        "line_max": "38",
        "line_min": "1",
        "total_bb": 74,
        "total_cc": 27,
        "cc_avg": 2,
        "total_funcs": 13,
        "total_lines": 136,
        "line_avg": "11.46666666667",
        "total_blanks": 8,
        "total_comments": 0
    },
    "file_sha256": "",
    "file_code": [
        "#include <stdio.h>\n",
        "\n",
        "void no_loop(void)\n"
    ]
}
```
```json
{
    "func_inline_asm": "",
    "file_name": "loop_tester.c",
    "func_api_res": [],
    "func_loc": 16,
    "func_code": [
        "int main(int *argc, char *argv[]) \n",
        "{\n",
        " \n",
        "     no_loop();\n",
        "     if_stmt();\n",
        "     if_else_stmt();\n",
        "     if_and_stmt();\n",
        "     if_or_stmt();\n",
        "     if_and_else_stmt();\n",
        "     if_or_else_stmt();\n",
        "     while_stmt();\n",
        "     for_stmt(); \n",
        "     do_while_stmt();\n",
        "     switch_stmt();\n",
        "     goto_stmt();\n",
        "\n",
        "     return 0;\n",
        "}\n"
    ],
    "func_comments_len": 0,
    "func_ploc": 18,
    "func_count_gimples": {
        "gimplecall": 12,
        "gimpleasm": 0,
        "gimplereturn": 1,
        "gimplenop": 0,
        "gimplephi": 0,
        "gimplecond": 0,
        "gimplerswitch": 0,
        "gimpleassign": 1,
        "gimplelabel": 1
    },
    "func_end_line": 147,
    "func_decl": {
```
5. TINTORERA SOURCE CODE METRICS

• Current metrics:

1. Number of:
   1. Lines
   2. Code
   3. Blanks
   4. Comments
   5. Colons

2. Average line length
3. Minimum line
4. Maximum line
5. Total Basic Blocks
6. Total Cyclomatic Complexity
7. Average Cyclomatic Complexity
5. SOURCE CODE COMMENT ANALYSIS
6. Tintorera Analysis Demos
6. DEMO I: LOOP TESTER
6. DEMO I: LOOP TESTER
6. DEMO I: LOOP TESTER

IF ELSE

WHILE

SWITCH
6. DEMO II: SENDMAIL CRACKADDR (CVE2002-1337)

Pure Complexity....
6. DEMO II: SENDMAIL CRACKADDR (CVE2002-1337)  
FUNCTION COMPLEXITY

Function = sendmail_crackaddr_cve2002_1337.c -> crackaddr

Function Details

<table>
<thead>
<tr>
<th>Function Name</th>
<th>crackaddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments (1)</td>
<td>[u'char *]</td>
</tr>
<tr>
<td>Return Type</td>
<td>char *</td>
</tr>
<tr>
<td>Function LOC</td>
<td>247</td>
</tr>
<tr>
<td>Function Physical LOC</td>
<td>334</td>
</tr>
<tr>
<td>Function Start Line</td>
<td>56</td>
</tr>
<tr>
<td>Function End Line</td>
<td>390</td>
</tr>
<tr>
<td>Comments</td>
<td>54</td>
</tr>
<tr>
<td>Blank Lines</td>
<td>33</td>
</tr>
</tbody>
</table>

Code Details & Metrics

<table>
<thead>
<tr>
<th>Basic Blocks</th>
<th>155</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Complexity</td>
<td>89</td>
</tr>
<tr>
<td>Metrics-&gt; Count Colons</td>
<td>125</td>
</tr>
</tbody>
</table>
6. DEMO II: SENDMAIL CRACKADDR (CVE-2002-1337)
FUNCTION COMPLEXITY
Mongoose is the most easy to use web server on the planet. A web server of choice for Web developers (PHP, Ruby, Python, etc) and Web designers.
### Application Summary

<<

### Application Details

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Files</td>
<td>2</td>
</tr>
<tr>
<td>Total Functions</td>
<td>156</td>
</tr>
<tr>
<td>Total Basic Blocks</td>
<td>1930</td>
</tr>
<tr>
<td>Total LOC</td>
<td>3507</td>
</tr>
<tr>
<td>Total Physical LOC</td>
<td>4306</td>
</tr>
<tr>
<td>Total Comments</td>
<td>337</td>
</tr>
<tr>
<td>Total Blanks</td>
<td>460</td>
</tr>
</tbody>
</table>
6. DEMO III: MONGOOSE WEB SERVER ANALYSIS

View of API used in application

- **insecure**: 41.7%
- **stdio.h**: 13.1%
- **process**: 7.1%
- **network**: 5.4%
- **inter_com**: 1.2%
- **file**: 22.6%
# 6. DEMO III: MONGOOSE WEB SERVER ANALYSIS

## Files Results

<table>
<thead>
<tr>
<th># All Funcs</th>
<th># File Funcs</th>
<th>File Name</th>
<th>Function Name</th>
<th>Basic Blocks</th>
<th>Cyclomatic Complexity</th>
<th>API Calls</th>
<th>Inline ASM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>main.c</td>
<td>main</td>
<td>7</td>
<td>2</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>main.c</td>
<td>start_mongoose</td>
<td>17</td>
<td>9</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>main.c</td>
<td>mongoose_callback</td>
<td>6</td>
<td>2</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>main.c</td>
<td>init_server_name</td>
<td>3</td>
<td>1</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>main.c</td>
<td>process_command_line_arguments</td>
<td>31</td>
<td>15</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>main.c</td>
<td>set_option</td>
<td>13</td>
<td>6</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>main.c</td>
<td>sdup</td>
<td>6</td>
<td>2</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>main.c</td>
<td>verify_document_root</td>
<td>9</td>
<td>5</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>main.c</td>
<td>show_usage_and_exit</td>
<td>9</td>
<td>3</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>main.c</td>
<td>die</td>
<td>3</td>
<td>1</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>main.c</td>
<td>signal_handler</td>
<td>3</td>
<td>1</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>mongoose.c</td>
<td>mg_start</td>
<td>34</td>
<td>17</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>mongoose.c</td>
<td>mg_stop</td>
<td>6</td>
<td>2</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>mongoose.c</td>
<td>free_context</td>
<td>14</td>
<td>6</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>mongoose.c</td>
<td>master_thread</td>
<td>19</td>
<td>8</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>mongoose.c</td>
<td>accept_new_connection</td>
<td>7</td>
<td>3</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>
Function = main.c -> main

Function Details

<table>
<thead>
<tr>
<th>Function Name</th>
<th>main</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments (2)</td>
<td>[u'int', u'char **']</td>
</tr>
<tr>
<td>Return Type</td>
<td>int</td>
</tr>
<tr>
<td>Function LOC</td>
<td>16</td>
</tr>
<tr>
<td>Function Physical LOC</td>
<td>17</td>
</tr>
<tr>
<td>Function Start Line</td>
<td>882</td>
</tr>
<tr>
<td>Function End Line</td>
<td>899</td>
</tr>
<tr>
<td>Comments</td>
<td>0</td>
</tr>
<tr>
<td>Blank Lines</td>
<td>1</td>
</tr>
</tbody>
</table>

Code Details & Metrics

<table>
<thead>
<tr>
<th>Basic Blocks</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Complexity</td>
<td>2</td>
</tr>
<tr>
<td>Metrics-&gt; Count Colons</td>
<td>9</td>
</tr>
</tbody>
</table>
Boa, a high performance web server for Unix-alike computers
6. DEMO IV: BOA WEB SERVER

Application Summary

Application Details

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Files</td>
<td>24</td>
</tr>
<tr>
<td>Total Functions</td>
<td>182</td>
</tr>
<tr>
<td>Total Basic Blocks</td>
<td>2714</td>
</tr>
<tr>
<td>Total LOC</td>
<td>6237</td>
</tr>
<tr>
<td>Total Physical LOC</td>
<td>8596</td>
</tr>
<tr>
<td>Total Comments</td>
<td>1470</td>
</tr>
<tr>
<td>Total Blanks</td>
<td>865</td>
</tr>
</tbody>
</table>
6. DEMO IV: BOA WEB SERVER

View of API used in application

- process: 31.9%
- stdio.h: 28.1%
- insecure: 16.3%
- file: 8.0%
- network_inter_com: 6.9%
- generic: 13.1%
6. DEMO IV: BOA WEB SERVER

![Diagram showing the total count of gimples for different operations.](image-url)
### Files Results

<table>
<thead>
<tr>
<th># All Funcs</th>
<th># File Funcs</th>
<th>File Name</th>
<th>Function Name</th>
<th>Basic Blocks</th>
<th>Cyclomatic Complexity</th>
<th>API Calls</th>
<th>Inline ASM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>index_dir.c</td>
<td>main</td>
<td>11</td>
<td>4</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>index_dir.c</td>
<td>index_directory</td>
<td>58</td>
<td>26</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>index_dir.c</td>
<td>stat</td>
<td>3</td>
<td>1</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>index_dir.c</td>
<td>select_files</td>
<td>6</td>
<td>2</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>index_dir.c</td>
<td>send_error</td>
<td>11</td>
<td>2</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>index_dir.c</td>
<td>http_escape_string</td>
<td>24</td>
<td>11</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>index_dir.c</td>
<td>html_escape_string</td>
<td>16</td>
<td>5</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>timestamp.c</td>
<td>timestamp</td>
<td>3</td>
<td>1</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>select.c</td>
<td>loop</td>
<td>42</td>
<td>23</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>select.c</td>
<td>fdset_update</td>
<td>40</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>sublog.c</td>
<td>open_gen_fd</td>
<td>8</td>
<td>3</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>sublog.c</td>
<td>open_net_fd</td>
<td>15</td>
<td>6</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>sublog.c</td>
<td>open_pipe_fd</td>
<td>12</td>
<td>5</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>util.c</td>
<td>parse_debug</td>
<td>19</td>
<td>9</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>util.c</td>
<td>print_debug_usage</td>
<td>6</td>
<td>2</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>util.c</td>
<td>strlower</td>
<td>6</td>
<td>2</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>
## 6. DEMO IV: BOA WEB SERVER

### Code Details & Metrics

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Start Line</td>
<td>46</td>
</tr>
<tr>
<td>Function End Line</td>
<td>135</td>
</tr>
<tr>
<td>Comments</td>
<td>21</td>
</tr>
<tr>
<td>Blank Lines</td>
<td>10</td>
</tr>
</tbody>
</table>

### Basic Blocks

- **Basic Blocks**: 42

### Code Complexity

- **Code Complexity**: 23

### Metrics-> Count Colons

- **Count Colons**: 24

### API Details

<table>
<thead>
<tr>
<th>Category</th>
<th>API</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>generic</td>
<td>close</td>
<td>select.c:70</td>
</tr>
</tbody>
</table>

### Inline ASM Details

```asm
__asm__ volatile__("cltd rep; stosl": "c" _d0, "D" _d1 :
"a" 0, "0" 32, "1"
&block_read_fdset._fds_bits[0]
: "memory");
```
6. DEMO V: OBFUSCATED C CODE ANALYSIS, ENDOH4.C

The International Obfuscated C Code Contest - http://www.ioccc.org/
## Files Results

<table>
<thead>
<tr>
<th># All Funcs</th>
<th># Funcs</th>
<th>File Name</th>
<th>Function Name</th>
<th>Basic Blocks</th>
<th>Cyclomatic Complexity</th>
<th>API Calls</th>
<th>Inline ASM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>endoh4.c</td>
<td>main</td>
<td>93</td>
<td>37</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>endoh4.c</td>
<td>G</td>
<td>38</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>endoh4.c</td>
<td>o</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6. DEMO V: OBFUSCATED C CODE ANALYSIS, ENDOH4.C
6. DEMO V: OBFUSCATED C CODE ANALYSIS, ENDOH4.C

The International Obfuscated C Code Contest - http://www.ioccc.org/
6. DEMO VI: OBFUSCATED C CODE ANALYSIS, MISAKA

The International Obfuscated C Code Contest - http://www.ioccc.org/
6. DEMO VI: OBFUSCATED C CODE ANALYSIS, MISAKA

The International Obfuscated C Code Contest - http://www.ioccc.org/
6. DEMO VI: OBFUSCATED C CODE ANALYSIS, MISAKA

The International Obfuscated C Code Contest - http://www.ioccc.org/
7. Conclusions
7. DRAWBACKS

- gcc-python-plugin needs more work, fails many times
- So do Tintorera...
- Only C / C++ code
7. CONCLUSIONS

• Tintorera helps to analyze C code faster & better

• Practical code understanding for:
  – Saving time
  – Security reviews
  – Fuzzing: what and where to fuzz
7. NEXT STEPS

• Better & focused analysis (security, etc.)

• Vulnerabilities Detection

• More metrics

• Code Diff

• Cooler reports!

• Other languages ¿?
8. Q&A

- Thanks!
- @simonroses / @vulnexsl
- www.vulnex.com